Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Nghia Van Nguyen

Production Assurance & Reliability Modelling Tools Comparison - A Case Study on A 4-bed Pressure Swing Adsorption System in Blue Hydrogen Production

Master's thesis in Reliability, Availability, Maintainability and Safety (RAMS)
Supervisor: Jørn Vatn
June 2023

NTNU
Norwegian University of Science and Technology

Nghia Van Nguyen

Production Assurance & Reliability Modelling Tools Comparison - A Case Study on A 4-bed Pressure Swing Adsorption System in Blue Hydrogen Production

Master's thesis in Reliability, Availability, Maintainability and Safety (RAMS)
Supervisor: Jørn Vatn
June 2023

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

Preface

This Master's thesis concludes my Master of Science degree in the Reliability, Availability, Maintainability and Safety (RAMS) program at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. This work was a continuation of my specialization project during the autumn 2022 when I studied and employed two reliability modelling tools to model a subsystem in blue hydrogen production plants. Using fundamental knowledge from the autumn project, the Master's thesis continues to experiment one more modelling tool. The thesis was completed in the spring semester of 2023 under the supervision of Professor Jørn Vatn from the RAMS department at NTNU. The study was inspired by SUBPRO-Zero Centre who conducts research that accelerates the energy transition for offshore industry.

This thesis assumes that the readers are master students from the RAMS program or experts having background knowledge in reliability, availability, and maintenance management. Some courses at NTNU that can provide this knowledge are (1) TPK4120 - Industrial safety and reliability, (2) TPK4140 - Maintenance Management, and (3) TPK450 - Data-driven prognostics and predictive maintenance.

Trondheim, 2023-06-02

Nghia Van Nguyen

Acknowledgment

This Master's Thesis represents the ending of several years of pursuing my higher education. I feel myself grateful for everything I have learnt and for everyone I have met on this unforgettable journey.

To my supervisor Professor Jørn Vatn of the department of Mechanical and Industrial Engineering at NTNU, I would like to express my deep sense of gratitude to you for your supervision. Thank you for your valuable input and constructive feedback during the process. Without your enthusiasm, patience, and insights within RAMS engineering, it would have been a challenge to finish this Master's thesis.

I would like to thank Thor Inge Bernhardsen and his colleagues at Equinor for providing valuable guidance at the beginning of the project. They helped to define the challenges faced by the energy industry and motivated me to conduct the research. In addition, my special thanks goes to Christophe Spaggiari at Peloton for training me to use one of the most common modelling tools and clarifying my curiosity on the way I employed the tool in this study.

To my friends and fellow students, thank you all. We have shared together joy and despair, inspiration and procrastination. I treasure every moment we had lunch-breaks and hung out together exploring the beautiful city, Trondheim, and its very rich culture. You made my student life more rewarding.

Finally, to my family. Studying abroad without any family member living in Norway is one of the most challenging experiences in my life. But you have always cheered for me and reminded me the reason why I have started this journey. I was blessed to receive your phone calls and messages everyday. They truly gave me strength and motivation to complete this Master program. Thank you!

N.V.N.

Executive Summary

This Master's thesis speculates upon production assurance in subsea blue hydrogen production. The future of energy is low carbon, and subsea blue hydrogen can help to shape this future. Hence, it has been an increased recognition that more attention needs to be paid to this form of energy carrier.

One major challenge associated to subsea blue hydrogen is high production cost. This issue continues to be an open problem that affects the possibility of producing blue hydrogen in large scale. As the rule of thumb, redundancy reduction is one of the simplest and most effective ways to cut cost. This, nevertheless, leads to another challenge of achieving high reliability of the production plant while reducing redundancy in some components and maintaining redundancy in critical components. A potential solution to this difficult problem could involve optimizing the configuration of production lines from a reliability and cost perspective. To that end, production assurance analysis is required.

Production assurance analysis is achieved primarily through the use of production availability. In order to access a production availability of a production line, system modeling, or more specifically, availability modeling must be employed. Some conventional methods have been developed for this purpose. However, these methods have certain limitations in terms of modelling "multi-state" systems, complex configurations, and realistic maintenance strategies. Dynamic reliability models such as Markov process might be used to capture some aspects of system behaviors in real life. To overcome the difficulties that analytical methods face with, simulation-based (simulation) methods utilized Python or MIRIAM RAM Studio tool seem to be effective. In recent years there has been an increased interest in utilizing simulation methods for calculating production availability in the industry. The question of how flexible these methods are compared to analytical methods is the area to which this thesis now turns.

Given that there is a relatively small body of literature that is concerned with comparison of flexibility in reliability modelling tools, this study aims to add more volume to this area of research. The study starts with reviewing existing comparisons of different system reliability modelling tools in literature. Then, a more comprehensive comparison is introduced based on a practical case study on a subsystem in the blue hydrogen production plant. Modelling tools used in this work are Markov process, discrete event simulation in Python, and MIRIAM RAM Studio. The case study specifically concentrates on building reliability models and accessing production availability of a 4-bed pressure swing adsorption unit. This system is worth studying because of its dynamic property in operational modes. In the course of this work we discovered that Markov process is less flexible than MIRIAM RAM Studio and discrete event simulation in Python. However, the ability to handle complex systems goes along with a computational cost, which is the case in Python and MIRIAM. These results corroborate the findings of a great deal of the previous work. Yet, considerably more work will need to be done.

Sammendrag

I denne masteroppgaven er det lagt vekt på produksjonssikkerhet i undervannsproduksjon av blått hydrogen. I fremtiden må energi i stadig større grad komme fra lavkarbonkilder, og blått hydrogen kan bidra til å forme den fremtiden. Derfor har det vært en økt erkjennelse av at mer oppmerksomhet må rettes mot denne formen for energibærer.

En stor utfordring knyttet til undervannsblått hydrogen er høy produksjonskostnad. Dette problemet påvirker muligheten for å produsere blått hydrogen i stor skala. Som tommelfingerregel er redundansreduksjon en av de enkleste og mest effektive måtene å kutte kostnader på. Dette fører likevel til en annen utfordring med å oppnå høy pålitelighet av produksjonsanlegget samtidig som redundans i enkelte komponenter reduseres og redundans i kritiske komponenter opprettholdes. En potensiell løsning på dette vanskelige problemet kan innebære å optimalisere konfigurasjonen av produksjonslinjer fra et pålitelighets- og kostnadsperspektiv. For det formål kreves det en produksjonssikkerhetsanalyse.

Produksjonssikkerhetsanalyse oppnås primært gjennom bruk av produksjonstilgjengelighet. For å få tilgang til en produksjonstilgjengelighet for en produksjonslinje, må tilgjengelighetsmodellering brukes. Noen konvensjonelle metoder er utviklet for dette formålet. Imidlertid har disse modellene visse begrensninger når det gjelder modellering av "multi-state" systemer, komplekse konfigurasjoner og realistiske vedlikeholdsstrategier. Dynamiske pålitelighetsmodeller som Markov prosessen kan brukes til å fange opp noen aspekter ved systematferd i virkeligheten. For å overvinne vanskelighetene fra analytiske metoder, ser simuleringsmetoder som Python og MIRIAM RAM Studio ut effektive. De siste årene har det vært en økt interesse for å dra nytte av simuleringsbaserte metoder for å beregne produksjonstilgjengelighet i industrien. Spørsmålet om hvor fleksible disse metodene er sammenlignet med analytiske metoder er det området vi sikter oss inn mot.

I og med at det er en tilkortkommenhet av forskning som omhandler sammenligning av fleksibilitet i pålitelighetsmodelleringsverktøy, er studiets hovedmål å legge mer volum til dette forskningsområdet. Med den hensikten ble det gjort en nøyaktig litteraturstudie å skaffe en oversikt over sammenligninger av ulike modelleringsverktøy. Deretter introduseres en mer omfattende sammenligning basert på en praktisk case-studie på et system i produksjonsanlegget for blå hydrogen. Modelleringsverktøy som brukes i dette arbeidet er Markov prosessen, diskret hendelsessimulering i Python og MIRIAM RAM Studio. Case-studien ble utført for å bygge pålitelighetsmodeller og få tilgang til produksjonstilgjengeligheten av en trykksvingningsadsorpsjonsenhet. Dette systemet er verdt å studere grunnet sitt dynamiske egenskap i driftsmoduser. Basert på funnene forfekter vi at Markov-prosessen er mindre fleksibel enn MIRIAM og diskret hendelsessimulering i Python. Evnen til å håndtere komplekse systemer går imidlertid sammen med en beregningskostnad, som er tilfellet i Python og MIRIAM. Disse resultatene bekrefter funnene fra mye av det tidligere arbeidet. Likevel er mer videre arbeid anbefalt.

Contents

Chapter 1

Introduction

In this chapter, the background for the problem at stake is presented, along with a description of the objectives to be defined. Next, the scope and limitations associated with the study are presented. The remainder of the chapter consists of a structural overview of the thesis.

