Made available by Hasselt University Library in https://documentserver.uhasselt.be

An artificial intelligence course for chemical engineers

Peer-reviewed author version

Wu , M; Di Caprio, U; Vermeire, F; Hellinckx, P; BRAEKEN, Leen; Waldherr, S & Leblebici, ME (2023) An artificial intelligence course for chemical engineers. In: Education for chemical engineers, 45 , p. 141 -150.

DOI: 10.1016/j.ece.2023.09.004
Handle: http://hdl.handle.net/1942/49543

An Artificial Intelligence Course for Chemical Engineers

Min Wu[1,*], Ulderico Di Caprio[1,*], Florence Vermeire[2], Peter Hellinckx[3], Leen Braeken[1], Steffen Waldherr[2,4], M. Enis Leblebici[1,**]

[1] Center for Industrial Process Technology, Department of Chemical Engineering, KU Leuven, Agoralaan Building B, 3590 Diepenbeek, Belgium

[2] Chemical Reactor Engineering and Safety (CREaS), Leuven (Arenberg), Celestijnenlaan 200f - box 2424, 3001 Leuven, Belgium

[3] IDLab, Faculty of Applied Engineering, University of Antwerp-imec, Sint-Pietersvliet 7, 2000 Antwerp, Belgium

[4] Molecular Systems Biology (MOSYS), Department of Functional and Evolutionary Ecology, Faculty of Life Sciences, University of Vienna, 1030 Vienna, Austria

*Equal contributions

**Corresponding author (muminenis.leblebici@kuleuven.be)

Abstract

Artificial intelligence and machine learning are revolutionising fields of science and engineering. In recent years, process engineering has widely benefited from this novel modelling and optimisation approach. The open literature can offer several examples of their applications to chemical engineering problems. Increasing investments are devoted to these techniques from different industrial areas, but insufficient information on a structured course covering these topics in a chemical engineering curriculum could be found. The course in this paper intends to reduce

this gap. We introduce one of the first courses on artificial intelligence applications in a chemical engineering curriculum. The course targets Master's students with a chemical engineering background and insufficient knowledge of statistical approaches. It covers the main aspects by utilising frontal lectures and hands-on exercises with active learning methods. This paper shows the methodology we adapted to introduce students to machine learning techniques and how they responded to each class. The student performances for each test are shown, as well as the survey results based on student feedback and suggestions. This work contains essential guidelines for educators who will provide an artificial intelligence course in a chemical engineering curriculum.

Keywords: modelling, optimisation, chemical engineering, artificial intelligence, python

1. Introduction

AI[1] and ML[2] are revolutionising modern societies, giving new tools to study and understand the behaviours of complex systems. These techniques allow catching patterns within a dataset, supporting scientists and engineers in the analysis and optimisation of physical systems. The related digitalisation via AI has already influenced the chemical industry, and they are likely to get more substantial over time (Udugama et al., 2022). For this reason, the usage of ML techniques is becoming more and more crucial in the chemical domain. The ML methods aim to obtain predictive models or identify patterns within system variables. They can be divided into three main categories: supervised learning, unsupervised learning, and reinforcement learning. In *supervised learning*, the aim is to infer a function $f: X \rightarrow Y$ using a set of data, each composed of a pair (x, y) where x belongs to the system input and y to the system output (Cunningham et al., 2008; Hastie et al., 2009a). For example, this could be the case of predicting a concentration in a reactive system, knowing its temperature, pressure, and reaction time. It is done using training data obtained from experiments. The dataset must contain the experiment input and its measured output. In *unsupervised learning*, the function f is inferred without the support of the experimental output information. Therefore, the training data for the models of this learning paradigm is not composed of the pair (x,y), but only the value of x is present. The final aim of this learning paradigm is to provide information about the correlation among the variables and whether a smaller set of latent variables can describe them (Greene et al., 2008; Hastie et al., 2009b). *Reinforcement learning* is a machine-learning paradigm where the behaviour of a dynamic system is learned via trial-and-error interaction. In this learning paradigm, the agent is an entity that changes the state of the dynamic system and experiences its new condition. At every timestep, the agent acts on

[1] AI: artificial intelligence

[2] ML: machine learning

the dynamic system. Following the action, the system moves to a new state; this is communicated to the agent with a reward signal. Therefore, the agent performs a further action on the system based on the previous information, and the loop continues. The actions performed by the agent are aimed at reward maximization (Kaelbling et al., 1996). This class of ML is typically used for control tasks, while the supervised and the unsupervised methods are used for modelling tasks. Like all other disciplines, chemical engineering widely utilises ML methods. Several studies and reviews are available in the literature, applying such techniques to model the behaviours of chemical systems, process scheduling, and control. Following a brief list of examples highlighting the variety of possibilities AI offers in chemical engineering. The learning paradigm is underlined for the reported examples to let the reader better comprehend the various AI categories.

Supervised learning

Di Caprio et al. estimated the transfer coefficient within a spray column for CO_2 capture (Di Caprio et al., 2023b). This study employed a hybrid modelling approach, employing physical information about the investigated system and data-driven information. The ML technique had better accuracy in the prediction task than the state-of-the-art technique, achieving errors of around 6%.

Vermeire et al. employed graph neural networks, an ML technique that deals with molecular representation, to estimate the free energy of solvation for many chemical compounds (Vermeire and Green, 2021). This study utilised data obtained by quantum chemistry models and experimental data applying transfer learning techniques. The final model returned a mean absolute error of 0.21kcal/mol.

Carranza-Abaid et al. employed neural network programming to let artificial neural networks simulate the behaviour of NRTL or Wilson equations to predict activity coefficients, creating a data-driven model structure able to respect physical constraints (i.e., to respect the Gibbs-Duhem equation) (Carranza-Abaid et al., 2023). Following the network programming, it was trained using experimental data, achieving an absolute average relative difference of around 2%. Following the

training, the excess properties obtained from the activity coefficients models were also accurately predicted.

Unsupervised learning

Zheng and Zhao employed unsupervised learning to predict chemical process faults (Zheng and Zhao, 2020). The proposed methodology employs data from a continuous plant in which both normal and faulty conditions are included. The algorithm can cluster the two situations, highlighting the different types of faults. This way, faster labelling of the various defective conditions can be executed and used to train a supervised learning technique to distinguish between them.

Zou et al. applied unsupervised learning to design a novel anion-exchange membrane (Zou et al., 2023). A set of compounds with known performances was employed. The molecular structures were provided to the unsupervised learning algorithm that classified them into various groups. Therefore, the scientists were aware of each group performance. Following, the trained model was used to fast-screen many molecular candidates and choose the most performant one for lab experimentation.

Reinforcement Learning

Elmaz et al. employed a reinforcement learning approach to generate an optimal controller for a solvent switch process (Elmaz et al., 2023). The agent aimed to develop a novel control strategy to reduce the process cost. The controller was trained on a digital model of the process and then tested on a pilot plant. It was challenged against the approach used by the company, and it achieved a 25% cost reduction per batch.

Wu et al. employed a reinforcement learning agent to control a continuous reaction system (Wu et al., 2023). The controller aimed to keep the process in optimal condition, countering the effect of raw material quality oscillation. Several agents were tested, and it was observed that with a reduced set of manipulable variables, the agent could develop a better control strategy in less training time than in the case where all the variables were controlled.

Gao et al. employed reinforcement learning approaches to develop new process configurations (Gao et al., 2023). In this work, the agent communicated with an environment composed of chemical simulation software returning a loss function to maximise the economic gain, reducing capital and operational expenses. Their agent was able to construct novel designs for continuous plants, and it was observed how the agent benefited from transfer learning techniques from shortcut methods to execute accurate modelling methods.

Besides the abovementioned examples, AI has been widely applied in chemical engineering for structure-properties prediction (Alshehri et al., 2022; Mann et al., 2022), forward reaction prediction (Johansson et al., 2020; Li et al., 2023; Mann and Venkatasubramanian, 2021a; Venkatasubramanian and Mann, 2022), inverse reaction prediction or retrosynthesis (Ishida et al., 2022; Mann and Venkatasubramanian, 2021b), flowsheet synthesis (Mann et al., 2023b; Vogel et al., 2023), process intensification (Gertig et al., 2020; L´opez-Guajardo et al., 2022), chemical product design (Mann et al., 2023a; Reddy and Ghodke, 2023; Trinh et al., 2021), process digital models (Di Caprio et al., 2023a; Wu et al., 2022) and to estimate transfer coefficients (Di Caprio et al., 2022; Valera et al., 2021). The abovementioned list highlights the possibilities AI offers to support chemical engineering challenges.

Besides academics, also chemical companies are interested in AI techniques because of the possibility of gaining novel information about their production processes and improving the process efficiency (Chiang et al., 2017).

The programming skills of chemical engineers are considered more and more critical, which are helpful for modelling, control, and optimisation tasks. The chemical engineering students are already trained to use Microsoft Excel®, Aspen®, and AIMMS® to deal with their simulation tasks. An AI course with Python programming training can give students more opportunities for their future careers, improving problem-solving skills, logical reasoning, systemic planning, and

creativity. It is believed that this chemical engineering education can respond to the job market demands and transform the market (Teles dos Santos et al., 2018).

Several authors are calling to develop a more structured AI class for the chemical engineering curriculum (Beck et al., 2016; Duever, 2019), because AI courses are often given as general engineering courses and the links to chemical applications or the chemical industry are rare to find. Aside from this, the most common reasons why AI projects fail are 1) the poor chemical engineering field knowledge of data scientists and vice versa, and 2) misalignment of the critical definition between business and technology providers (Demirkan and Dal, 2014; Elprin, 2018; Redman, 2018). Thus, an introductory AI class in a chemical engineering curriculum is becoming crucial for the future technical careers of students and a broader application of such techniques in chemical industries. In this frame, some universities are already moving toward reshaping the chemical engineering curriculum to include AI classes. Many university in North America have started integrating such classes in their chemical engineering curricula (Columbia University, 2022; MIT, 2022; University of Toronto, 2020). In addition, the broader documented course structure about AI teaching to chemical engineering students is from Prof. Venkatasubramanian from Columbia University (Venkatasubramanian, 2022).

Despite the great efforts oversea to integrate AI disciplines into the chemical engineering curriculum, to the best of our knowledge, no efforts toward this direction have been detected in Europe, and the work presented in this paper is the first attempt toward this direction. This paper introduces one of the few structured courses about AI applications in chemical processes for chemical engineering students. The paper will show the main topics illustrated during the course and the utilised teaching procedures. In addition, this work will show the problems that the students should solve and the evaluation of their performance during the course. This work gives guidelines to other educators in academia and industries for shaping a data science class for chemical engineering students.

2. Course content

The course targets the final-year Master's students with a proven background in process control, reaction engineering, modelling and simulation. The students are expected to understand the conceptual knowledge of AI techniques and to apply the learned methodologies in chemical engineering tasks (Krathwohl, 2002). An overview of the course objectives based on "A Revision of Bloom's Taxonomy: An Overview" has been given in the supporting information.

An overview of the course content and structure can be seen in Figure 1. The course was given in eleven classes, offered once a week, and took 3 hours for each class. The classes cover theoretical parts with frontal lectures and practical parts with hands-on exercises applying active learning methods. The main topics are statistical modelling, data processing, optimisation, hybrid modelling, and MPC[3]. The statistical modelling part introduced AI techniques, such as ANN[4], MRF[5], and polynomial regression. The evaluation of students consists of two parts: 20% is based on the individual exercises, and 80% is based on the final group project. For the individual exercises, the general assessments were summative, and the formative assessments were given at the requests of the students. As part of the final evaluation, the teaching team created the group project cases based on the literature and relevant industrial cases, and these were distributed randomly to the groups. For the final group project, both summative and formative assessments were provided. Each group with two members worked on the randomly selected project. The examination projects were about hybrid or ML modelling in chemical cases (e.g., biochemical and separation processes). Each group had a different project and they were asked to apply the

[3] MPC: model predictive control

[4] ANN: artificial neural network

[5] MRF: multivariate rational function

learned techniques in the classes and to bring their own creative ideas to solve the tasks. The projects were provided in the fifth class. The deadline to deliver the projects was one week after the last class of the course, which gave the students six weeks to deliver the task. Sections 2.1 and 2.2, give the course lectures and exercises details.

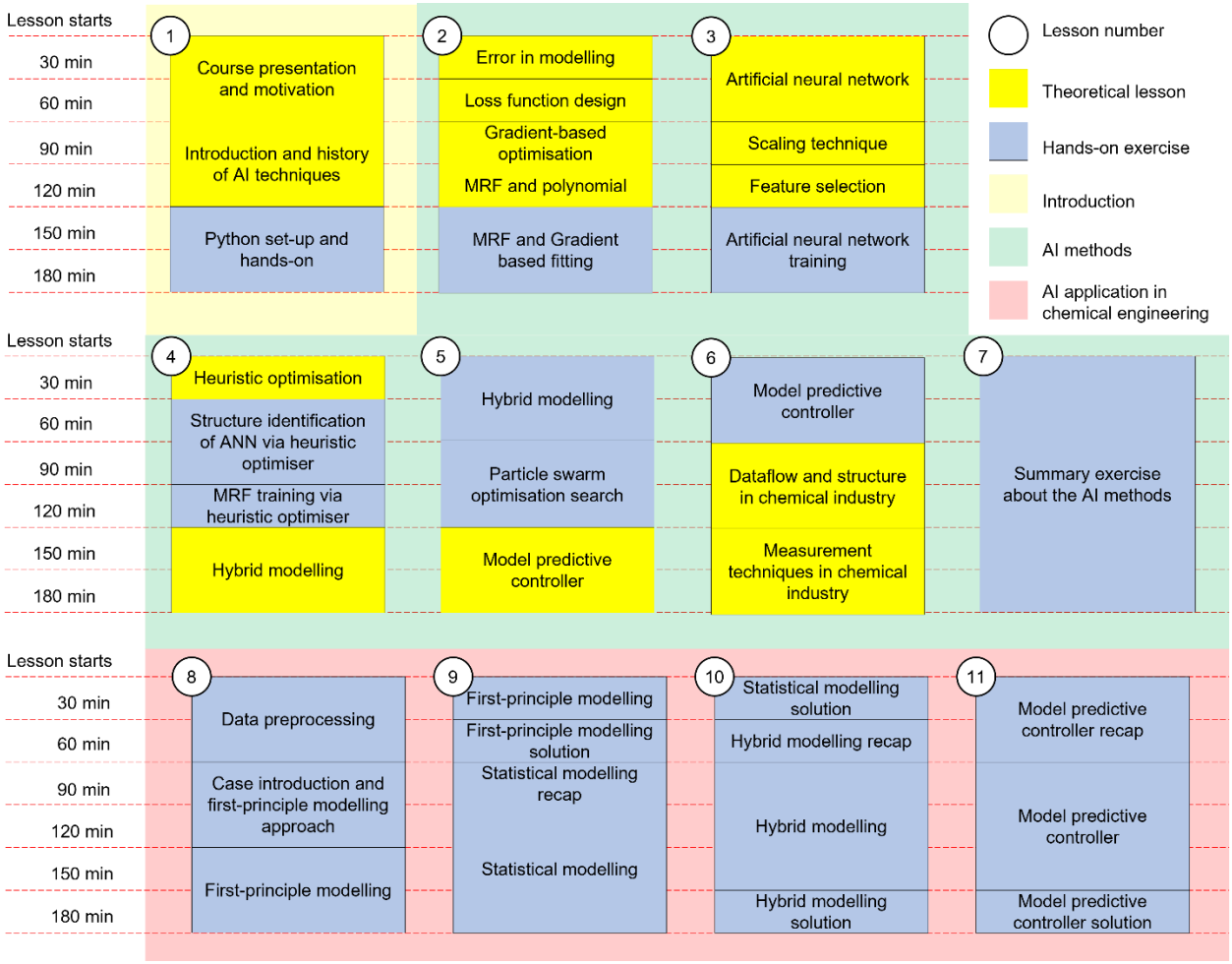


Figure 1: Overview of course content. There are eleven 180-minute classes in this course, consisting of three major parts: introduction, AI methods, and applications in chemical engineering, as shown in light yellow, light green, and light red squares. All classes can be sorted into theoretical lessons (yellow squares) and hands-on exercises (blue squares).

2.1. Theoretical parts (frontal lectures)

The frontal lectures were given in the first six classes of the course, along with the corresponding Python practices. The following explains the details of the lectures.

Lecture 1 - Introduction

The objective of the first class is to give the students a general overview of this course and some backgrounds of AI. It is expected that the students can know what is included in the course, and how they would be examined at the end of the course. Also, they should understand what ML is, why ML is needed in chemical engineering tasks, and the benefits AI can bring to their future career.

Hence, the introduction of the course structure, AI, its applications, and basic Python coding were provided to the students. Recommended materials were given before the start of the course, including related books, papers, and websites. These materials allowed students to know the related topics a week before the course starts. This class was also used to communicate the students the evaluation criteria and format. In this way, the students can better plan their study routes.

The introduction in this class gave the definition, the development history, and the categories of AI. Examples of supervised learning, unsupervised learning, reinforcement learning, and swarm intelligence were shown. Furthermore, AI applications in different industries were introduced, especially in chemical engineering with modelling, control, and optimisation tasks. At the end of this class, the Python coding introduction and installation were given, helping the students install Anaconda and Spyder on their laptops. The most relevant Python libraries were also introduced, while conducting the corresponding practices. The details of the practices can be found in section 2.2.

Lecture 2 - Error, loss functions and gradient-based optimisation

This lecture aims showing the students the main components to perform an ML training (e.g., function structure, optimiser and loss function). In addition, the students were taught how an ML training stage approach works. After the lecture, the students are expected to know the main components of the mathematical framework supporting ML. In addition, they are expected to critically evaluate the applicability of the various methodologies explained in the lecture.

The students were already familiar with the concept of experimental errors. This previous knowledge was utilised to introduce the concepts of errors in modelling. In particular, the difference between the experimental and model error was addressed, together with the usage of the error in the model training. The slides utilised for the course reported some snippets of Python codes regarding implementing the theory just learned. The lecturer commented on the lines of the code one by one, explaining the algorithm without any actual implementation on the computer. When possible, the built-in functions of the recommended libraries were used. In this way, it is expected that the students create a link between the theories explained in the frontal lessons and the practical codes implemented in the hands-on part. In addition, the students downloaded the presented slides from the learning platform of the university, which can be a quick-to-use handbook for future cases.

After introducing the errors and their usage for training tasks, the students learned the concept of parameter optimisation to fit a function. This task is developed at different levels; at first, the students faced very simple examples to solve and gained skills to execute the next more complex task. Namely, the scaffolding technique was applied. This results in significant independency from the students in understanding the task. The students were already familiar with the minimisation of a cost function since they experienced it in other courses in their curriculum (e.g., reaction engineering). However, to introduce the curve fitting, a very simple example was given to the student involving linear functions. In this frame, the concepts and the formulas of mean absolute

error and mean squared error were also introduced. Following, more complex loss functions were introduced, involving regularisation techniques or minimisation of parameters contained in the equation sets.

After the introduction of the loss function, the optimiser was introduced. At the first, a general view on gradient-based and heuristic optimisers was given, highlighting their advantages and disadvantages. Following, the lesson was narrowed down on the gradient-based optimisers. Therefore, the main routines behind gradient descent, stochastic gradient descent, and linear programming were introduced to the students.

At the end of this class, a brief introduction to function approximation was given to the students. However, this topic is explained in detail in Lecture 3.

Lecture 3 – Artificial neural networks, scaling and feature selection

This lecture aims to introduce ANNs and give them more information about model and training design strategies. After this lecture, the students are expected to know the rationale behind the ANN structure and the methodology to train them. Furthermore, they are expected to learn the main data scaling and feature reduction techniques.

At the first, the students were introduced to the main concepts of function approximation. They were taught some of the parametric functions employed in ML, namely polynomial, MRF and ANN (focusing on the multilayer perceptron), telling them the advantages and disadvantages of these functional forms. Following, the students were introduced to the neural network, highlighting the structure of a perceptron and their organisation in layers. Once the student had a clear view of how an ANN works, the basic concepts of the back-propagation algorithm were introduced. The main steps of the algorithm were presented without the mathematical details and proof of the algorithm. Following, the students were introduced to the typical regularisation techniques and the

importance of fine-tuning the ANN structure to avoid overfitting. In this frame, the cross-validation approach and bias-variance trade-off were also shown.

At the end of the ANN section, the main drawbacks of utilising and training an ANN were shown. A particular focus was given to the high data demand for the training. In addition, the difficulties in explaining the correlations between the input and the output of the network were highlighted. The lesson concluded by providing the students with some Python examples.

Following, the students were introduced to data pre-processing and feature selection for continuous variables. The concepts of standardisation and normalisation were explained together with their drawbacks. The slide also contained some codes about how to implement these techniques with Python. Next, the students were introduced to feature selection, highlighting its importance for the reduction of the model parameter and the training efficiency. Namely, the univariate, the recursive feature selection and Pearson correlation matrix for independent feature selection were taught. At the end of this section, the Python commands to implement these functions from available open-source libraries were provided.

Lecture 4 - Heuristic optimisation and hybrid modelling

This lecture aims to teach the students the main heuristic optimisation algorithms and introduce them to hybrid modelling. After this class, the students are expected to learn the main heuristic optimisation and the rationale behind such a group of algorithms. Furthermore, they are expected to understand the main paradigms of hybrid modelling and evaluate the most suitable structures for a given chemical engineering problem. At this point of the course, the students have the skills to identify the parameters of a statistical function and minimise a loss function using a gradient-based optimiser. Heuristic optimisation was introduced to the student as an alternative approach to gradient-based optimisation, dealing with discontinuous functions. First, an overall explanation

about the heuristic optimisers and their rationale was given. Following, the algorithms of PSO[6] and DE[7] were extensively explained.

In the second part of this lesson, the students were taught about hybrid modelling. The topic was introduced with an example where the commercial simulation software significantly deviates from the experimental data. This was used as an example to show that a further modelling step was needed to compensate for the reality gap. Further, an example of reality-gap compensation was given to the students utilising Raoult's law for non-ideal mixtures which they had already encountered during their past classes. Leveraging the previous knowledge of the students about Raoult's law, the concepts of the first-principle and statistical models were introduced. The hybrid model was introduced as a combination of these two techniques. The teaching team also illustrated all the advantages and disadvantages of adopting only first-principle or statistical modelling together with the main advantages of employing a hybrid model. The most frequently used hybrid model structures were illustrated to the students. In addition, the students were taught how to integrate the statistical model suitably into the first-principle model.

Lecture 5 - Model predictive control

The objective of this class is to let students apply the hybrid modelling techniques in MPC for chemical engineering tasks. In their previous studies, the students have learned the classic control for chemical processes. The MPC was compared with the proportional–integral–derivative controller for a better understanding, listing advantages and disadvantages of both. Following, a typical closed-loop MPC algorithm was provided, introducing how MPC works step by step. The

[6] PSO: particle swarm optimisation

[7] DE: differential evolution

types and elements of MPC were explained, too, including linear MPC, nonlinear MPC, models, cost functions, constraints, and optimisers.

Moreover, the MPC applications and current research work were shown to the students. The potential usages of AI applications in chemical process control were also stated. At the end of this class, a sample code of MPC in Python with detailed explanations was shown to the students, which can be a reference for their future tasks.

Lecture 6 – Data flow and measurements in the chemical industry

This lecture aims to teach the students the main problems in data coming from chemical engineering sources and how to work with them. After the class, the students are expected to learn the main measurement techniques used in a production plant and their measurement problems. In addition, they should be able to distinguish between the various signal correction techniques and critically evaluate their applicability for the various cases. In this lecture, the students were taught about the various possible data scenario for chemical engineering problem and challenges hindering and extensive design-of-experiment.

Following, the different graphical data representation techniques were shown to the students and examples in the Matplotlib library (Hunter, 2007) were given. The last part of the lesson highlighted the typical problem of data obtained from an industrial environment (i.e., missing data, noisy data and biased data) and how to deal with them. At the end of this section, some Python codes about signal filtering and denoising profiles were reported and explained.