1.1 Background

The global demand for hydrogen in the future is expected to increase significantly. Hydrogen is considered as a carbon-free energy carrier (Van Cappellen et al., 2018) that can help to decarbonise the industry. Therefore, it is of interest for many countries around the world to reach net-zero greenhouse gas (GHG) emissions by 2050 that they have committed to. In a rapid global energy transition context, global hydrogen demand is anticipated to reach up to 660 Megatonne in 2050 (Wappler et al., 2022). Annual hydrogen production in the world is about 75 Megatonne, of which 76% is generated from natural gas and 23% from coal according to International Energy Agency (IEA, 2019). While green hydrogen generation produced by electrolyzers supplied by renewable electricity is capital-intensive (Noussan et al., 2021), hydrogen generated from fossil fuels, e.g., natural gas, coal, and oil, with carbon capture technology, so-called blue hydrogen, seems to be economically advantageous and feasible to be implemented in industry (Van Cappellen et al., 2018). Accordingly, the production of blue hydrogen is currently one of the hottest topics in the field of renewable energy research.

Blue hydrogen can provide a balanced energy transition for the future (Irena, 2019). When hydrogen is produced by fossil-based solutions, e.g., steam methane reforming (SMR), autothermal reforming (ATR), partial oxidation, and coal gasification, coupled to Carbon Capture and Storage (CCS), it can help to decrease most of their GHG emissions and therefore is identified with the prefix "blue" (IEA, 2019). CCS-enabled pathways prove to be a smart way to supply hydrogen to the world while waiting for future cost reductions in renewable energy and electrolyzers to be intensively mature. The demand for blue hydrogen shows a potential for countries with

natural gas resources to produce clean hydrogen and secure the hydrogen sector (Irena, 2019). Not only is the rapid energy transition needed to be accelerated, but socio-economic balance should be taken into consideration given the recent job losses in the hydrocarbon energy industry. Hydrocarbon energy transition with CCS technology will ease these socio-environmental challenges (Cloete et al., 2022).

Researchers have recently paid a lot of attention towards blue hydrogen technology innovation. Conventionally, there are two ways of producing blue hydrogen, namely Steam Methane Reforming (SMR) and Autothermal Reforming with a Gas Heated Reformer (ATR+GHR), both of them require CCS to decarbonise their processes. A comprehensive comparison of techno-economic and greenhouse gas (GHG) emissions aspects for natural gas-based blue hydrogen production technologies (SMR and ATR) can be found in Oni et al. (2022). Up-to-date techno-economic analysis and life cycle assessment of these two technologies with CO_2 capture are reported in Santos et al. (2017) and Khojasteh Salkuyeh et al. (2017). Sorption-enhanced steam reforming (SE-SMR) involving an in-situ CO_2 capture process is claimed to be an innovative technology in producing decarbonised and high-purity hydrogen (Clough et al., 2018; Yan et al., 2020). A process simulation for a gas switching reforming (GSR) process for hydrogen production with integrated CO_2 capture is introduced in Nazir et al. (2019), showing a small energy penalty compared to SMR and a potential to scale up. Large energy penalty which is usually associated with the air separation unit can be eliminated by chemical looping reforming (CLR) technology (Rydén et al., 2006). In general, many alternative technologies to produce blue hydrogen can be found and their feasibility depends on important factors such as feedstock availability, technology readiness level, and economy. So far, researchers have focused on how to improve the performance of blue hydrogen plants that are located onshore; the idea of moving such plants to the sea bed in the open literature remains primitive.

In blue hydrogen production, pressure swing adsorption (PSA) is an important unit. It separates hydrogen from impurities in synthesis gas, providing high-purity hydrogen as feedstock for various applications. PSA system is preferable since its low energy consumption, costs and precision in hydrogen separation (99.99%) (Ruthven and Knaebel, 1994). Also, PSA processes are easily operated because they require neither rotating equipment nor circulation solutions (Yang, 1987). An adsorber in a PSA unit normally works in a sequence of four steps (adsorption, depressurization, purge, and repressurization). PSA is then usually designed to have several adsorbers (beds or vessels), so-called "polybed", allowing multiple beds to undergo the adsorption step simultaneously while others are regenerated. By doing that, it assures a highly continuous throughput. Having a large number of adsorption beds can improve the performance of PSA but also make the operation more complex with complicated step sequences. The first commercial hydrogen PSA unit was installed in 1966 and it was composed of 4 adsorbers (Yang, 1987). At the moment, there are about 1000 polybed hydrogen PSA in operation around the world with up to

16 beds (Luberti and Ahn, 2022). A summary of PSA technology development in the first 30 years since 1976 is conducted in Stöcker et al. (1998). More recent polybed hydrogen PSA processes are systematically reviewed in Luberti and Ahn (2022).

General speaking, a system could be modeled from many points of view such as availability, maintainability, logistics, risk, and human error (Kawauchi and Rausand, 2002). For PSA processes, there is a great deal of effort that goes into developing and analyzing mathematical and chemical models of the system, but not from a RAMS engineering perspective. Mathematical and chemical models usually involve quantities such as temperature, pressure, flow rates of fluids and gasses, and acceleration of particles. They heavily rely on the theory and techniques related to differential and difference equations. As a result, hydrogen purification performances, e.g., purity, recovery, productivity, are usually evaluated by solving a number of equations of mass balance, energy balance, momentum balance, pressure gradient, and adsorption rate (see Yang and Doong (1985); Biswas et al. (2010); Ribeiro et al. (2012); Luberti et al. (2014); Xiao et al. (2015, 2018); Martínez et al. (2022)). These models are valuable for the modification of feeding flow rate, adsorbent properties parameters, adsorption bed characteristics, adsorption pressure, and so forth. Other aspects that impact hydrogen production externally such as availability of resources, weather constraints, and maintenance strategies are left uncovered. What is lacking in the modelling world of PSA processes is reliability models scrutinizing the reliability and availability of the system based on reliability data, i.e., failure rate and mean time to repair.

Reliability modelling and simulation is an approach that helps to perform production assurance in production systems. Two ways of establishing reliability models are the analytical method and the simulation-based method. Analytical approach, e.g., fault tree analysis (FTA), and Markov chains, is preferably applied for simple systems. Hokstad (1988) introduces an approximation calculation of production availability based on a failure model of an oil and gas production system. This approach is limited by two assumptions: the system is not in a time-dependent state (1), and multiple repairs are not carried out during the same intervention (2). Vesteraas (2008) shows the calculation of production availability by using Markov chains. The idea of this approach adapted from Kawauchi and Rausand (2002) is that the system under consideration can be divided into subsystems, and the production availability can be accessed by measuring the throughput capacity distribution and applying a simple merging rule for parallel and series subsystems. This approach is quite flexible giving the possibility of modelling different maintenance strategies and systems of dependent components. The limitation of analytical methods, in general, is the assumptions laid on the system for the sake of simplicity. Simulation-based approach relying on Monte-Carlo simulation whose events, e.g., failures and repairs, occur in the system with a defined time delay can give more flexibility in describing the realistic behaviour of systems. Borgonovo et al. (2000) uses the Monte Carlo simulation as a flexible tool to calculate the system unavailability of complex systems with periodic maintenance

strategies and a limited number of repair teams. To achieve realistic models for evaluation of the production availability of multi-state, multi-output plants, Zio et al. (2006) also applies a Monte Carlo simulation model. To date, extensive research has been carried out on detailed investigation of individual modelling tools. However, there are few studies that have compared most commonly used modelling tools from both analytical and simulation-based method.

To the best of our knowledge, subsea blue hydrogen production systems in general and PSA processes, in particular, have not been studied in the open literature from the RAMS engineering point of view. Therefore, a reliability model of PSA is lacking in the open literature, leading to the need for a production assurance study of blue hydrogen production. In addition, reliability modelling comes along with confusion in choosing the right tool and method to better model a specific system. Researchers have been so far concentrating on studying and improving individual tools, resulting in a gap when it comes to a systematic comparison of different modelling tools corresponding to different constraints. That requires us to fill the gap and increase the volume of study in production assurance of PSA processes.

1.2 Objectives

The main goal of the current study is to establish a comparison of different reliability modelling tools from analytical approach such as Markov process, and simulation-based approach, e.g., discrete event simulation in Python and MIRIAM RAM Studio. To illustrate such comparison, making models for a case study is required. A 4-bed pressure swing adsorption (PSA) unit in blue hydrogen production is selected to be the case study. This intention defines the second main objective which is to review the state-of-the-art technology in producing and purifying hydrogen. Also, reliability data such as mean time to failure and mean time to repair are required to make these models. In short, objectives of the present study can be summarized as follows:

1. Explore state-of-the-art technology in blue hydrogen production
2. Study pressure swing adsorption (PSA) process, focus on a 4-bed PSA unit as a case study
3. Perform a literature review of reliability modelling methods and modelling tool comparison
4. Acquire reliability data for components in the 4-bed PSA unit
5. Build reliability models to calculate production availability for the 4-bed PSA unit and compare the results from different models

1.3 Approach

This section outlines a concrete overview of the approach used in this work to meet the above objectives.

The first three tasks defined in the previous section are grouped into a so-called "literature review" task. The approach to conduct a systematic literature review of up-to-date blue hydrogen technology and reliability modelling tools consists of 6 steps (see Figure 1.1): (1) select keywords associated with the main topic, (2) search for articles and documents that contain the keywords on several research platforms, e.g., ORIA, web of science, ScienceDirect, Engineering Village, IEEE transactions, and Google Scholar, (3) relevant articles and documents are shortlisted by reading the abstracts, ranking the number of citations and publishing date, and sorting out the fields of study, (4) analyze highly relevant articles to summarize the message that these articles are supposed to convey, along with citing literature properly, (5) evaluate the results, if not satisfied then go back to step (1), and (6) report summaries of the literature for inclusion in the research report.

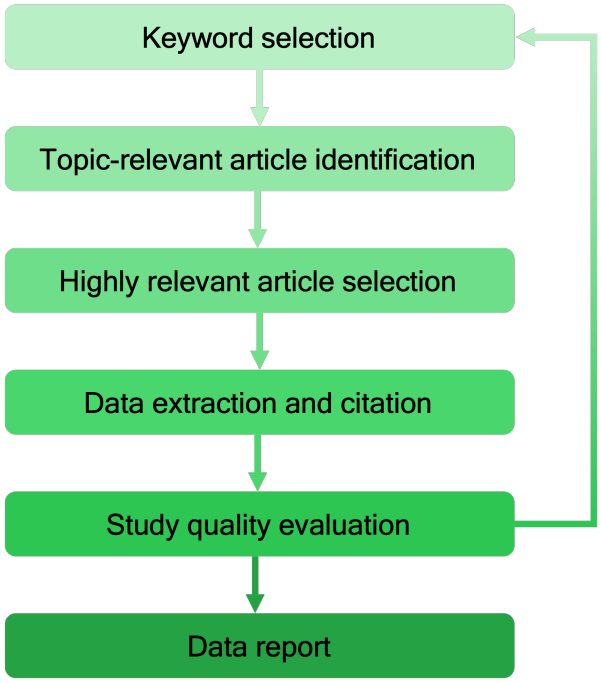


Figure 1.1: Approach to conduct a literature review.

The two last objectives related to modelling part are achieved by applying the following approach. Firstly, reliability data including mean time to failure and mean time to repair of components in a PSA system are collected based on OREDA (Offshore and onshore REliability DAta project) database. Secondly, properties and production requirements of the PSA unit are defined. Thirdly, reliability models of the PSA are created by using various tools including Markov

process, Python with SimPy library, and MIRIAM RAM Studio. Finally, the results from these models are analysed and compared to conduct a final comparison of different reliability modelling methods.

1.4 Contributions

The contributions of this study are twofold.

The first contribution of this Master's thesis is a comparison of different reliability modelling tools in the light of modelling a complex, dynamic subsystem in hydrogen production, i.e., the pressure swing adsorption (PSA) system, as a case study. This comparison helps to better understand the applicability and flexibility of various reliability modelling tools in RAMS engineering. By doing that, the study bridges the gap that exists in the open literature when most articles tend to cover individual reliability modelling tools.

The second contribution of the study is that it provides insights into how to improve the PSA process. These insights will, hopefully, turn into tangibly increased production availability in the future blue hydrogen sector. To that end, production availability of the PSA unit will be calculated in two scenarios (1) without a buffer and (2) with a buffer. The purpose of doing this is to challenge the flexibility of the modelling tools in addressing the dynamic of the buffer. Also, this is meant to test the benefit of having a buffer to support the PSA system as needed. The findings of this study are worth considering in production assurance of hydrogen production in real life since it can be applied and expanded for multi-bed PSA units.

1.5 Limitations

Since the study was limited to investigate the production availability of the PSA unit based on reliability models, physical and mathematical models of the PSA are not part of the scope of this research. Physical and mathematical models require different background than the knowledge in RAMS engineering, and hence might be time-consuming for RAMS students to make these models. Accordingly, chemical reactions in general and adsorption properties in particular inside the PSA process are not addressed in this work.

Another limitation of this study is that reliability data used as input for modelling the case study are rather generic. Different types of pressure swing adsorption processes with modified structures and adjusted technology are used worldwide. The PSA unit used in the case study does not represent any typical version of commercial PSA processes. Also, reliability data of components in the PSA unit are collected from different systems and data resources, regardless of the environmental and operational conditions. Uncertainty about such data might exist but it is not examined in this research.

Due to the lack of time and resources, it is unfortunate that the study did not include a large number of reliability modelling tools. Only one analytical method and two simulation tools are used in the case study. In fact, some commercial software with simulation features require licence, not to mention the time and necessary guidance needed to practice using them. Thus, we selected some of the most common tools in the industry to study in this work.

1.6 Outline

This Master's thesis includes six chapters. The first chapter presents the background for the topic under study, the main objectives and corresponding approaches, along with limitations related to the work. Next, Chapter 2 presents some theoretical backgrounds including the concept of production assurance, relevant definitions, and a literature review of reliability modelling tools comparison. This is followed by a description of two blue hydrogen production technologies and one of the most important subsystems in hydrogen production, namely the pressure swing adsorption (PSA) system. After looking into the literature review and state-of-art technology in blue hydrogen, a case study is carried out in Chapter 4. In this chapter, a 4-bed PSA system with(out) a buffer is modeled by several modelling tools. By having reliability models, production availability of the PSA system is accessed. Then, discussions about these findings are drawn and summed up in Chapter 5. Finally, conclusions and recommendations for future work are presented in Chapter 6.

Chapter 2

Production Assurance & Reliability Modelling Tools

In the following chapter, the concept of production assurance and production availability are presented along with the differentiation of such terminology from others. To perform a production assurance analysis, one might need a reliability modelling tool to calculate the production availability of a system or process. Basic knowledge of three different modelling tools, namely Markov process, discrete event simulation in Python, and MIRIAM RAM Studio are then reviewed. Finally, a literature review will be presented in the last section of this chapter.

2.1 Theoretical Background

2.1.1 Production Assurance

Production assurance is a term used in production assurance programs (PAP) and production assurance analyses in the oil and gas industry for many decades. It is a systematic evaluation and calculation that is carried out to assess the performance of a system. According to the literature, many terminologies refer to production assurance, e.g., production regularity, production availability, throughput availability, and deliverability, yet they are interchangeably used (Aven, 1987). The concept of "regularity" is first defined in standard NORSOK-Z016 (1998) as follows:

> " A term used to describe how a system is capable of meeting demand for deliveries or performance. Production availability, deliverability or other appropriate measures can be used to express regularity "

Based on NORSOK-Z016 (1998) standard, a new standard, i.e., ISO 20815 standard: "Petroleum, petrochemical and natural gas industries – Production assurance and reliability management", was issued in 2008. The standard was last updated in October 2018. In ISO-20815 (2018), the

term "production assurance" is used to replace "regularity". Production assurance is defined as the follows:

> "Activities implemented to achieve and maintain a performance that is at its optimum in terms of the overall economy and at the same time consistent with applicable framework conditions.
>
> Note 1: Production assurance is not only limited to cover production of oil and gas, but can also be other activities such as drilling operations, downhole well intervention, subsea intervention, offshore loading operations, for which production assurance activities and reliability management are needed.
>
> Note 2: Production assurance activities relate closely to the integrity management of the installations. "

Also, ISO-20815 (2018) differentiates production assurance from production availability. Production availability denoted by A is the ratio of the mean actual production to the planned production, $A_{planned}$, within a specified period of time from t_1 to t_2 (ISO-20815, 2018; NORSOK-Z016, 1998). Consider a production plant system where the production rate at time t is denoted by $D(t)$, which can be the number of items/products produced per time unit at time t. The planned production rate for such system is denoted by $D_0(t)$. Production availability from time t_1 to time t_2, $A(t_1, t_2)$, is expressed mathematically as:

$$A(t_1, t_2) = \frac{\textit{Mean actual production in } (t_1,\ t_2)}{\textit{Planned production in } (t_1,\ t_2)} = \frac{\int_{t_1}^{t_2} E\big(D(T)\big)\,dt}{\int_{t_1}^{t_2} D_0(T)\,dt} \tag{2.1}$$

From Equation 2.1, one can notice that production availability is a volume-base performance measure. Hence, production availability differs from availability which is a time-based measure (ISO-20815, 2018). When the product has a high cost of deferred production, it is crucial to achieve high production availability such that Life Cycle Cost (LCC) is reduced (Kawauchi and Rausand, 2002). Hereafter, production availability can be used to express production assurance which is governed by a production assurance program.

Production availability is closely related to deliverability which is defined in NORSOK-Z016 (1998) as follows:

> "The ratio of deliveries to planned deliveries over a specified period of time, when the effect of compensating elements such as substitution from other producers and downstream buffer storage is included. "

Accordingly, one can notice the difference between production availability and deliverability is that the former concerns all the products that can be produced while the latter only takes deliv-

erable products into account. All concepts mentioned above are summarized and discussed in Barabady et al. (2010).