2.2. Practical parts (hands-on exercises)

Exercise 1 – Python libraries practice: numpy, pandas, matplotlib, and scikit-learn

In the first exercise, the students are expected to get familiar with the widely used Python libraries and to apply these libraries in their future tasks. The libraries include numpy (Harris et al., 2020),

pandas (McKinney, 2010), matplotlib (Hunter, 2007), and scikit-learn (Pedregosa et al., 2011). The example codes and the explanations line by line were given.

In practices of numpy and pandas, the students learned how to import, save and utilise data. The example guidelines made all students successfully repeat the previously given calculations. Students were also asked to make a basic line plot based on the provided data in matplotlib practice, and to fit a linear regression in scikit-learn practice. The details of the practices can be found in the supporting information.

Exercise 2 – Multivariate rational function and gradient-based fitting

The aim of this exercise is to let the students get practice with the function fitting. It requires the use of the concepts learned during the theoretical classes to model a biochemical process.

The task was to train a statistical model and a Michaelis-Menten reaction equation using gradient-based optimisers (Cornish-bowden, 2015). Overall, the students found some difficulties in managing this exercise. The main problems were related to computing a numerical derivative on the data, and creating train and test sets. The students were able to reutilise the codes introduced in the theoretical lesson, and most of the questions were related to the coding not included in the theoretical lesson course materials.

Exercise 3 – Artificial neural networks

The aim of the exercise is to let the students understand the main steps required to train an ANN in Python and let them train their first ANN. In this exercise, the students were asked to model the overall gas-liquid mass-transfer coefficient in a spray column for the CO_2 capture utilising the process input variables as inputs of the model. The students were asked to train various ANNs with different hyperparameters, and to comment on their differences. In addition, the students

were asked to remove one of the input variables (i.e., absorption temperature), and to comment about the impact on the final model accuracy.

The students did not experience any severe bottlenecks during coding. Most of the questions were related to formulating the problem and clarifying the case. Probably, this is related to the fact that all the codes needed for this exercise were already reported in the theoretical lesson slides, and the main task of the students was to modify some of the parameters and see how this impacted the model performance. The students managed the task better than done in Exercise 2 because of two main factors: 1) the availability of the codes in the slides and 2) the experience obtained from the previous Python hands-on classes.

Exercise 4 – Optimisation of neural networks structure and heuristic optimisation

The objective of this exercise was to let the students practice with the non-gradient-based optimiser (i.e., heuristic and Bayesian optimiser). This way, the students are expected to learn the steps to run a heuristic optimisation in python both to solve a parameter identification problem and to identify the hyperparameters while training an ANN. The exercise was composed of two parts: 1) identify the best ANN structure utilising a Bayesian optimisation algorithm, and 2) perform the training on a biochemical system model utilising the DE algorithm. Both the exercises were based on the cases of Exercise 3 and Exercise 2, respectively

Because of compatibility issues with the Keras-tuner library, most students could not execute the first part of the exercise. The ones that performed the exercise smoothly thanked the code provided in the theoretical class. The majority of the questions were about the wideness of the exploration space for the optimiser and how to decide it. In the academic year the paper refers to, the students used Anaconda suite installed on their own computers. The teaching team realised that inappropriate libraries installation was the main reason for the incompatibility issue. The

following academic year, the hands-on sessions were executed on Google Colab suite. In this case, the students did not experience any incompatibility issues.

All the students executed the second part of the exercise and resolved the task without significant problems. Also, in this case, the students raised questions about the wideness of the exploration space of the variables and the correct population number utilised to execute the search.

Exercise 5 – Parameter identification by particle swarm optimisation

The aim of this class is to let students learn how to solve the ODEs[8] in Python and to estimate optimal parameters based on experimental data. This exercise includes the parameter identification by PSO introduced in previous classes. The task aims to find the parameter values of a mathematical model that gives the best fit with existing experimental data. Thus, in a dynamic model-based parameters estimation process, an objective function returning the mean squared error between the model predicted state values and the observed values is minimised, subject to a set of ODEs. The used case is the α-pinene isomerisation (Fuguitt et al., 1947).

Four tasks in total were delivered to students. Firstly, the students imported the provided data in the required formats. The differential equations were then edited by students based on the task description information. The cost function between experimental and modelled values was thirdly defined for further optimisation usage. Questions about PSO settings and boundaries were also asked. Finally, the concentration profiles with optimal parameters were plotted.

[8] ODE: ordinary differential equation

Exercise 6 – Model predictive control

This class aimed to let students get their first experience of how MPC is implemented in Python, by introducing the library Gekko (Beal et al., 2018). An example of a chemical reactor was given, and the explanations line by line was provided as well.

The example is about controlling a continuous stirred tank reactor with one irreversible and exothermic reaction. It is desired to maintain the temperature at a constant setpoint to maximise the conversion of the reactant. The control variable is jacket temperature. The students were asked to use linear MPC and nonlinear MPC. The students found that the performance of nonlinear MPC was better than the linear MPC. Therefore, they can understand the features of Gekko library, which is helpful for future tasks.

Exercise 7 – Summary exercises of AI methods

Class 7 were entirely dedicated to a hands-on exercise. The students were asked via a survey the topics they found challenging to understand among the ones explored in the previous classes. The survey was given to the students at the end of Class 6. The survey results in the heuristic optimisation and the MRFs were the topics in which the students found problems for the exercise part. For this reason, the teaching team created an exercise considering MRF and heuristic optimisation. The students were asked to fit first-order MRF, second-order MRF and polynomial using DE.

The exercise was conducted in a hybrid way using active learning approaches. For each exercise step, the lecturer asked the students what steps to take. After a small discussion, the solutions suggested by students were implemented in the lecturer codes. Then the students were asked to repeat the step in their own way to face the difficulties, ask questions or explore the solutions implemented by the lecturer.

Exercise 8-11 – Model predictive control of a continuous stirred tank reactor with hybrid modelling

This task was given in three subsequent classes and consisted of controlling a CSTR[9] with hybrid modelling. The expected solutions would combine the knowledge learned in the previous exercises. In this way, students can better understand the ML concepts in hybrid modelling and apply the learned techniques to chemical engineering problems. The whole task was divided into four sub-tasks: first-principle modelling, statistical modelling, hybrid modelling, and MPC. In this case, a CSTR with Van der Vusse reaction is modelled and controlled. Cyclopentenyl production from cyclopentadiene by acid-catalysed electrophilic addition of water in a dilute solution is presented (Klatt and Engell, 1998). The students first used different methods to model the process and compared the model performances. Then the optimal model was utilised in the MPC task to understand the importance of the models in the control task.

In the first sub-task, students need to solve the differential equations for first-principle modelling and plot their results compared to the provided data. The second sub-task is to use ANN and MRF to predict reaction rates. The corresponding coefficient of determination and mean squared errors were also calculated and compared. The third sub-task combines the previous first-principle and statistical models to establish the hybrid model. Another approach to develop the hybrid model with a refinement layer was also required to investigate. The final stage utilises the hybrid model in MPC to control the reactor temperatures. The main problems students faced in classes were the saving and loading of trained models. Due to the different versions of libraries, the same codes did not work for some students.

[9] CSTR: continuous stirred tank reactor

3. Results

3.1. General overview

A survey was performed near the end of the course to obtain the feedback of the 14 students to get improvement insights for the following year and for internal evaluation purposes. The questions started from the usefulness of recommended materials listed in the first class (e.g., books, videos, papers, and websites) and course materials (e.g., slides and exercise descriptions).

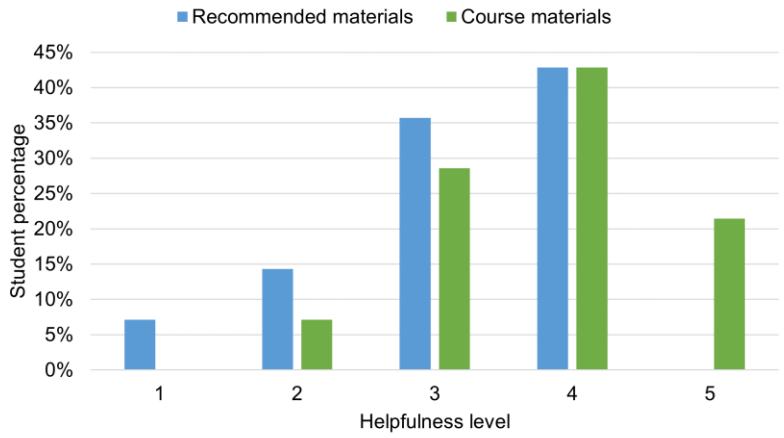


Figure 2: Level for recommended materials(blue), and course materials(green). Level 1 means not helpful at all, and level 5 means very helpful. The helpfulness is increasing from level 1 to level 5.

It can be seen from Figure 2 that more than 50% of the students believed that the recommended materials were not helpful enough, with levels lower or equal to 3. The recommended materials in the first class are the fundamentals of AI and Python programming. The readers can find the material list in the supporting information of this paper. However, most students did not go through the material details due to the limited time. More suitable materials should be recommended for the following semesters, enhancing students interest. Above 50% of the students were satisfied

with the course materials described in section 2. The information or materials should be modified based on the following feedback in sections 3.2 and 3.3.

Besides the students' opinions on the materials, their preferred course arrangements are shown in Figure 3. 69% of the students thought the course should be given once a week. Totally 3 hours should be taken with 1 hour for the theoretical parts (lectures) and the other 2 hours for practical parts (exercises). 23% of the students preferred 2 hours for lectures. Only 8% of the students were willing to take the course twice a week. These decisions could be driven by their busy schedules in the year of their Master's studies and their willingness of learning about more practical solutions rather than theorical aspects of ML.

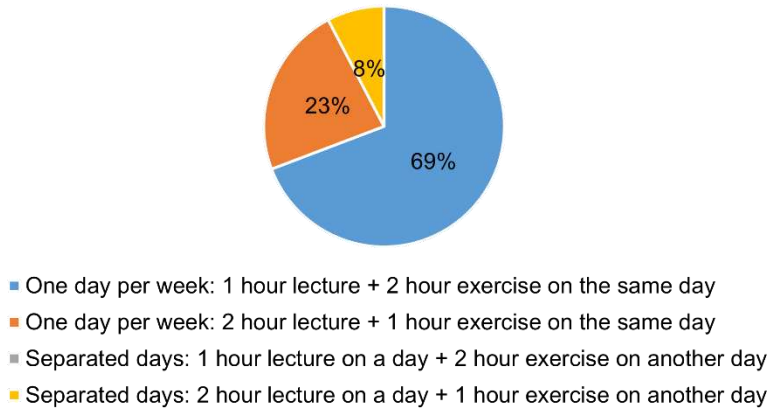


Figure 3: Preferred lectures from the students and exercise structure of the course. Most students prefer to take the course once a week instead of twice a week.

3.2. Theoretical parts (frontal lectures)

Figure 4 reports the survey results for the theoretical lectures. Figure 4a shows that 50% of the students found statistical modelling the most interesting topic explained during the theoretical lectures. It can be hypothesised that this is because of the extensive time spent on this topic. According to this survey, the second most interesting topic is MPC. This data science class was

framed in the more extensive course "New methods for process control". Therefore, it expected a high interest from the students in the topics regarding the control of chemical processes. In addition, the students already had previous experiences in process control. They were probably more able to frame MPC in the broader topic of process control. However, some students gave the MPC topic a meagre degree. Therefore, two groups of students can be identified in this course. The first group gave high scores to the MPC and showed high interest. The second group allocated this subject to the lower grades of the scale. This is related to the complex math underlying this topic. The optimisers and the data pre-processing obtained mid-scored in the overall scales.

In Figure 4b, one can see the answers from the students to the statement, "I found it difficult to implement the theoretical topics in Python". Around 42% of the students agreed with this sentence with a score of 4 out of 5. It means that most of the students found difficulties translating the theoretical lectures into a practical solution implemented in Python. It is hypothesised that this is related to the low expertise of the students in the usage of Python. The students had never practised with a programming language before this course and found it challenging to develop a stepwise code. This hypothesis can be supported by the fact that most students agreed with a score of 2 or 3 out of 5 to the statement "I found the topics in the theoretical lectures beneficial". More than 70% of the students voted for these two options. This means that the students comprehend the utility of the topics covered during the theoretical lectures but found a bottleneck in implementing them in Python syntax.