Quantitative analysis in general and production availability in particular is a big interest of RAMS engineers with a purpose of designing, controlling, and optimizing system performance based on well-defined criteria or standards. As a rule of thumb, a model that simply duplicates the behaviour of the system itself is usually employed to access the production availability. A general definition of a "model" is introduced by Cassandras and Lafortune (2010) as the followings. A model is a mean of mathematical tool that links input and output variables of a real system. That is:

$$y = g(u) = \Big[g_1\big(u_1(t), u_2(t), ..., u_p(t)\big), ..., g_m\big(u_1(t), u_2(t), ..., u_p(t)\big)\Big]^T \tag{2.2}$$

where

$$u(t) = \big[u_1(t), ..., u_p(t)\big]^T \tag{2.3}$$

and

$$y(t) = \big[y_1(t), ..., y_m(t)\big]^T \tag{2.4}$$

are input variables and output variables, respectively; g denotes the mathematical relationship between input and output; t is the time of interest; T means the transpose of a vector (Cassandras and Lafortune, 2010). It is worth pointing out that the model is just only the approximation of true behaviour of system in real life. Nevertheless, the terms "system" and "model" are used interchangeably given that the "model" is good enough (Cassandras and Lafortune, 2010).

Aven and Pedersen (2014) propose a framework to link the desired output of a production line and the stochastic variation of uptimes and downtimes in order to understand the uncertainty distribution for the output production availability. More specifically, a model g governed by the characteristic X of the system, i.e., the lifetimes and restoration times of specific components, is introduced in Aven and Pedersen (2014). Such model defines the production Y_g of the system as:

$$Y_g = g(X) \tag{2.5}$$

Given that X has a probability distribution captured by H_X, then the probability distribution of Y_g is established in Aven and Pedersen (2014) by:

$$\begin{aligned} P(Y_g \leq y) &= \int P\big(g(X) \leq y | X = x\big) dHX(x) \\ &= \int P\big(g(x) \leq y\big) dHX(x) \end{aligned} \tag{2.6}$$

In practice, Monte Carlo simulation is used to compute this probability distribution by running the simulations with specific values x of X and calculating the fractions of simulations that have

production $g(x)$ not larger than y (Aven and Pedersen, 2014).

Noticing that Equation 2.6 is a differential equation that is usually seen in continuous-state systems/models, it is advised to put some effort into highlighting the concept of continuous-state model and discrete-state model. Cassandras and Lafortune (2010) clarify that continuous-state models have the state space of a continuum containing all n-dimensional of real numbers, while in discrete-state models the state space is a discrete set. Similar clarification is given to continuous-time systems and discrete-time systems, which can be found in Cassandras and Lafortune (2010). Another important pair of terminology goes to time-driven and event-driven systems. The former refers to systems, or more specifically, continuous-state systems whose state changes continuously as time changes, hence input, output and state of the systems will be expressed through the time variable (Cassandras and Lafortune, 2010). The latter means that system state changes only at certain points in time triggered by an event/transaction, meaning that the state remains unaffected between two consecutive events. In time-driven systems, state transitions are synchronised by the clock, whereas, in event-driven systems, state transitions are defined by a combination of asynchronous and concurrent event processes (Cassandras and Lafortune, 2010). If a system has a discrete state space and its state transition mechanism is event-driven, then the system is called a discrete event system.

2.1.2 Reliability Modelling Tools

Markov Process

A Markov chain or Markov process is a stochastic process describing a sequence of stochastic events in which the process satisfies the Markov/Markovian property. Some usually differentiate between discrete-time stochastic process and continuous-time stochastic process. The main difference between these two is that events in the former occur at discrete time steps. Markov chains, hence, refer to discrete-time stochastic process, while Markov processes imply the continuous-time case (Medhi, 2003). Yet, there is no definitive agreement in the literature on the use of these terms. One might use the term continuous-time Markov chain (CTMC) to refer to Markov processes. Another way to classify systems is based on the nature of the state space, i.e., continuous-state and discrete-state. This study focuses on discrete-state Markov process; more about Markov chain can be found in Ross (2019a). The Markov property mentioned above is commonly known as a memoryless property, which means that the future behaviour of the process depends only on the present moment t, not on the events attained in the past up until the present moment t (Limnios and Oprişan, 2001; Rausand and Høyland, 2004). Markov process is used as statistical model of many real-world processes (Gagniuc, 2017). In RAMS engineering, it is especially useful for the investigation of systems that have several states whose transition rates are failure and repair rates.

The procedure when employing Markov model contains the following steps:

- Define system states and transition rates between them. System states are preferably illustrated diagrammatically in a state transition diagram (IEC-61165, 2006)
- Identify states that cause a system failure
- Establish a transition rate matrix and find a resolution of the Markov model by using suitable software tools
- Analyze the results

These key steps are illustrated by the following example. Consider a stochastic process $\{X(t), t \geq 0\}$ where $X(t)$ denotes the state of the system at time t. Then $S = \{0, 1, 2, ..., r\}$ is called a state space which contains all the possible states of the system. The stochastic process $\{X(t), t \geq 0\}$ turns into a homogeneous continuous-time Markov chain (CTMC) with the assumption that the sojourn time in each state is exponentially distributed. The Markov (memoryless) property is then expressed as:

$$P\big[X(t+s) = j | X(s) = i, X(u) = x_u, ..., X(0) = x_0\big] = P\big[X(t+s) = j | X(s) = i\big] \tag{2.7}$$

for any $t, s \geq 0$ and $0 \leq u < s$. Equation 2.7 shows that the conditional distribution of the future $X(t+s)$ given the present $X(s)$ and the past $X(u)$, $0 \leq u < s$, depends only on the present and is independent of the past. If this equation is independent of s, the Markov process is said to be homogeneous. If we denote the sojourn time in state i (the time the system stays in state i before making a transition) by τ_i, the Markov property follows that:

$$P[\tau_i > t + s | \tau_i > s] = P[\tau_i > t] \tag{2.8}$$

Again, this means that sojourn time in each state is memoryless and therefore must be exponentially distributed (Ross, 2019b).

Next, we denote the transition rate from state i to state j by a_{ij}. The collection of all transition rates is then arranged in a matrix called the transition matrix, denoted by A:

$$\mathrm{A} = \begin{bmatrix} a_{00} & a_{01} & \dots & a_{0r} \\ a_{10} & a_{11} & \dots & a_{1r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r0} & a_{r1} & \dots & a_{rr} \end{bmatrix} \tag{2.9}$$

In the transition rate matrix, the entries of row i, a_{ij}, $i \neq j$, denote the transition rates out of state i, commonly referred to as the departure rates. Similarly, the entries in column i, a_{ij}, $i \neq j$ denote the transition rates into state i.

Let $P_{ij}(t)$ denote the time-dependent (conditional) transition probability that the system at state j at time t when we know the process starts in state i at time $t = 0$. Mathematically,

$$P_{ij}(t) = P\big[X(t) = j | X(t = 0) = i\big] \tag{2.10}$$

As a rule of thumb, the initial state i is usually omitted in the notation, hence, we have $P_j(t)$. To find $P_j(t)$, we need to solve the (forward) Kolmogorov differential equation (Ross, 2019b):

$$P(t) \cdot A = \dot{P}(t) \tag{2.11}$$

where $P(t) = \big[P_1(t), P_2(t), ..., P_r(t)\big]$ and $\dot{P}(t) = \big[\dot{P}_1(t), \dot{P}_2(t), ..., \dot{P}_r(t)\big]$

Equation 2.11 can be rewritten as:

$$\dot{P}(t) = \lim_{\Delta t \to 0} \frac{P(t + \Delta t) - P(t)}{\Delta t} = P(t) \cdot A \tag{2.12}$$

From there, a time-dependent solution for the Markov process can be found by iteratively solving the following equation for a small time interval Δt (See Appendix C.1.1):

$$P(t + \Delta t) \approx P(t)(A\Delta t + I) \tag{2.13}$$

where I is the identity matrix, and the initial state probabilities are known.

One might want to calculate the steady-state (stationary state) probabilities, which are the value of $P_j(t)$ when $t \to \infty$ (IEC-61165, 2006). In practice, it is impossible to run a system until infinite time. Hence we examine a system's performance in a "long run" which is generally understood as a sufficiently long period of time allowing all state probabilities to reach some stable, fixed values (Cassandras and Lafortune, 2010). The steady-state probabilities, $P_0, P_1, ..., P_r$, are calculated by solving the following equations:

$$[P_0, P_1, \ldots, P_r] \cdot \begin{bmatrix} a_{00} & a_{01} & \ldots & a_{0r} \\ a_{10} & a_{11} & \ldots & a_{1r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r0} & a_{r1} & \ldots & a_{rr} \end{bmatrix} = [0, 0, \ldots, 0] \tag{2.14}$$

and

$$P_0 + P_1 + \cdots + P_r = 1 \tag{2.15}$$

Hence, outputs of a Markov model are the probability of being in given system states. These probabilities can be used as measure of availability performance which is useful in reliability/ availability prediction.

Semi-Markov Process

Markov models can not handle problems that are associated with non-exponentially distributed event time, which restricts its application in many real-time situations (Lisnianski and Levitin, 2003). An aging process, for example, has an increased degradation rate, and a Weibull distribution for the time to failure is usually used instead of exponential distribution. Semi-Markov processes (SMPs) are then introduced by Levy (1954) and Smith (1955) to handle non-Markovian models where the time spent in a given state is allowed to follow non-exponential (general) distribution (Pyke, 1961; Feller, 1964; Korolyuk et al., 1975; Ross, 1996). In others words, if the exponential distribution of holding times is satisfied, and if the waiting time in a state and the next state reached are independent, the Semi-Markov process becomes a CTMC (Ross, 1996; Limnios and Barbu, 2009; Trivedi et al., 2015).

A Semi-Markov process is constructed by the so called Markov renewal process (MRP). Let recall the Markov process $\{X(t), t \geq 0\}$ with discrete countable state space $S = \{0, 1, 2, ..., r\}$, and let $t_0 = 0, t_1, t_2, ...$ $(t_i < t_{i+1})$ be the jump times at which transitions occur. Accordingly, the sequence $\{X_n = X(t_n), n \geq 0\}$ establishes a Markov chain, and the sojourn times (inter-arrival/inter-jump times), $T_n = t_n - t_{n-1}, n = 1, 2, ...$ are independent and exponentially distributed with means that may depend on the state of X_n. If these transition intervals $T_n, n = 1, 2, ...$ have an independent arbitrary distribution, and that the mean may depend not only on the state of X_n but also on the state of X_{n+1}, the two-dimension process $\{X_n, t_n, n \geq 0\}$ is then called Markov renewal process with state space S (Pyke, 1961; Çinlar, 1969; Medhi, 2003). That is, with $i, j \in S$, we have:

$$\begin{aligned} &P\{X_{n+1} = j, T_{n+1} \leq t | (X_0, T_0), (X_1, T_1), ..., (X_n = i, T_n)\} \\ &= P\{X_{n+1} = j, T_{n+1} \leq t | X_n = i\} \end{aligned} \tag{2.16}$$

A Semi-Markov process is then defined as a continuous-time process:

$$Y(t) = X_n, t \in [t_n, t_{n+1}) \tag{2.17}$$

Equation 2.17 shows that $Y(t)$ returns the state of the process at its most recent transition. The Markov chain $\{X_n, n \geq 0\}$ is called the embedded Markov chain of the Semi-Markov process $Y(t)$.

The difference between Markov process, Markov renewal process, and Semi-Markov process is subtle. The entire Semi-Markov process is not Markovian (memoryless). That is, the time spent in states between transitions does not necessarily possess the memoryless property. This gives Semi-Markov processes an advantage compared to the Markov processes, which is the ability to take into account all distributions on the positive real axis. Semi-Markov process possesses the Markov property only at the specified jump times (Medhi, 2003). Therefore, Semi-Markov processes allow the probability of transition at a given time from a state to be dependent on the time spent in the state. In general, Semi-Markov processes might be considered as a gen-

eralisation of the Markov process (Grabski, 2016). Compared to the Markov renewal process, Semi-Markov process is equivalent except that a state in Semi-Markov process is defined for every given time, not just at the jump times.

An equation to solve the state probabilities of the Semi-Markov processes is proposed by Nunn and Desiderio (1977). Let $K_{ij}(t)$ be the probability of a transition from state i (at time zero) to state j between t and $t+dt$. That is

$$K_{ij}(t) = P\{X_{n+1} = j, t_{n+1} - t_n \leq t | X_n = i\} \tag{2.18}$$

with $i, j \in S$ and $t \geq 0$. The value $K_{ij}(t)$ satisfies:

$$\int_0^\infty dt \Sigma_{j=1}^N K_{ij}(t) = 1 \tag{2.19}$$

where N is the number of states and $1 \leq i \leq N$. The value $K_{ij}(t)$ is called the Semi-Markov kernel. A matrix $K(t)$ whose entries are Semi-Markov kernel is called kernel matrix:

$$K(t) = [K_{ij}(t)] \tag{2.20}$$

Let $F_i(t)$ be the cumulative density function (cdf) of the sojourn time (holding time) in the state i (Grabski, 2016), then we have:

$$K_{ij}(t) = P_{ij}(t) F_i(t) \tag{2.21}$$

Recall $P_{ij}(t)$, the conditional transition probabilities that the system is in state j at time t, given that it starts in state i at time $t = 0$. According to Nunn and Desiderio (1977), $P_{ij}(t)$ can be computed by solving the Markov renewal equation:

$$P_{ij}(t) = \delta_{ij}\left(1 - \int_0^t F_i(t)dt\right) + \Sigma_k \int_0^t K_{ik}(\tau) d\tau P_{kj}(t-\tau) \tag{2.22}$$

where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise. Notice that the quantity $K_{ik}(\tau)$ is the probability that the system makes a transition to state k between times τ and $\tau + d\tau$, and $P_{kj}(t-\tau)$ is the probability that the system ends up at state j in the remaining time $t-\tau$ after entering state k. If we define matrix $E(t)$ by

$$E(t) = [E_{ij}(t)] = \left[\delta_{ij}\left(1 - \int_0^t F_i(t)dt\right)\right], \tag{2.23}$$

Equation 2.22 can be expressed in matrix form as

$$P(t) = E(t) + \int_0^t dK(\tau) P(t-\tau) \tag{2.24}$$

Equation 2.22 has the form of a linear Volterra equation of the second kind (Fröberg, 1972) and in general is hard to solve in time domain. Nevertheless, it can be solved by means of Laplace-Stieltjes domain method (Nunn and Desiderio, 1977; Yin et al., 2002). That is,

$$P^{\sim}(s) = [I - K^{\sim}(s)]^{-1} E^{\sim}(s) \tag{2.25}$$

where $E^{\sim}(s) = \int_0^{\infty} e^{-st} dE(t)$ and $K^{\sim}(s) = \int_0^{\infty} e^{-st} dK(t)$ (Fricks et al., 2001). Then, $P(t)$ is calculated by taking the inverse Laplace-Stieltjes transform of $P^{\sim}(s)$. However, the transform technique is not a trivial task, especially when the system has a large number of states and density functions are complex. In this case, numerical solution of Equation 2.22 is obtained (Nunn and Desiderio, 1977). In addition, empirical estimators of the stationary distribution for semi-Markov processes are also worked out, see Limnios et al. (2005).

In regards to non-exponential distributions, k-stage Erlang distribution can be used in Semi-Markov processes to deal with deterministic distribution such as deterministic repair time. Let $E_k(\lambda)$ denote a k-stage Erlang distribution with a shape parameter, k, and a rate parameter, λ. The Erlang distribution is used to find a probability that the k^{th} event occurs at t time by using the following equation:

$$P(X = t) = \frac{\lambda^k t^{k-1} e^{-\lambda t}}{(k-1)!} \quad \text{for t,} \lambda \geq 0 \tag{2.26}$$

Erlang distribution has a close relation with Gamma and exponential distributions. It is a special form of Gamma distribution wherein the shape parameter k is restricted to a positive integer. Erlang distribution can be seen as a distribution of the sum of k independent and identically distributed exponential random variables (Yin et al., 2002; Jin et al., 2004). That is, if we have

$$X_i \sim \text{Exponential}(\lambda), \tag{2.27}$$

then

$$\sum_{i=1}^{k} X_i \sim E_k(\lambda) \tag{2.28}$$

Hence, Erlang distribution is proved to have the Markov property (Khaleghei and Makis, 2015). In short, Erlang distribution is used to model continuous variables by using a number of discrete states in a continuous-time Markov chain. By doing that, Semi-Markov processes can be treated as Markov processes.

Discrete Event Simulation (DES) in Python with SimPy

According to Cassandras and Lafortune (2010), discrete event simulation is defined as a process that numerically models and evaluates a discrete-event system. The general concept of discrete event simulation is explained by these authors as the followings. Let ϵ and X denote the event

set and state space of a discrete-event system. At any state x, there are so-called feasible events that may occur at this state. Each feasible event, e.g., event i, has a clock value, y_i, which is the amount of time required until such event occurs. Now assume at current time t with the current state, x, the event with the smallest clock value at the state x is then called the triggering event, denoted by e'. Let $\Gamma(x)$ be the set of feasible events of state x, $\Gamma(x) \subseteq \epsilon$, e' is expressed as

$$e' = arg \min_{i \in \Gamma(x)} \{y_i\} \tag{2.29}$$

Interevent time is then defined as the amount of time spend at state x, that is

$$y^* = \min_{i \in \Gamma(x)} \{y_i\} \tag{2.30}$$

Next, we move to the next state x' and simultaneously update the time and clock values for all feasible events in the new state x' by setting:

$$t' = t + y^* \tag{2.31}$$

$$y'_i = y_i - y^* \tag{2.32}$$

When the triggering event e' is activated, i.e., at time t', a next occurrence of this event is scheduled at time $(t' + r)$, where r is a lifetime sample supplied by the computer. This creates a so-called a scheduled event list (SEL) which is always ordered on a smallest-scheduled-time-first basis. In short, the simulation procedure is a continuous repetition of the following steps (with initialization of state x_0 and time $t = 0$):

1. Remove the triggering event (e', t') from the scheduled event list (note that t' is its occurrence time)
2. Update simulation time to t' and state to x'
3. Delete from the scheduled event list any entries that are not feasible events in the new state
4. Add to the scheduled event list any feasible event which is not already scheduled. The scheduled event time for an added event is given by $(t' + r)$, where r is a lifetime obtained from the random variate generator
5. Reorder the updated scheduled event list based on a smallest-scheduled- time-first scheme

The simulation procedure above is simplified when a library named SimPy is used in Python. SimPy is a process-based discrete-event simulation framework implemented in Python (SimPy, 2013). In SimPy, the behaviour of active objects such as customers or vehicles is modelled by

"processes". These processes are governed by an "environment" which keeps track of the current simulation time and allows the processes to interact with the environment or with each other via events. When a process function yields an event, SimPy adds the process to the event list in chronological order with respect to the time of execution and suspends the process until the event is triggered. This is basically similar to posting an event to a pending event set (PES) or scheduled event list (SEL).