While the survey results indeed indicate two challenges: 1) the perceptions of students of the recommended materials not being sufficient; 2) the time constraints they face when going through the material, it is essential to consider the broader context. The course materials are designed to provide a comprehensive coverage of the subject matter, and trimming down content may risk compromising the breadth of knowledge imparted to students. The experience of students can vary from year to year due to individual backgrounds, prior knowledge, and learning preferences.

A broader curriculum affords students the flexibility to delve into various innovative techniques and approaches.

Another possible solution is to split the course into two parts, with one focusing on fundamentals and the other on advanced topics including recent AI trends. However, assessing whether students are adequately prepared for the advanced topics in Part 2 after completing Part 1 and ensuring that Part 1 and Part 2 align coherently would be essential. The availability of resources and instructors to support both parts of the course effectively must be considered.

In summary, while the survey results provide valuable insights into the challenges faced by our students, it is important to approach any potential adjustments to the course content with a careful and comprehensive perspective, providing the best possible learning experience to all students.

a)

	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6
Classic optimisation	7.14%	14.29%	7.14%	14.29%	**28.57%**	**28.57%**
Data processing	14.29%	7.14%	**28.57%**	21.43%	7.14%	21.43%
Heuristic optimisation	7.14%	7.14%	7.14%	**35.71%**	21.43%	21.43%
Hybrid modelling	21.43%	21.43%	21.43%	14.29%	7.14%	14.29%
Model predictive control	0.00%	**35.71%**	21.43%	0.00%	**28.57%**	14.29%
Statistical modelling approaches: ANN, MRF, Polynomial	**50.00%**	14.29%	14.29%	14.29%	7.14%	0.00%

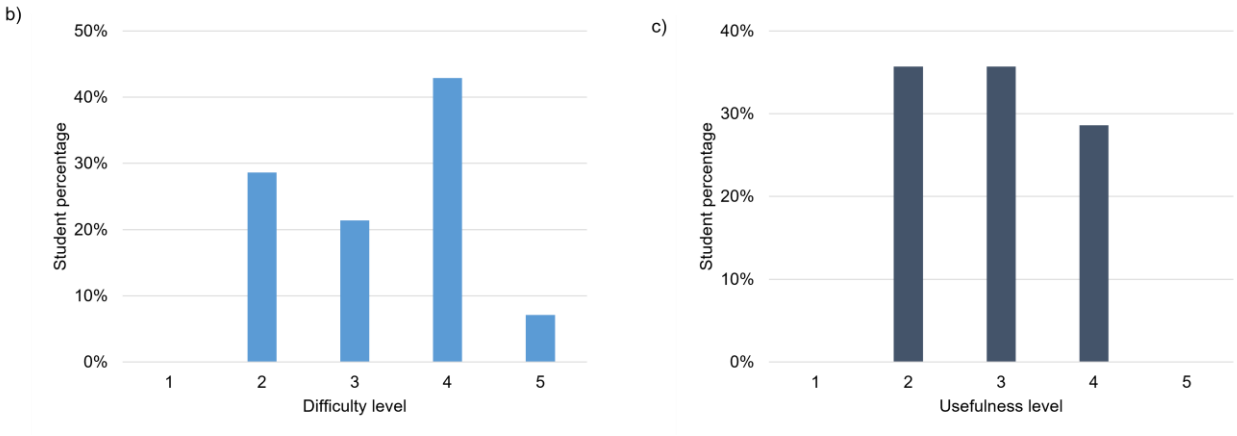


Figure 4: Results of the survey answered by the students at the end of the course about the theoretical lectures. Level 6 to 1 is from the most interesting/ difficult /useful topics to the least interesting/ difficult /useful ones. a): Interest heat map of different topics in theoretical lectures; b): Difficulty level of student feedback "I found it difficult to implement the theoretical topics in Python"; c): Usefulness level of student feedback "I found the topics in the theoretical lectures beneficial".

3.3. Practical parts (hands-on exercises)

The exercises are connected to the theoretical parts, such as classic optimisation, data processing, heuristic optimisation, hybrid modelling, statistical modelling, and MPC. In the classes, the sample codes in Python were provided. Detailed information on exercise tasks is provided in the supporting information of this paper.

Figure 5 reports the results of the survey for the hands-on classes. Figure 5a gives the student interest on the different topics covered in the hands-on classes. One can see that about 50% of students agree that statistical modelling is the most interesting topic, including ANN, MRF, and polynomial regression. Heuristic and classic optimisation are the topics with low interesting levels. These algorithms are too far from the chemical engineering area, and the students found them more challenging to understand than the other techniques. This could be the reason why students show little interest. The average interest levels for different topics are also shown in Figure 5b, indicating similar results. It can give future educators the insights that the high amount of statistical or hybrid modelling content can encourage student interest.

a)

	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6
Classic optimization	21.43%	**42.86%**	14.29%	7.14%	7.14%	7.14%
Data processing	21.43%	7.14%	**28.57%**	21.43%	0.00%	21.43%
Heuristic optimization	**28.57%**	7.14%	**28.57%**	21.43%	14.29%	0.00%
Hybrid modelling	7.14%	14.29%	14.29%	**21.43%**	**21.43%**	**21.43%**
Model predictive control	14.29%	**28.57%**	7.14%	21.43%	**28.57%**	0.00%
Statistical modelling approaches: ANN, MRF, Polynomial	7.14%	0.00%	7.14%	7.14%	28.57%	**50.00%**

b)

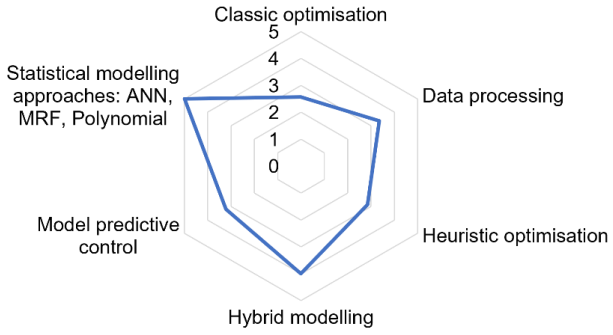


Figure 5: The interest levels in different topics of practical classes. Level 6 to 1 is from the most interesting topics to the least interesting ones. a): Interest heat map of student percentages in different topics. The values are calculated by the number of selections dividing the total number

of students. Each student can select one topic for each level. b): Average interesting levels for different topics.

Figure 6 reports the challenges the students encountered during the hands-on exercises. An equal amount of students think "installation and setting up the environments" is the most difficult and the least difficult. During the course, some students encountered problems while installing Python on their computers. Moreover, the recommended materials guiding them installing the required software on their computer were not fully explored by the students. This resulted in the issues while running the exercises. In Figure 6, the motivation to use Python in chemical engineering is the least difficult aspect for the students. None of the students worked with Python before this course. Once they faced a complex problem, they would use Excel instead of more advanced programming tools. Python gives more customization possibilities than commercial software such as Excel. In their future work, they will have the ability to justify if it is suitable to apply Python to solve their problems in chemical engineering. The second most challenging aspect encountered from the students while executing the exercise is "get the documentation and support", this could be related to the various formats of libraries and functions available in Python, which are suitable for advanced users but might be non-friendly to beginners. Thanks to the availability of ChatGPT (OpenAI, 2022), students can now access comprehensive explanations of various libraries up to September 2021.

a)

	Level 1	Level 2	Level 3	Level 4
Lack of the chemical engineering problems to use python	7.69%	38.46%	**46.15%**	7.69%
Installation and setting up the environment	**38.46%**	7.69%	15.38%	38.46%
Get the documentation and support	15.38%	**38.46%**	15.38%	30.77%
Lack of the motivation to use python in chemical engineering problems	**38.46%**	15.38%	23.08%	23.08%

b)

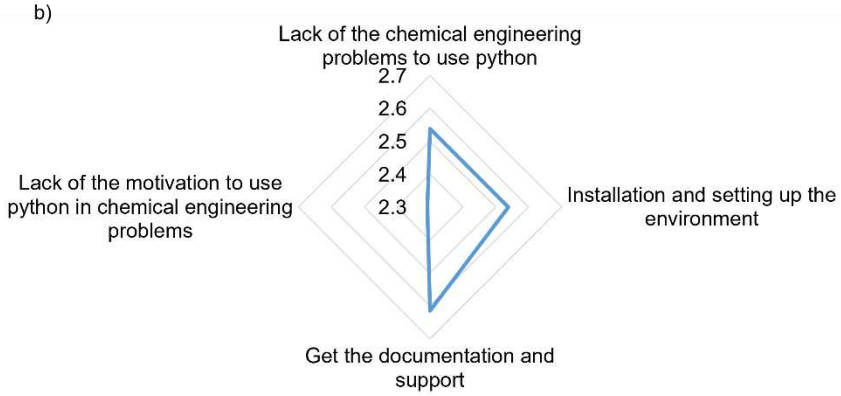


Figure 6: The difficulty levels for different aspects in the course. Level 4 to 1 is from the most difficult aspect to the least difficult one based on the students' feedback. a): The student percentages are calculated by the number of selections dividing the total number of students. Each student can select one aspect for each level. b): Average difficulty levels for different aspects.

The practical part of the course was conducted in two work streams. 40% of the time was spent on hands-on coaching, explaining the codes line by line. 60% of the time was given to the students for solving individual tasks and asking questions. The survey asked if the students would bring a modification for this time-split. The answers are shown in Figure 7. 29% of students desired more hands-on coaching time, increasing to 50%. The other 50% of the time is for individual questions and tasks. According to this feedback, the time distribution for this course in the following semesters will be adjusted.

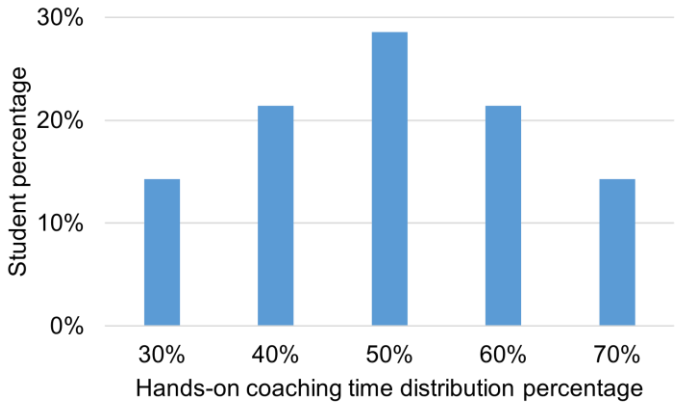


Figure 7: Expected time distribution for practical parts from students.

The last question of the survey asked about the confidence in using the learned techniques for future tasks. The results are reported in Figure 8. Approximately 15% of students have high confidence in such topic. More than half of students show confidence above or equal to level 3, which gives promising feedback for future education. The students realise the importance of applying AI and related techniques to chemical engineering tasks.

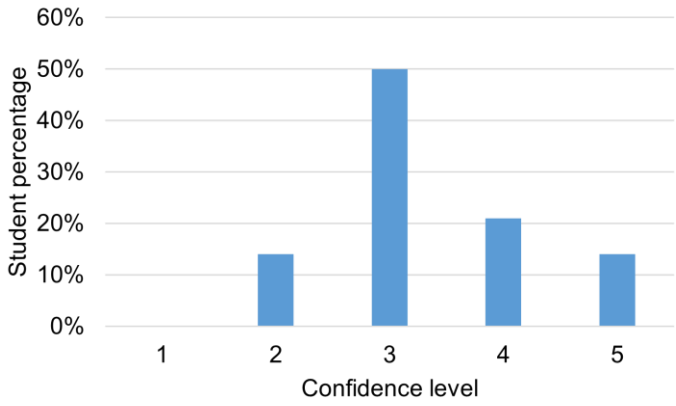


Figure 8: Confidence level of students using the learned techniques in this course to solve future tasks. Level 5 indicates the highest confidence, and level 1 is the lowest.