A critical event type in SimPy is the "timeout" event which allows the process to hold its state until a certain amount of simulated time has passed (SimPy, 2013). Simply put, a "timeout" event passes the environment a delay. By using this feature, one can generate events and schedule them at a given simulation time. Adapting this concept into the reliability modelling area, "timeout" events are used to simulate the Time-To-Failure (TTF) and Time-To-Repair (TTR).

In SimPy, various types of shared resources with limited capacity, e.g., tanks, servers, and checkout counters, can be included and modelled in the simulation. Two commonly used resources in SimPy are:

1. Resources: Resources that can be used by a limited number of processes at a time, e.g., a store with two counters

2. Containers: Resources that model the production and consumption of a homogeneous, undifferentiated bulk. It could either be continuous (like gas) or discrete (like apples)

The main principle when working with resources in SimPy is that processes request these resources to become an "owner" and have to release the resources once they are done using them. This is performed by using the *request()* and *release()* functions. Note that releasing a resource will always succeed immediately. In terms of containers, SimPy enables processes to either put something into the containers or get something out by the *put()* method and *get()* method, respectively. Both functions return an event that is triggered when the corresponding condition is satisfied. More specifically, when the container is full or empty, the processes have to queue up and wait.

Simulation time and how many replications to run can be defined by programmers in Python. By default, a simulation will run as long as there are events in the event list. However, one can terminate the simulation at a specific time by providing an argument to the *run()* function. With respect to the appropriate number of replications (N) to run to achieve an accurate output, Hoad et al. (2007) mentions three main methods in the literature for choosing this number. These methods are (1) rule of thumb by Law and McComas (1991), (2) graphical method by Robinson (2004), and (3) confidence interval (with specified precision) method (Robinson, 2004; Banks et al., 2005). The rule of thumb suggests to run 3 to 5 replications. With the graphical method, programmers plot the cumulative mean of the target variable against the number of replications and find the "flat" point on the graph. The confidence interval method runs in-

creasing numbers of replications until the confidence intervals are achieved. In this study, we use the graphical method as a means in making decision about the number of replications in both Python and MIRIAM RAM Studio.

MIRIAM RAM Studio

MIRIAM RAM Studio is a cloud-based production performance simulation tool for the oil & gas and process industries. MIRIAM RAM Studio allows users to perform a wide range of studies, from conceptual design to operations and maintenance planning, thanks to its flexibility of defining flow networks representing the systems or components of interest. Some typical applications of MIRIAM are production availability calculation, alternatives comparison for increased productivity, cost and downtime estimation, identification of contribution of individual item in the loss of production.

Drawing a flow network is the first thing to do to model a system in MIRIAM RAM Studio. The flow network in MIRIAM is built by three main network elements: boundary points, process stages containing items, and storage units. All networks must contain at least one entry boundary point, one process stage, and one discharge boundary point (See Figure 2.1 for a simple model). The network elements are represented by specific blocks connected by arrows (arcs, links). Each process stage and storage unit must have at least one downstream link and at least one upstream link. Entry points must have at least one downstream link while discharge points must have at least one upstream link. Description of these elements is the followings:

- Entry and discharge boundaries: These define points at which flow enters and leaves the system. There are three possibilities for modeling a flow, namely Constant, Calendar, and Stochastic. By using Calendar and/or Seasonal distribution, it is possible to define flow rates/capacities that changes throughout the simulation period, allowing specific situation to be modeled as accurately as possible. Especially, discharge boundary differentiates between min flow and demand flow. While the former makes sure that the system will produce at least either such min flow or null, the latter defines the required flow and will be the reference level of the production availability. In case there are many discharge points in a system, a prioritized list can be defined to maximise flow at certain discharge points

- Process stages: They contain items/components in a system. Users have freedom to model one specific item (a valve, pump, or pipeline) or a subsystem with several items in parallel or series within a process stage. Redundancy/ k-out-of-n configurations are easily modelled in process stage by properties of the parallel streams: (1) number of streams in parallel, (2) number of streams normally running, and (3) number of streams required to run. The two last properties will define how many streams are running or idle. Other ba-

sic properties in process stages include throughputs, transformation, and flow capacity, which can vary during the lifetime of the system thanks to calendar and seasonal profiles. Components in process stages are defined by data including failure and repair distribution, common cause factor, maintenance strategies and schedules (repair resources and spares, mobilization time for the resources), life cycle cost, and emissions. In terms of input data, different time units can be used and mixed, such as *Y* for year, *H* for hour, *M* for month and *D* for days, *W* for week

- Storage units: They are usually used as storage tanks which compensate for lack of input or production during downtime. The flow through the unit is only bounded by the maximum fill rate and the maximum withdrawal rate

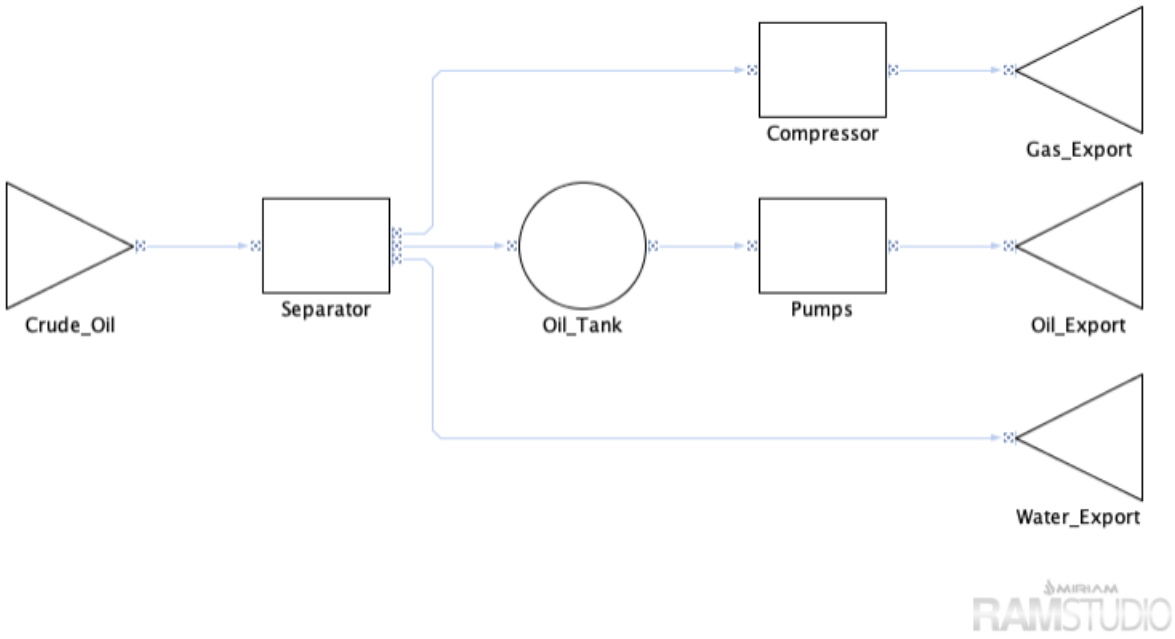


Figure 2.1: Simple flow network in MIRIAM RAM Studio.

Methodology used in MIRIAM is a combination of flow algorithm and next event Monte Carlo Simulation method. In particular, Linear Programming (LP) algorithm is the flow algorithm that is utilized in MIRIAM (Bazaraa et al., 2004). MIRIAM RAM Studio converts the flow network with predefined constraints into linear equations which are then solved to optimise the flow at every points of the network. The default objective is to meet the demand at the discharge points. Thus, demand flow will be part of a so-called cost function. The Linear Programming will balance the flows, throughputs, and streams such that it maximises the demand flow in the cost function. In fact, one flow network can have multiple demand flows, users can change the

priority of fulfilling the demand flows via objectives panel. The objective higher on the list is prioritized over the one lower on the list, and vice verse. Such procedure is also applied to entry points, process stages, and storage units.

MIRIAM provides various ways to validate and simulate models. The simulation will not start until the validation has been performed successfully. Two features "Warnings" and "Errors" in MIRIAM automatically check for inconsistencies in the network and input errors. Warnings are informative issues, meaning that the validation system has detected situations that are somehow unusual but perfectly valid. Whereas, Errors are critical issues. The simulation can not be started with Errors in the model. Besides that, users can test out models by activating flow simulation. Here one can force certain streams in process stage to be failed and test the simulation in order to validate the model. In terms of simulation setting, users can specify run length, number of replications, playback option, and so forth in simulation scenario. If playback option is chosen, after running simulations, MIRIAM returns a complete list of simulated events and the corresponding flow chart, which helps to easily understand the model. The playback property of the simulation uses different colours in the flow chart to show the status of an element. For example, the green color at discharge points means that actual flow equals demand flow. If actual flow is smaller than demand flow, the color is purple. After running simulations, MIRIAM RAM Studio creates a series of reports about the system's performance. These reports show production performance, blaming contributors to unavailability, downtime statistics, resource utilization, and seasonal & annual reporting. All input parameter and reports can be exported to Microsoft Excel for further analysis.

Advantages of MIRIAM RAM Studio are several. MIRIAM has an ability to handle multiple flows and record production availability results for several boundary points. In MIRIAM RAM Studio, flow is the central object of modeling. Hence there are many opportunities to specify a flow in MIRIAM. Multiple flows can go in/ out of the system. In process stages, flows can be split according to chemical or physical processes. For example, a separator modelled in a process stage can separate oil and gas into different flows. Models in MIRIAM can deal with seasonal capacity changes of the network during simulation through modeling the production profile (Wang, 2012). MIRIAM also features advanced blaming analysis to compute how much an individual item is responsible for loss of production. The principle is that when production/ throughput is less than the demand value at a discharge point, the items that are currently failed are blamed for the production loss. Moreover, MIRIAM RAM Studio is a user friendly collaboration platform, allowing project team members work on the same models, share, and modify them regardless of locations. Additionally, MIRIAM is useful in modeling external auxilary systems, e.g., electric system, water supplier, which items in the flow network depend on (Vesteraas, 2008).

2.2 Literature Review

In this section, a literature review is performed by using two search engines, i.e., *webofscience.com* and *engineeringvillage.com*. Figure 2.2 shows the number of patents on production assurance in the world between 1978 and 2022. The data were obtained using the keywords "production assurance" and "production availability". It is shown in Figure 2.2 that the interest in production assurance has accumulated in the last 40 years. Since 1978, there have been 460 papers associated with the keywords "production assurance" and "production availability". What can be clearly seen in Figure 2.2 is a considerable amount of literature that has been published on production availability in the last two decades. On average, more than 20 papers have been published each year since 2005. Figure 2.3 shows the papers' citation topics. Among all the fields, Safety & Maintenance, our main area of interest, accounts for approximately 24% of the total number of these papers. The literature review is conducted based on the papers falling into this Safety & Maintenance category.

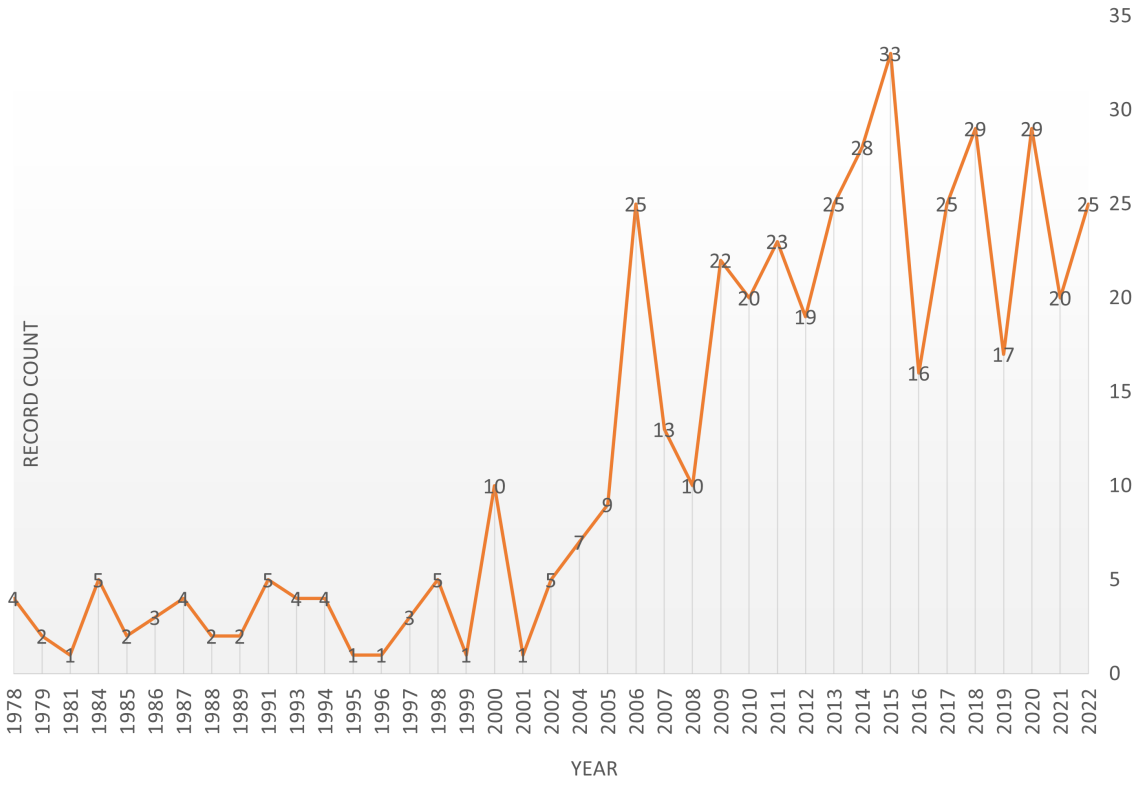


Figure 2.2: Number of papers addressing topics related to "production assurance" and "production availability".

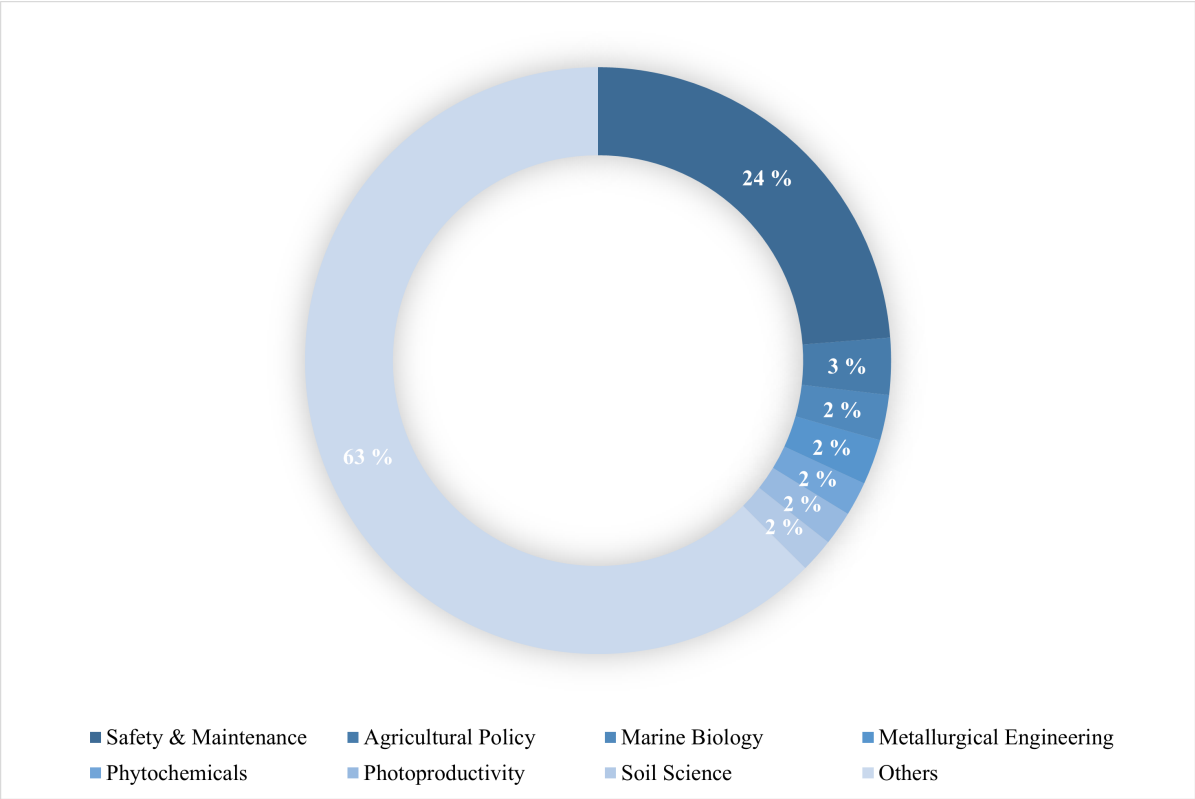


Figure 2.3: Field of study of 460 papers.

2.2.1 Reliability Modelling Tools Comparison

There are two main approaches to accessing production availability, namely the analytical methods and simulation-based (simulation) methods (Kawauchi and Rausand, 2002; Barabady et al., 2010). The former method could be referred as conventional method including reliability block diagram (RBD), fault tree analysis (FTA), Markov modelling, Petri net, etc. They are used to inductively calculate availability and production availability of a system based on predefined formulae from theoretical models for system availability (Barabady et al., 2010). Simulation methods predict production availability of a system by simulating its behaviour over a period of time using reliability data and specific operational rules. Most of the tools in this category use Monte Carlo method to carry out the simulations. Analytical tools are usually limited in term of size and complexity of systems. Compared to analytical methods, simulation methods are flexible but they require more effort, time, and cost to be performed (Barabady et al., 2010). One might want to compare different tools from the two modelling approaches. However, there is a relatively small body of literature that is concerned with systematic comparison of reliability modelling tools.