4. Conclusions

In this work, we introduced an artificial intelligence course for chemical engineering Master's students. This is one of the first documented attempts in European universities. The topics covered by this course were mainly about applications of ML techniques to model and optimise chemical processes. Most of the classes were divided into frontal lectures and hands-on exercises. The evaluation of students performance by the teaching team was conducted from final group projects with two members as well as the weekly individual exercises. The course evaluation from the students was performed from an anonymous online survey. The students showed the highest interest in statistical modelling for both lectures and exercises, followed by model predictive control. Because of their poor previous knowledge of the topics covered in the course, the students found it challenging to implement the concepts explained in the frontal lectures into practical solutions in Python for the hands-on exercises. One of the strong recommendations is to modify the Bachelor's curriculum so that the students improve their programming skills. In this way, the students would get into this course or similar courses with high confidence in their Master's studies. In the alternative, the structure of similar courses should focus more on hands-on coaching to enhance the student programming capacities. The students were curious about AI-related topics, and some gave excellent performance in the final exam. With this work, we encourage other European universities to expand their teaching curriculum with AI-related courses.

Funding: This work was supported by VLAIO, DAP2CHEM: Real-time data-assisted process development and production in chemical applications (HBC.2020.2455).

References

Alshehri, A.S., Tula, A.K., You, F., Gani, R., 2022. Next Generation Pure Component Property Estimation Models: With and Without Machine Learning Techniques. AIChE J. 6. https://doi.org/https://doi.org/10.1002/aic.17469.

Beal, L.D.R., Hill, D.C., Abraham Martin, R., Hedengren, J.D., 2018. GEKKO optimization suite. Processes 6. https://doi.org/10.3390/pr6080106

Beck, D.A.C., Carothers, J.M., Subramanian, V.R., Pfaendtner, J., 2016. Data science: Accelerating innovation and discovery in chemical engineering. AIChE J. 62, 1402–1416. https://doi.org/https://doi.org/10.1002/aic.15192

Carranza-Abaid, A., Svendsen, H.F., Jakobsen, J.P., 2023. Thermodynamically consistent vapor-liquid equilibrium modelling with artificial neural networks. Fluid Phase Equilib. 564, 113597. https://doi.org/10.1016/j.fluid.2022.113597

Chiang, L., Lu, B., Castillo, I., 2017. Big data analytics in chemical engineering. Annu. Rev. Chem. Biomol. Eng. 8, 63–85. https://doi.org/10.1146/annurev-chembioeng-060816-101555

Columbia University, 2022. MS in Chemical Engineering with a Concentration in Data and Computational Science [WWW Document]. URL https://www.cheme.columbia.edu/ms-chemical-engineering-concentration-data-and-computational-science (accessed 7.19.22).

Cornish-bowden, A., 2015. One hundred years of Michaelis – Menten. Perspect. Sci. 4, 3–9. https://doi.org/10.1016/j.pisc.2014.12.002

Cunningham, P., Cord, M., Delany, S.J., 2008. Supervised learning, in: Machine Learning Techniques for Multimedia. pp. 21–49. https://doi.org/10.1007/978-3-540-75171-7_2

Demirkan, H., Dal, B., 2014. Why do so many analytics projects fail? [WWW Document]. Analytics. URL https://pubsonline.informs.org/do/10.1287/LYTX.2014.04.02/full/ (accessed 7.19.22).

Di Caprio, U., Kayahan, E., Wu, M., Mercelis, S., Hellinckx, P., Van Gerven, T., Waldherr, S.,

Leblebici, M.E., 2022. Optimization of an artificial neural network structure for modelling carbon capture in spray columns, in: Montastruc, L., Negny, S. (Eds.), 32nd European Symposium on Computer Aided Process Engineering, Computer Aided Chemical Engineering. Elsevier, pp. 1411–1416. https://doi.org/https://doi.org/10.1016/B978-0-323-95879-0.50236-8

Di Caprio, U., Wu, M., Elmaz, F., Wouters, Y., Vandervoort, N., Anwar, A., Mercelis, S., Waldherr, S., Hellinckx, P., Leblebici, M.E., 2023a. Hybrid modelling of a batch separation process. Comput. Chem. Eng. 177, 108319. https://doi.org/10.1016/j.compchemeng.2023.108319

Di Caprio, U., Wu, M., Vermeire, F., Van Gerven, T., Hellinckx, P., Waldherr, S., Kayahan, E., Leblebici, M.E., 2023b. Predicting overall mass transfer coefficients of CO2 capture into monoethanolamine in spray columns with hybrid machine learning. J. CO2 Util. 70, 102452. https://doi.org/10.1016/j.jcou.2023.102452

Duever, T.A., 2019. Data science in the chemical engineering curriculum. Processes 7. https://doi.org/10.3390/pr7110830

Elmaz, F., Di Caprio, U., Wu, M., Wouters, Y., Van Der Vorst, G., Vandervoort, N., Anwar, A., Leblebici, M.E., Hellinckx, P., Mercelis, S., 2023. Reinforcement learning-based approach for optimizing solvent-switch processes. Comput. Chem. Eng. 176, 108310. https://doi.org/10.1016/j.compchemeng.2023.108310

Elprin, N., 2018. Data Science: 4 Reasons Why Most Are Failing to Deliver [WWW Document]. KDnuggets. URL https://www.kdnuggets.com/2018/05/data-science-4-reasons-failing-deliver.html (accessed 7.19.22).

Fuguitt, R.E., Erskine Hawkins, J., Fuguitt, R.E., 1947. Rate of the Thermal Isomerization of α-Pinene in the Liquid Phase. J. Am. Chem. Soc. 69, 319–322. https://doi.org/10.1021/ja01194a047

Gao, Q., Yang, H., Shanbhag, S.M., Schweidtmann, A.M., 2023. Transfer learning for process

design with reinforcement learning, Computer Aided Process Engineering. Elsevier Masson SAS. https://doi.org/10.1016/B978-0-443-15274-0.50319-X

Gertig, C., Leonhard, K., Bardow, A., 2020. Computer-aided molecular and processes design based on quantum chemistry : current status and future prospects. Curr. Opin. Chem. Eng. 27, 89–97. https://doi.org/10.1016/j.coche.2019.11.007

Greene, D., Cunningham, P., Mayer, R., 2008. Unsupervised learning and clustering, in: Machine Learning Techniques for Multimedia. pp. 51–90.

Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E., 2020. Array programming with NumPy. Nature 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hastie, T., Tibshirani, R., Friedman, J., 2009a. Overview of Supervised Learning, in: The Elements of Statistical Learning. pp. 83–85. https://doi.org/10.1007/b94608_2

Hastie, T., Tibshirani, R., Friedman, J., 2009b. Unsupervised Learning, in: The Elements of Statistical Learning. pp. 83–85. https://doi.org/10.1007/b94608_14

Hunter, J.D., 2007. Matplotlib: a 2D Graphics Environment. Comput. Sci. Eng. 9, 90–95.

Ishida, S., Terayama, K., Kojima, R., Takasu, K., Okuno, Y., 2022. AI-Driven Synthetic Route Design Incorporated with Retrosynthesis Knowledge. J. Chem. Inf. Model. 62, 1357–1367. https://doi.org/10.1021/acs.jcim.1c01074

Johansson, S., Thakkar, A., Kogej, T., Bjerrum, E., Genheden, S., Bastys, T., Kannas, C., Schliep, A., Chen, H., Engkvist, O., 2020. AI-assisted synthesis prediction. Drug Discov. Today Technol. 32–33, 65–72. https://doi.org/10.1016/j.ddtec.2020.06.002

Kaelbling, L.P., Littman, M.L., Moore, A.W., 1996. Reinforcement Learning: A Survey. J. Artif.

Intell. Res. 4, 237–285.

Klatt, K.U., Engell, S., 1998. Gain-scheduling trajectory control of a continuous stirred tank reactor. Comput. Chem. Eng. 22, 491–502. https://doi.org/10.1016/S0098-1354(97)00261-5

Krathwohl, D.R., 2002. A Revision of Bloom ' s Taxonomy: An Overview. Theory Pract. 41. https://doi.org/10.1207/s15430421tip4104_2

L´opez-Guajardo, E.A., Delgado-Licona, F., ´Alvarez, A.J., Nigam, K.D.P., Montesinos-Castellanos, A., Morales-Menendez, R., 2022. Process intensification 4 . 0 : A new approach for attaining new , sustainable and circular processes enabled by machine learning. Chem. Eng. Process. - Process Intensif. 180. https://doi.org/10.1016/j.cep.2021.108671

Li, B., Su, S., Zhu, C., Lin, J., Hu, X., Su, L., Yu, Z., Liao, K., Chen, H., 2023. A deep learning framework for accurate reaction prediction and its application on high-throughput experimentation data. J. Cheminform. 15, 1–12. https://doi.org/10.1186/s13321-023-00732-w

Mann, V., Brito, K., Gani, R., Venkatasubramanian, V., 2022. Hybrid, Interpretable Machine Learning for Thermodynamic Property Estimation using Grammar2vec for Molecular Representation. Fluid Phase Equilib. 561, 113531. https://doi.org/10.1016/j.fluid.2022.113531

Mann, V., Gani, R., Venkatasubramanian, V., 2023a. Fluid Phase Equilibria Group contribution-based property modeling for chemical product design : A perspective in the AI era. Fluid Phase Equilib. 568, 113734. https://doi.org/10.1016/j.fluid.2023.113734

Mann, V., Gani, R., Venkatasubramanian, V., Company, S., 2023b. Intelligent Process Flowsheet Synthesis and Design using Extended SFILES Representation, Computer Aided Process Engineering. Elsevier Masson SAS. https://doi.org/10.1016/B978-0-443-15274-0.50036-6

Mann, V., Venkatasubramanian, V., 2021a. Predicting Chemical Reaction Outcomes: A Grammar Ontology-based Transformer Framework. AIChE J. 3.

https://doi.org/https://doi.org/10.1002/aic.17190.

Mann, V., Venkatasubramanian, V., 2021b. Retrosynthesis prediction using grammar-based neural machine translation : An information-theoretic approach. Comput. Chem. Eng. 155, 107533. https://doi.org/10.1016/j.compchemeng.2021.107533

McKinney, W., 2010. Data Structures for Statistical Computing in Python. Proc. 9th Python Sci. Conf. 1, 56–61. https://doi.org/10.25080/majora-92bf1922-00a

MIT, 2022. MIT Subject Listing & Schedule Fall 2022 Search Results [WWW Document]. URL http://student.mit.edu/catalog/search.cgi?search=Machine+Learning+for+Molecular+Engineering&style=verbatim (accessed 8.18.22).

OpenAI, 2022. Introducing ChatGPT [WWW Document]. URL https://openai.com/blog/chatgpt (accessed 9.18.23).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 2825–2830.

Reddy, P.S., Ghodke, P.K., 2023. Image Analysis Using Artificial Intelligence in Chemical Engineering Processes: Current Trends and Future Directions, in: Image Processing and Intelligent Computing Systems. p. 22. https://doi.org/https://doi.org/10.1201/9781003267782

Redman, T.C., 2018. Are You Setting Your Data Scientists Up to Fail? [WWW Document]. Harv. Bus. Rev. URL https://hbr.org/2018/01/are-you-setting-your-data-scientists-up-to-fail (accessed 7.19.22).

Teles dos Santos, M., Vianna, A.S., Le Roux, G.A.C., 2018. Programming skills in the industry 4.0: are chemical engineering students able to face new problems? Educ. Chem. Eng. 22, 69–76. https://doi.org/10.1016/j.ece.2018.01.002

Trinh, C., Meimaroglou, D., Hoppe, S., 2021. Machine learning in chemical product engineering:

The state of the art and a guide for newcomers. Processes 9. https://doi.org/10.3390/pr9081456

Udugama, I.A., Bayer, C., Baroutian, S., Gernaey, K. V, Yu, W., Young, B.R., 2022. Digitalisation in chemical engineering : Industrial needs , academic best practice , and curriculum limitations. Educ. Chem. Eng. 39, 94–107. https://doi.org/10.1016/j.ece.2022.03.003

University of Toronto, 2020. Emphasis in Analytics [WWW Document]. URL https://gradstudies.engineering.utoronto.ca/professional-degrees/emphasis-in-analytics/ (accessed 7.19.22).

Valera, V.Y., Codolo, M.C., Martins, T.D., 2021. Artificial neural network for prediction of SO2 removal and volumetric mass transfer coefficient in spray tower. Chem. Eng. Res. Des. 170, 1–12. https://doi.org/10.1016/j.cherd.2021.03.008

Venkatasubramanian, V., 2022. Teaching artificial intelligence to chemical engineers: experience from a 35-year-old course. Chem. Eng. Educ. 56.

Venkatasubramanian, V., Mann, V., 2022. Artificial intelligence in reaction prediction and chemical synthesis. Curr. Opin. Chem. Eng. 36, 100749. https://doi.org/10.1016/j.coche.2021.100749

Vermeire, F.H., Green, W.H., 2021. Transfer learning for solvation free energies: From quantum chemistry to experiments. Chem. Eng. J. 418, 129307. https://doi.org/10.1016/j.cej.2021.129307

Vogel, G., Balhorn, L.S., Schweidtmann, A.M., 2023. Learning from flowsheets : A generative transformer model for autocompletion of flowsheets. Comput. Chem. Eng. 171, 108162. https://doi.org/10.1016/j.compchemeng.2023.108162

Wu, M., Di Caprio, U., Elmaz, F., Metten, B., De Clercq, D., Van Der Ha, O., Mercelis, S., Hellinckx, P., Braeken, L., Leblebici, M.E., 2022. A comparative study of swarm intelligence and artificial neural networks applications in modeling complex reaction processes, Computer Aided Chemical Engineering.

Wu, M., Elmaz, F., Di Caprio, U., De Clercq, D., Mercelis, S., Hellinckx, P., Braeken, L., Vermeire, F., Leblebici, M.E., 2023. Real-time optimization of a chemical plant with continuous flow reactors via reinforcement learning, Computer Aided Chemical Engineering. Elsevier Masson SAS. https://doi.org/10.1016/B978-0-443-15274-0.50073-1

Zheng, S., Zhao, J., 2020. A new unsupervised data mining method based on the stacked autoencoder for chemical process fault diagnosis. Comput. Chem. Eng. 135, 106755. https://doi.org/10.1016/j.compchemeng.2020.106755

Zou, X., Xu, G., Fang, P., Li, W., Jin, Z., Guo, S., Hu, Y., Li, M., Sun, Z., Yan, F., 2023. Unsupervised Learning-Guided Accelerated Discovery of Alkaline Anion Exchange Membranes for Fuel Cells. Angew. Chemie Int. Ed. 19. https://doi.org/https://doi.org/10.1002/anie.202300388

Supporting Information of An Artificial Intelligence Course for Chemical Engineers

Min Wu[1,*], Ulderico Di Caprio[1,*], Florence Vermeire[2], Leen Braeken[1], Peter Hellinckx[3], Steffen Waldherr[2,4], M. Enis Leblebici[1,**]

[1] Center for Industrial Process Technology (CIPT), Department of Chemical Engineering, KU Leuven, Agoralaan Building B, 3590 Diepenbeek, Belgium

[2] Chemical Reactor Engineering and Safety (CREaS), Leuven (Arenberg), Celestijnenlaan 200f - box 2424, 3001 Leuven, Belgium

[3] IDLab, Faculty of Applied Engineering, University of Antwerp-imec, Sint-Pietersvliet 7, 2000 Antwerp, Belgium

[4] Molecular Systems Biology (MOSYS), Department of Functional and Evolutionary Ecology, Faculty of Life Sciences, University of Vienna, 1030 Vienna, Austria

*Equal contributions

**Corresponding author (muminenis.leblebici@kuleuven.be)

1. Recommended materials

Books

- "Hands On Machine Learning with Scikit Learn, Keras , and TensorFlow" by Aurelien Geron
- "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython " by Wes McKinney
- "Hybrid Modeling in Process Industries" by Jarka Glassey
- "Artificial intelligence in chemical engineering" by Thomas E. Quantrille

Papers

- Venkatasubramanian , V. The Promise of Artificial Intelligence in Chemical Engineering: Is It Here, Finally? (2018) doi:10.1002/aic.16489.
- von Stosch, M., Oliveira, R., Peres, J. & Feyo de Azevedo, S. Hybrid semi parametric modeling in process systems engineering: Past, present and future. Comput . Chem. Eng. 60, 86 101 (2014).
- Chiang, L., Lu, B. & Castillo, I. Big Data Analytics in Chemical Engineering. Annu . Rev. Chem. Biomol . Eng. 8, 63 85 (2017).

Links

- Python fundamentals: https://pythonprogramming.net/
- Anaconda: https://www.anaconda.com/
- Spyder: https://www.spyder-ide.org/
- Numpy: https://numpy.org/

- Pandas: https://pandas.pydata.org/, https://pythonprogramming.net/introduction-python3-pandas-data-analysis/
- Sklearn: https://scikit-learn.org/stable/
- Keras: https://pythonprogramming.net/introduction-deep-learning-python-tensorflow-keras/

2. Exercises

Exercise 1 - Python libraries practice: numpy, pandas, matplotlib

1. Numpy exercises

a. Convert a given 1-D array with 27 elements from 0 to 26, to a 3-D array

b. Stack 2 numpy arrays horizontally i.e., 2 arrays having the same 1st dimension (number of rows in 2D arrays)

a1 = np.array([[1,2], [3,4], [5,6]])

a2 = np.array([[7,8], [9,10], [10,11]])

c. Stack 2 numpy arrays above vertically i.e., 2 arrays having the same last dimension (number of columns in 2D arrays)

d. From 2 numpy arrays [1,2,3,4,5] and [1,3,2,4,5], extract the indexes in which the elements in the 2 arrays match

e. Output a 5-by-5 array of random integers between 0 and 10

2. Pandas exercises

a. For the given a list [2, 4, 5, 6, 9], output the corresponding pandas series

b. For the given a list [2, 4, 5, 6, 9], output the corresponding pandas series with odd indexes only

c. Generate the series of dates from 1st May, 2021 to 12th May, 2021 (both inclusive)

d. Apply the function, f(x) = x/2 on every element of a given pandas series

e. For the given a dictionary

dictionary = {'name': ['Vinay', 'Kushal', 'Aman'],

'age' : [22, 25, 24],

'occ' : ['engineer', 'doctor', 'accountant']}

convert it into corresponding dataframe and display it

3. Matplotlib exercises

a. Make a python function $x3 + 2x2 - \sqrt{5x} + 6$.

b. Make the average and the standard deviation of the X and Y:

   X =[0,1,2,3,…,100]

   Y is the output of the function above

c. Plot the function in the range [0,100] in x axis

   X label

   Y label

   Blue dash line

d. In the same plot, add another line y = x in red, show the two legends

Exercise 2 - Multivariate rational function

Bioprocess are driving important innovations in chemical and pharmaceutical industries. They are employed in several processes such as production of alcohol and antibiotics. Usually the reaction network of bioprocesses is very wide, and their behavior is hard to model by using first-principle models.

Your start-up has found a new method to produce a commercial molecule by using more resistant microorganism, it means a lower production price. However, the prediction of their behavior is not well defined, and a reaction model is required to design a reactor for the commercial plant.

Some lab experiments have been run in batch mode in which the concentration of the reactant is measured at different concentration of micro-organism. A dataset containing these information is available and the columns are

•Time [h]: the time elapsed from the start of the experiment

•SubProd [g/L]: the concentration of the reactant at the referred time-step

•ConcEnz[g/L]: the concentration of the microorganism used for that experiment

Model the results to get the reaction rate by using a multivariate rational function using differential evolution algorithm as optimizer. Answer to the following questions:

a. Model them by using only 1st order MRF. Comment the result

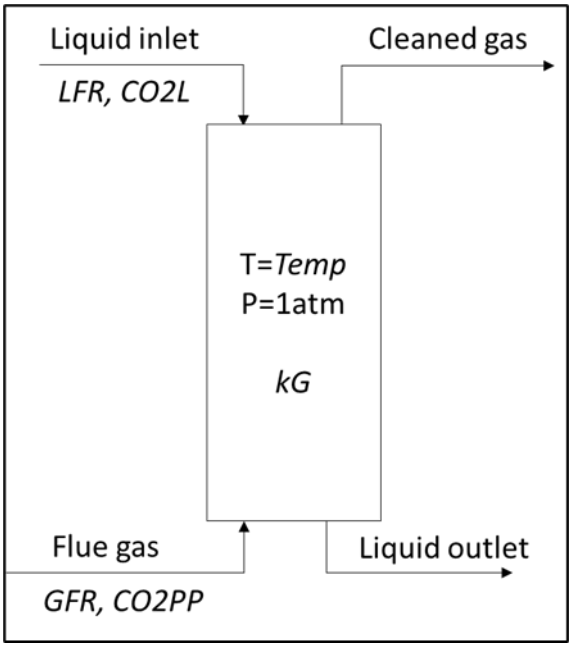
b. Model them by using 2nd order MRF. Comment the result

c. Model them by using a Michaelis Menten equation. Comment the result and compare it to the results obtained with a black box model.

Exercise 3 – Artifical neural networks

The energy production is one of the main contributors to the global warming due to the high amount of CO2 released during the combustion of fossil fuels. To reduce the amount of CO2 released in the atmosphere the energy plants are always equipped with a CO2 capture section in which the flue gas of the power plant is treated before the releasing in atmosphere.

An energy company would like to increase the efficiency CO2 capture column. During the years they have collected a database that correlates the process variables with the gas-side mass-transfer coefficient in the stripping column of the process. The stripping process works at atmospheric pressure.



The process is reported in Fig1 and the variable space is represented by:

- The liquid flowrate of the stripping stream (LFR in the dataset) in kmol/h
- The gas flowrate of the stripped stream (GFR in the dataset) in kmol/h
- The stripping temperature (Temp in the dataset) in Celsius degrees
- The CO_2 loading inside the stripping stream at the column inlet (CO2L in the dataset) in CO2mol/kmol
- The CO_2 loading of the flue gas at the column inlet (CO2PP in the dataset) in %vol
- The mass-transfer coefficient gas-side related to the condition showed in the row (kG in the dataset) in kmol*m-2*h-1*kPa-1

Hypothesizing that the flowrates does vary between in inlet and the outlet of the column, model the value of kG by using an Artificial Neural Network as function of all the other variables. In addition answer to the following questions:

a. Train an ANN that has no hidden layers. Scale the data by linear scaling the data in the range [0,1]. (3 points)

b. Train an ANN that has one hidden layer with 12 nodes inside. What is the difference with the network trained before? Why there are these differences? (3 points)

c. What is the impact of the loss function? Use the mean squared error and the mean absolute error to assess its impact on the final results. (3 points)

d. Assess the impact of the activation function? What is the better one for each layer? Why? (2 points)

e. What is the best number of the layer and nodes? Why is it correct? (2 points)

f. Drop the columns "Temp" in the dataset. What is the final result? Is it possible to train an ANN with these conditions? Why? (3 points)

Exercise 4 – Optimization of neural networks structure and differential evolution

1. ANN structure optimization

Find the best ANN structure for the exercise executed in Class#3. Do it using keras-tuner (tutorial1, tutorial2):

a. Utilize Bayesian optimization to perform the search of the optimal structure (3 points)

b. Utilize the random search to perform the structure optimization (3 points)

c. What is the different of the performance between Bayesian optimization and the random search. Why is there this difference? (2 points)

2. MRF parameter identification trough differential evolution

Tune the parameters of the MRF function of the exercise executed in Class#2.

a. Do it using the differential evolution algorithm contained in scipy. Use mean squared error as cost function. (3 points)

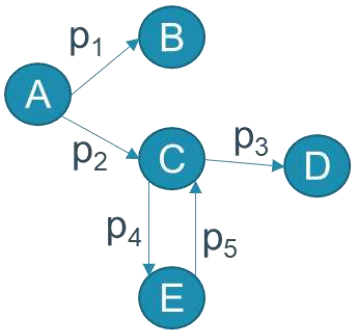
b. Build the cost function using the mean absolute error and train it with the differential evolution algorithm in scipy. What is the difference in the training performances compared to the case with the mean squared error? Why? (3 points)

Exercise 5 – Parameter identification by particle swarm optimization

1. Introduction

Bioinformatics expertise is usually required to address some issues related with slow and expensive processes from the biotechnology area. This type of skills may be used to understand the behavior of certain organisms or biochemicals, such as the case of the parameter estimation problem on the α-pinene isomerization dynamic model. A parameter estimation problem aims to find the parameter values of a mathematical model that gives the best possible fit with existing experimental data. Thus, in a dynamic model based parameter estimation process, an objective functional that gives the mean squared error between the model predicted state values and the observed values is minimized, subject to a set of differential equations. [1]

The α-pinene isomerization was studied in 1973 by Box, Hunter, MacGregor and Erjavec and goals the estimation of a set of five velocity parameters $p_1, \ldots, p_5$ of a homogeneous chemical reaction. The reaction scheme for this chemical reaction that describes the thermal isomerization of α-pinene (A), to dipentene (B), and alloocimen (C), which in turn produces α-and β-pyronene (D), and a dimer (E). Fuguitt and Hawkins observed the concentrations of the reactant and the four products in eight time instants of the interval [0, 36420]. After the chemical reaction orders having been defined, the formulation of the mathematical model describing the concentration of each compound over time may be set. The following system of first order ODEs us assumed to describe the dynamics of the concentration of the compounds over time.

Reactions	Concentration changes
	$\frac{dy_A}{dt} = -(p_1 + p_2)y_A$ $\frac{dy_B}{dt} = p_1 y_A$ $\frac{dy_C}{dt} = p_2 y_A - (p_3 + p_4)y_C + p_5 y_E$ $\frac{dy_D}{dt} = p_3 y_C$ $\frac{dy_E}{dt} = p_4 y_C - p_5 y_E$

The velocity parameters $p_1, \ldots, p_5$ are unknown but can be estimated by swarm intelligence (e.g. particle swarm intelligence). For t ∈ [0, 36420], with the initial conditions

$$y_A(0) = 100, y_B(0) = 0, y_C(0) = 0, y_D(0) = 0, y_E(0) = 0.$$

2. Tasks

a. The experimental observed values are given in the following table. Please enter the data in your python file. Please pay attention to the data type you are using. (2 points)

Time	0.0	1230.0	3060.0	4920.0	7800.0	10680.0	15030.0	22620.0	36420.0
A_exp	100	88.35	76.4	65.1	50.4	37.5	25.9	14.0	4.5
B_exp	0	7.3	15.6	23.1	32.9	42.7	49.1	57.4	63.1
C_exp	0	2.3	4.5	5.3	6.0	6.0	5.9	5.1	3.8
D_exp	0	0.4	0.7	1.1	1.5	1.9	2.2	2.6	2.9
E_exp	0	1.75	2.8	5.8	9.3	12.0	17.0	21.0	25.7

```
#Import the libraries
import numpy as np
from scipy.integrate import solve_ivp

t = np.array([0,1230.0, 3060.0, 4920.0, 7800.0, 10680.0, 15030.0, 22620.0, 36420.0])
A_exp = … #Enter the data here
B_exp = … #Enter the data here
C_exp = … #Enter the data here
D_exp = … #Enter the data here
E_exp = … #Enter the data here
```

b. Please define the functions in python for the concentration changes using the following structure. (3 points)

```
def sim(p1,p2,p3,p4,p5):
    def bfun(t,Y):
        A = Y[0]
        B = Y[1]
        C = Y[2]
        D = Y[3]
```

```
        E = Y[4]
        dAdt = … #Enter the differential equation here
        dBdt = … #Enter the differential equation here
        dCdt = … #Enter the differential equation here
        dDdt = … #Enter the differential equation here
        dEdt = … #Enter the differential equation here
        return np.array([dAdt, dBdt, dCdt, dDdt, dEdt])
    solb = solve_ivp(fun=bfun, t_span = [0,t[-1]-t[0]],y0=[…], #Enter the initial conditions
                     t_eval=t-t[0])
    tnew = solb.t+t[0]
    C1= solb.y[0, :]
    C2= solb.y[1, :]
    C3= solb.y[2, :]
    C4= solb.y[3, :]
    C5 = solb.y[4,:]
    return tnew, C1,C2,C3,C4,C5
```

c. Please define the cost function which gives the error between the experimental observed values and the model predicted values. (5 points)

```
def cost_func(params):
    p1 = params[0]
    p2 = params[1]
    p3 = params[2]
    p4 = params[3]
    p5 = params[4]
    tnew, C1,C2,C3,C4,C5= sim(p1,p2,p3,p4,p5)
    error = … #Enter the cost function
    return error
```

d. Please use the PSO to minimize the error and get optimal velocity parameters $p_1, \ldots, p_5$. Could you try different cost functions, PSO settings and the boundaries? Which kind of cost functions and settings of PSO get the best performance in this case? What about the boundary influence? (6 points)

```
import pyswarms as ps

#Settings for the pso
dimensions = 5
iters = 100
n_particles = 30
options = {'c1': 0.5, 'c2': 0.3, 'w':0.8}

#Boundaries
lower_b = [0.0,     0.0 ,   0.0,  0,  0 ]   #lower bound
upper_b = [5e-4,  5e-4, 5e-4,  5e-4,  5e-4] #upper bound
bounds = (np.array(lower_b ),
               np.array(upper_b))

#Define the function used in PSO
def opt_func(particle):
    n_particles = particle.shape[0]  # number of particles
    dist = [cost_func(particle[i]) for i in range(n_particles)]
    return np.array(dist)

optimizer = ps.single.GlobalBestPSO(n_particles=n_particles, dimensions=dimensions,
                                    options=options, bounds=bounds)
cost, best_pos = optimizer.optimize(opt_func, iters=iters)
```

e. Please give the plot about the concentration changes with the final optimal velocity parameters $p_1, \ldots, p_5$. Is the difference between the experimental and modeled data big? What else can we improve apart from velocity parameters? (4 points)

Exercise 8-11 – Model predictive control of a continuous stirred tank reactor with hybrid modeling

1. Case Description

Continuous stirred tank reactors (CSTRs) often exhibit highly nonlinear dynamics, especially when consecutive and side reactions are present. In this case, a CSTR with Van der Vusse reaction is modelled and controlled.

The production of cyclopentenol (B) from cyclopentadiene (A) by acid-catalysed electrophilic addition of water in dilute solution is presented [2]. Due to the strong reactivity of both the educt and the product, dicyclopentadiene (D) is produced by the Diels-Alder reaction as a side product, and cyclopentanediol (C) as a consecutive product by addition of another water molecule. The complete reaction scheme is as displayed in Fig. 1a), where A is converted into B and D. The B)is converted further into C. In this reaction scheme, B is the product, C and D are unwanted. The three reactions are described by the reaction rate constants k_1, k_2, k_3. Reaction 1 and 2 are the first-order reactions, and Reaction 3 is the second-order reaction.

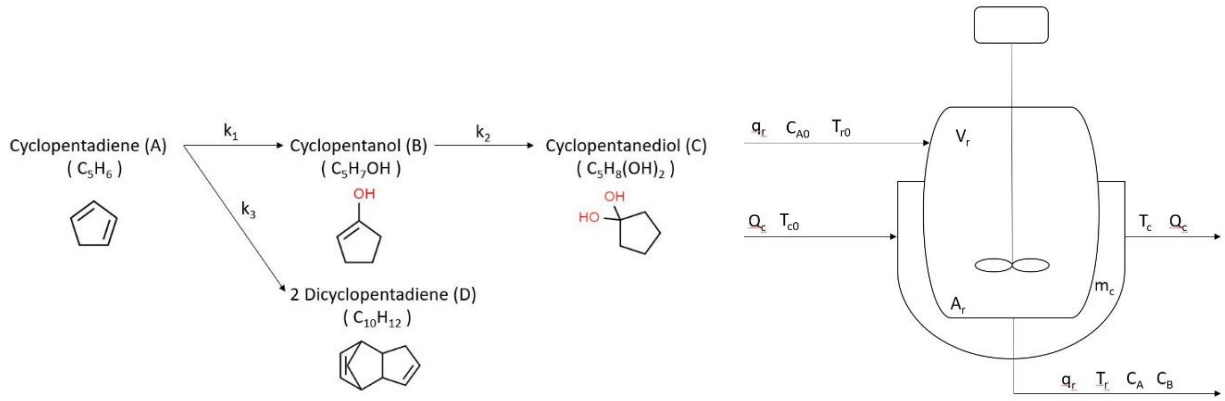


Fig.1: a) Van de Vusse reaction b) Continuous stirred tank reactor

As seen in Fig. 1b), the CSTR consists of a cooling jacket, where inlet flowrate reactor (q_r), inlet concentration of A (C_{A0}), inlet temperature (T_{r0}), inlet temperature of cooling water (T_{c0}), heat exchange by cooling water (Q_c), heat exchange surface (A_r), reactor volume (V_r), reactor temperature (T_r), outlet temperature cooling water (T_c), cooling water mass (m_c), outlet concentration of A (C_A), and outlet concentration of B (C_B) are marked.

2. White-box Model

In this section, a white-box model for CSTR with Van de Vusse reaction should be developed by using the provided parameters in Table 1, 2, 3.

Table 1: Arrhenius parameters

Parameter	Value	Unit

Pre exponential factor 1 (A_1)	$2.145 \cdot 10^{10}$	1 / min
Pre exponential factor 2 (A_2)	$2.145 \cdot 10^{10}$	1 / min
Pre exponential factor 3 (A_3)	$1.5072 \cdot 10^{8}$	1 / min mol
Activation energy 1 (E_{a1})	81130.5	kJ / kmol
Activation energy 2 (E_{a2})	81130.5	kJ / kmol
Activation energy 3 (E_{a3})	71167.8	kJ / kmol

Table 2: CSTR process parameters

Parameter	Value	Unit
Volume reactor (V_r)	0.01	m^3
Inlet concentration of A (C_{A0})	5.1	kmol / m^3
Inlet temperature (T_{r0})	330	K
Heat exchange surface (A_r)	0.215	m^2
Cooling water mass (m_c)	5	kg
Heat coefficient (U_r)	67.2	kJ / min m^2 K
Inlet flowrate reactor (q_r)	$2.365 \cdot 10^{-3}$	m^3 / min
Density of the mixture inside the reactor (ρ)	934.2	kg / m^3
Heat capacity of the mixture inside the reactor (C_p)	3.01	kJ / kg K
Heat capacity of the cooling (C_{pc})	2.0	kJ / kg K
Gas constant (R)	8.314	kJ / K kmol

Table 3: Reaction enthalpies

Parameter	Value	Unit
Reaction enthalpy 1 (h_1)	-4200	kJ / kmol
Reaction enthalpy 2 (h_2)	11000	kJ / kmol
Reaction enthalpy 3 (h_3)	41850	kJ / kmol

The differential equations to simulate the CSTR are:

$$\frac{dC_A}{dt} = \frac{q_r}{V_r} \cdot (C_{A0} - C_A) - k_1 C_A - k_3 C_A^2$$

$$\frac{dC_B}{dt} = -\frac{q_r}{V_r} \cdot C_B + k_1 C_A - k_2 C_B$$

$$\frac{dC_C}{dt} = -\frac{q_r}{V_r} \cdot C_C + k_2 C_B$$

$$\frac{dC_D}{dt} = -\frac{q_r}{V_r} \cdot C_D + k_3 C_A^2$$

$$\frac{dT_r}{dt} = \frac{-h_r}{\rho \cdot C_p} + \frac{U_r \cdot A_r (T_c - T_r)}{\rho \cdot V_r \cdot C_p} + \frac{q_r \cdot (T_{r0} - T_r)}{V_r}$$

The enthalpy of the system (h_r) can be composed of three reactions, given by:

$$h_r = h_1 \cdot k_1 \cdot C_A + h_2 \cdot k_2 \cdot C_B + h_3 \cdot k_3 \cdot C_A^2$$

The cooling temperature (Tc) is considered as constant, and Arrhenius equation is employed to calculated the reaction rate constants:

$$k_i = A_i e^{-\frac{Ea_i}{RT}}, i = 1, 2, 3$$

a. Please plot the concentrations of A, B, C, D, reactor temperature, and jacket temperature in 30 minutes, at the condition (Ca_initial = 5.1 kmol/m³, Cb_initial = 0 kmol/m³, Cc_initial = 0 kmol/m³, Cd_initial = 0 kmol/m³, Tr_initial = 330K, Tc = 330K). Please pay attention to the units you are using. (3 points)

b. Could you plot the attached experimental data for the same condition as 2.1 in "Actual_data.csv" as well? (1 point)

c. Please compare the experimental data and the modeled data. Do you have some ideas to improve the current model? (1 points)

3. Black-box Model

In section 2, we used Arrhenius equations to calculate the reaction rates. While some reaction rates cannot be represented by Arrhenius equations. In this section, we will try to get the reaction rates (r_A, r_B, r_C, r_D) and the enthalpy of the system (h_r) from the concentrations (C_A, C_B, C_C, C_D) and the temperature (T_r) via machine learning approaches. The available data should be split into 70% for the training and the other 30% for the testing.

a. Please get the reaction rates (r_A, r_B, r_C, r_D) and the enthalpy of the system (h_r) based on the provided data using MRF integrated with particle swarm optimization. (2 points)

b. Please get the reaction rates (r_A, r_B, r_C, r_D) and the enthalpy of the system (h_r) based on the provided data using ANN. For the ANN approach, the scaling of the data may be necessary.(2 points)

c. Please assess the resulting models from 3.1-3.2 by the testing set with determination coefficients R^2 and mean squared errors. (2 points)

4. Hybrid Model

In section 3, we have developed black-box models for the reaction rates and the system enthalpy based on the provided data through MRF or ANN. In section 2, we can see there is still some improvement space for the white-box model with Arrhenius equations. The hybrid models usually take both advantages of black-box and white-box models. In this section we will establish hybrid models with developed ANN model and refinement layer.

a. Please save the obtained models or scalers from section 3. The following hints give how to save and load scalers with "pickle" and models in "keras" . (1 point)

```
#Save the ScalerX with the file name "ScalerX.pickle"
with open('ScalerX.pickle', 'wb') as f:
    pickle.dump(ScalerX, f)

#Load "ScalerX.pickle" as ScalerX
ScalerX = pickle.load(open( "ScalerX.pickle", "rb" ))
```

```
model.save('fit_model.h5')
fit_model = keras.models.load_model('fit_model.h5')
```

b. Please combine them with the differential equations in section 1 to develop the hybrid model of the CSTR. Give the plots of concentrations and temperatures in 30 minutes and compare the newly modelled data and provided actual data. (1 point)

c. Another method to establish a hybrid model is through a refinement layer. Please calculate the deviations between "Actual_data.csv" and the data from the white-box model in section 2. Please make sure the time steps are the same. Please develop a hybrid based on the refinement layer, and give the plots of concentrations and temperatures in 30 minutes. (2 points)

d. What about using "Actual_data_100.csv" with 100 samples to train your refinement layer? Does it give a better performance than the one in section 4.3? Why? (1 point)

e. Please compare the hybrid models you have obtained in 4.2, 4.3, and 4.4 by calculating MSE or R^2 (1 point)

5. Model Predictive Control

This reactor is dynamically modeled as a Continuously Stirred Tank Reactor (CSTR) as shown in section 2. Please use the provided demo codes to control the reactor temperature of the CSTR. The estimated time is 60 min. It is desired to maintain the reactor temperature at setpoints by adjusting the jacket temperature (Tc). The heating/cooling jacket temperature can be adjusted between 300 K and 500 K. The reactor temperature is desired at 450 K at the first 20 min, at 350 K at the second 20 min, at 400 K at the third 20 min. The time step can be every 1 min.

a. Please fill in the empty parts (marked in yellow) in the demo codes using Gekko [3] and give the plots. (1 point)

b. Could you also give the plots if the m.T.DMAHI = 50 or 100 (marked in red)? Why is the performance different? (1 point)

c. Based on the previous study, if we would like to maximize our product B. Which jacket temperature should we set? (1 point)

```
#%%Import the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import keras
import pickle
from scipy.integrate import odeint
import pathlib
file_path = pathlib.Path().absolute()
from gekko import GEKKO

#%%Define the parameters and your reator model (for the easy calculation,
#we can use the white-box model from section 2)
k10 = 2.145*10**10 #2.145e10
k20 = 2.145e10
k30 = 1.5072e8
Ea1 = 81130.5
Ea2 = 81130.5
Ea3 = 71167.8

Vr = 0.01
Ca0 = 5.1
Tr0 = 330
Ar = 0.215
qr = 2.365e-3
rho = 934.2
Cp = 3.01
Cpc = 2
R = 8.314
h1 = -4200
```

```
h2 = 11000
h3 = 41850
Ur = 67.2
Tc = 330

def reactor(S,t,Tc):
    Ca = S[0]
    Cb = S[1]
    Cc = S[2]
    Cd = S[3]
    Tr = S[4]

    #Enter the differential equations, you can copy it from section 2
    dCadt = ...
    dCbdt = ...
    dCcdt = ...
    dCddt = ...
    dTrdt = ...

    return [dCadt,dCbdt,dCcdt,dCddt,dTrdt]

#%%Read the actual data, we use the end of the actual data as the new starting points
S_actual=np.array(pd.read_csv(file_path/'Actual_data.csv'))
tspan_actual = S_actual[:,1]
Ca_actual = S_actual[:,2]
Cb_actual = S_actual[:,3]
Cc_actual = S_actual[:,4]
Cd_actual = S_actual[:,5]
Tr_actual = S_actual[:,6]

#%%
#Current jacket temperature
u_ss = Tc
# Feed Temperature (K)
Tf = Tr0
# Feed Concentration (kmol/m^3)
Caf = Ca0
Cbf = 0
Ccf = 0
Cdf = 0
# Current Steady State Conditions
Ca_ss = Ca_actual[-1]
Cb_ss = Cb_actual[-1]
Cc_ss = Cc_actual[-1]
Cd_ss = Cd_actual[-1]
T_ss = Tr_actual[-1]
S0 = np.zeros(5)
S0[0] = Ca_ss
S0[1] = Cb_ss
S0[2] = Cc_ss
S0[3] = Cd_ss
S0[4] = T_ss
#Use the library Gekko
m = GEKKO(remote=False)

#Set the jacket temperature changing range
Tc_min = ...
Tc_max = ...

"""
MV
Manipulated variables, jacket temperature
STATUS = 1, MV is calculated by the optimizer
FSTATUS =0,the measurement should not be used either in estimation or in updating a
    parameter in the model
```

```
"""
m.Tc = m.MV(value=Tc,lb=Tc_min,ub=Tc_max)
#MV tuning
m.Tc.STATUS = 1
m.Tc.FSTATUS = 0
 """
 DMAX
 Delta MV maximum step per time step,
 Applies a hard constraint that prevents the MV from being changed by more
 than the specified value in one time step.
 """
 m.Tc.DMAX = 100    #General
 m.Tc.DMAXHI = 20   # slow action up
 m.Tc.DMAXLO = -100 # quick action down
 """
 CV
 Controlled variables, reactor temperature
 STATUS  = 1, CV is calculated by the optimizer
 FSTATUS = 1, an objective function is added to minimize the model prediction to the
measurements
 TR_INIT = 1, Setpoint trajectory initialization,re-center with coldstart/out-of-service
 """
 m.T = m.CV(value=T_ss)
 #CV tuning
 m.T.STATUS = 1
 m.T.FSTATUS = 1
 m.T.TR_INIT = 1
 m.T.TAU = 1     #Time constant for controlled variable response
 DT = 0.5        # acceptable target region

 """
 Var
 Calculated by solver to meet constraints
 """
 m.Ca = m.Var(value=Ca_ss)
 m.Cb = m.Var(value=Cb_ss)
 m.Cc = m.Var(value=Cc_ss)
 m.Cd = m.Var(value=Cd_ss)

 #Define the euqations used for the control
 #for example, dCadt can be defined as
 m.Equation(m.Ca.dt() == qr/Vr*(Caf - m.Ca) - k10*m.exp(-Ea1/(R*m.T))*m.Ca - k30*m.exp(-
Ea3/(R*m.T))*m.Ca**2)
 m.Equation(m.Cb.dt() == ...)
 m.Equation(m.Cc.dt() == ... )
 m.Equation(m.Cd.dt() == ...)
 m.Equation(m.T.dt() == ...)

 """
 CV_TYPE=1, the objective is the absolute error.
 """
 m.options.CV_TYPE = 1
 """
 Model Predictive Control (MPC) is implemented with IMODE=6 as a simultaneous solution
 """
 m.options.IMODE = 6
 """
 Solver slection: Interior Point OPTimizer (IPOT)
 """
 m.options.SOLVER = 3

 """
 time
 Sets the time array indicating the discrete elements of time discretization
 """
```

```
158.  m.time = [0,1]
159.
160.  # define the Time Interval (minutes)
161.  t = np.linspace(0,...,...)
162.
163.  # Store results for plotting
164.  Ca = np.ones(len(t)) * Ca_ss
165.  Cb = np.ones(len(t)) * Cb_ss
166.  Cc = np.ones(len(t)) * Cc_ss
167.  Cd = np.ones(len(t)) * Cd_ss
168.  T = np.ones(len(t)) * T_ss
169.  Tsp = np.ones(len(t)) * T_ss
170.  u = np.ones(len(t)) * u_ss
171.
172.  # Set point for the jackete temperature
173.  Tsp[0:20] = ...
174.  Tsp[20:40] = ...
175.  Tsp[40:60] = ...
176.
177.  # Create plot
178.  plt.figure()
179.  plt.ion()
180.  plt.show()
181.  for i in range(len(t)-1):
182.      # simulate one time period
183.      ts = [t[i],t[i+1]]
184.      # Simulate CSTR, use the predefind function
185.      S = odeint(reactor,S0,ts,args=(u[i],))
186.
187.      # retrieve measurements
188.      Ca[i+1] = S[-1][0]
189.      Cb[i+1] = S[-1][1]
190.      Cc[i+1] = S[-1][2]
191.      Cd[i+1] = S[-1][3]
192.      T[i+1] = S[-1][4]
193.
194.      # insert measurement
195.      m.T.MEAS = T[i+1]
196.      # solve MPC
197.      m.solve(disp=True)
198.      #The setpoint high (SPHI) is the upper final target value
199.      m.T.SPHI = Tsp[i+1] + DT
200.      #The setpoint low (SPLO) is the lower final target value
201.      m.T.SPLO = Tsp[i+1] - DT
202.
203.      # retrieve new Tc value
204.      u[i+1] = m.Tc.NEWVAL
205.      # update initial conditions
206.      S0[0] = Ca[i+1]
207.      S0[1] = Cb[i+1]
208.      S0[2] = Cc[i+1]
209.      S0[3] = Cd[i+1]
210.      S0[4] = T[i+1]
211.
212.      #% Plot the results
213.      plt.clf()
214.      plt.subplot(2,1,1)
215.      plt.plot(t[0:i],Ca[0:i],'b-',linewidth=1,label=r'$C_A$')
216.      plt.plot(t[0:i],Cb[0:i],'r-',linewidth=1,label=r'$C_B$')
217.      plt.plot(t[0:i],Cc[0:i],'g-',linewidth=1,label=r'$C_C$')
218.      plt.plot(t[0:i],Cd[0:i],'k-',linewidth=1,label=r'$C_D$')
219.      plt.ylabel('Concentration \n($kmol/m^3$)')
220.      plt.legend(loc='center left')
221.
222.      plt.subplot(2,1,2)
```

```
        plt.plot(t[0:i],Tsp[0:i],'k-',label=r'$T_{sp}$')
        plt.plot(t[0:i],T[0:i],'b-',label=r'$T_{meas}$')
        plt.plot(t[0:i],u[0:i],'r-',label=r'$T_{jacket}$')
        plt.ylabel('Reactor \nTemperature (K)')
        plt.xlabel('Time (min)')
        plt.legend(loc='center left')
        plt.draw()
        plt.pause(0.5)

```

References

[1] A. M. A. C. Rocha, M. C. Martins, M. F. P. Costa, and E. M. G. P. Fernandes, "Direct Sequential Based Firefly Algorithm for the α-Pinene Isomerization Problem," 2016.

[2] K. U. Klatt and S. Engell, "Gain-scheduling trajectory control of a continuous stirred tank reactor," *Comput. Chem. Eng.*, vol. 22, no. 4-5 /5, pp. 491–502, 1998, doi: 10.1016/S0098-1354(97)00261-5.

[3] APMonitor, "GEKKO Python Tutorials," 2022. [Online]. Available: http://apmonitor.com/wiki/index.php/Main/GekkoPythonOptimization. [Accessed: 22-Sep-2022].