The first attempt to compare simulation and analytic approach to reliability modelling might be made by Hokstad (1988). Two analytical techniques described in Hokstad (1988) are asymp-

totic calculations and Markov analysis. The issue associated with asymptotic method is that it is less flexible for modelling maintenance strategies. Markov analysis can handle certain maintenance strategies in small and medium size systems but it is limited to ones where all relevant time periods are exponentially distributed (Hokstad, 1988).

Within analytical methods for reliability, availability, maintainability and safety analysis, a general comparison can be found in IEC 60300-3-1 (IEC, 2003). This standard provides an overview of methodologies, advantages and disadvantages, data input and other conditions for using various techniques from the analytical side. It is therefore a good resource with necessary information needed for choosing the most appropriate analysis methods.

Boiteau et al. (2006) compares Markov analysis with Monte-Carlo simulation in computing production availability of a simple system from an oil extraction installation. The findings from this research imply that the Markov analysis is more accurate and less computationally expensive than the stochastic simulation. However, Markov analysis suffers from the exponential blow-up of the total number of system states, resulting in a huge Markov graph (Boiteau et al., 2006). Stochastic simulation can deal with multi-state systems but it is time-consuming. Briš (2013) combines a simulation approach with stochastic Petri nets to perform the availability assessment of a multi-state, multi-output offshore installation. The results of this study are compared with the results published in Zio et al. (2006) where minimal cut sets and the Monte Carlo simulation approach are employed to evaluate the availability of the same system. Either Petri nets approach or the method based on minimal cut sets and Monte Carlo simulation gives comparable results (Briš, 2013). In general, Monte Carlo simulation provides the flexibility to include realistic aspects of system behaviour, e.g., corrective and preventive maintenance with stochastic or deterministic durations, repair resources, and repair priorities, which the analytical techniques have failed to do (Zio et al., 2006; Briš, 2013).

Since there are not so many articles focusing on the comparison of modelling tools in open literature, the literature review is expanded towards Master's thesis at NTNU Open. When it comes to the comparison of commercial software, MIRIAM Regina and Relex Reliability Studio are reviewed by Vesteraas (2008). It is stated that these software are different and not straightforward to compare (Vesteraas, 2008). Relex Reliability Studio focuses on system failure and success, making it possible to compute availability by both simulation and analytical methods. The analytical method in Relex works well to simple systems only by measuring availability through reliability block diagrams (RBD), fault trees, and FMECAs. Whereas, MIRIAM Regina focuses on throughput of systems and computes production availability by combining a sophisticated flow algorithm with Monte Carlo simulation. Vesteraas (2008) mentions that the flow algorithm in MIRIAM is advanced and easy to understand than Relex. Relex, however, can perform optimization of maintenance intervals based on cost or system downtime, which is not possible in MIRIAM. Additionally, statistics results in Relex such as MTBF, MTTF, availability, and reli-

ability have confidence intervals. Whereas MIRIAM returns the production availability with a standard deviation only (Vesteraas, 2008). Nevertheless, both software are good at implementing advanced maintenance strategies in the model. Besides simulation tools, Vesteraas (2008) also compares three analytical methods, i.e., the ordinary computation, renewal method, and quasi renewal method, for computing availability. Production availability calculated by these methods is not so accurate and rather complicated, not mention that it is time consuming if the system gets large (Vesteraas, 2008). Wang (2012) combines RAM analysis with life cycle cost evaluation and compares analytical and simulation approaches to RAM analysis. He discusses advantages of three commercial simulation tools, i.e., Miriam Regina, Maros and Extendsim, and obtains models by using Miriam Regina only. His conclusion is that analytical approach which is characterised by predefined formulas is rigid and weak at modelling large and complex systems, while simulation approach is more flexible and capable of getting more accurate results. Generally, these software programs enable us to take into consideration more features that reflect the system in real life. Sun (2017) discussed analytical approach and simulation tools including Maros and Taro, MIRIAM Regina and ExtendSim. However, only ExtendSim is employed in the case study of this research. It is said that the simulation approaches are more flexible than analytical approach but simulation tools are rather time and cost consuming, and require a solid mathematical and programming knowledge (Sun, 2017).

Some efforts have been made to improve the existing tools. Kawauchi and Rausand (2002) propose a new approach that divides the system under consideration into subsystems, and the production availability can be accessed by measuring the probability distribution of throughput capacity (PDC) of basic subsystems. In doing that, Markov diagrams and a simple merging rule for parallel and series subsystems are applied. The system PDC for a case study obtained by such method is compared with the one obtained by using the modelling constructs applied in the software package UNIRAM. The comparison shows a certain similarity between these results, given that there are two main differences between the proposed method in Kawauchi and Rausand (2002) and the UNIRAM method: (1) in the former, Markov modelling is used to model the components, while Fault Tree Analysis is used in the latter; (2) different ways to combine two subsystems (Kawauchi and Rausand, 2002). The new approach is quite flexible giving the possibility of modelling different maintenance strategies and systems of dependent components (Kawauchi and Rausand, 2002). Using the same case study as in Kawauchi and Rausand (2002), Kloul and Rauzy (2017) test a new modelling methodology called Production Trees for production availability assessment. This new approach is implemented as a library in AltaRica modelling patterns, providing similar results obtained analytically in Kawauchi and Rausand (2002) (Kloul and Rauzy, 2017).

2.2.2 Uncertainty In Production Availability

The source of uncertainty in the production availability results is also covered in Vesteraas (2008). One of the main sources of uncertainty comes from the assumptions that make the model deviate from the real-life system. Two other sources of uncertainty lie on the input data and the number of simulation iterations (Vesteraas, 2008).

In recent years, researchers have investigated a variety of approaches to present and understand the uncertainty of production assurance analyses. Persskog-Lundtofte et al. (2015) indicates that the uncertainty stems from the poor reliability input data, i.e., failure rates and repair rates. Such reliability data is normally not relevant for a specific component of the production system since it is accessed from one component and applied to another one that is constructed and operated differently. To better understand the uncertainty associated with such quality aspects, a practical method ranking components based on data quality is introduced in Persskog-Lundtofte et al. (2015). This method seems to be aligned with the ranking of subsystems/components according to their criticality in Wang (2012). Wang (2012) claims that uncertainties come from three areas, i.e., parameter, model, and completeness. Accordingly, using more reliable data, realistic models, and document assumptions can help to reduce uncertainties (Wang, 2012). In addition, Aven and Pedersen (2014) proposes a framework linking the variation of lifetimes and restoration times to the uncertainty distribution for the production availability calculation. As part of the insights highlighted in this paper, the assumptions made during the analysis need to be assessed with respect to sensitivity. The main reason for this is that one assumption can be more or less uncertain than the others leading to different results when modifying some of the assumptions. This is where the sensitivity analysis needs to be carried out to investigate the impact of changes in the process configuration and design (Wang, 2012).

Chapter 3

Blue Hydrogen Technology

In this chapter, an overview of blue hydrogen production processes is represented. In particular, the operation of two blue hydrogen production technologies is described. Each process includes hydrogen production units, carbon capture, CO_2 transportation, sequestration, and hydrogen storage.

3.1 Steam Methane Reforming

A Steam Methane Reforming (SMR) system consists of four main sequential units: desulfurized, reformer, and shift reactors followed by a separation unit. A schematic representation of the SMR process is shown in figure 3.1. The feed gas for the process is natural gas, which is desulfurized before entering the reforming process to avoid the production of sulfur oxides and contamination of catalysts in the reformer. Desulfurized natural gas reacts with steam within the reforming reactor at high temperature (800-900° C) and high pressure (15-50 bar) (Damen et al., 2006; Ritter and Ebner, 2007; Soltani et al., 2014; Luo et al., 2018) in the presence of catalysts, producing syngas which is a mixture of hydrogen and carbon monoxide:

$$CH_4 + H_2 \rightleftharpoons 3H_2 + CO \tag{3.1}$$

Equation 3.1 is a highly endothermic reaction (Luo et al., 2018), meaning that it requires a large amount of heat to be provided by an external source, e.g., burning natural gas or the purge gas from PSA in a furnace. This fuel requirement needed for the reforming reaction in the reactor becomes a big drawback of SMR technology (Martínez et al., 2014). The hydrogen-rich syngas is fed into the two water-gas shift (WGS) reactors (high-temperature water gas shift and low-temperature water gas shift (Martínez et al., 2014; Song et al., 2015)), increasing the hydrogen

yield through the addition of steam:

$$CO + H_2O \rightleftharpoons CO_2 + H_2 \tag{3.2}$$

The mixture of hydrogen, carbon dioxide and carbon monoxide is then transferred to a syngas purification/pressure swing adsorption (PSA) separating hydrogen from other substances and giving high-quality hydrogen (99.99% purity) (Damen et al., 2006).

Note that the process flow diagram in Figure 3.1 is simplified for the purpose of introducing the concept of blue hydrogen. Hence, this diagram is not meant to be used as a block diagram for the system. In our case study presented in the next chapter, the subsystem under study and its configuration will be examined in detail.

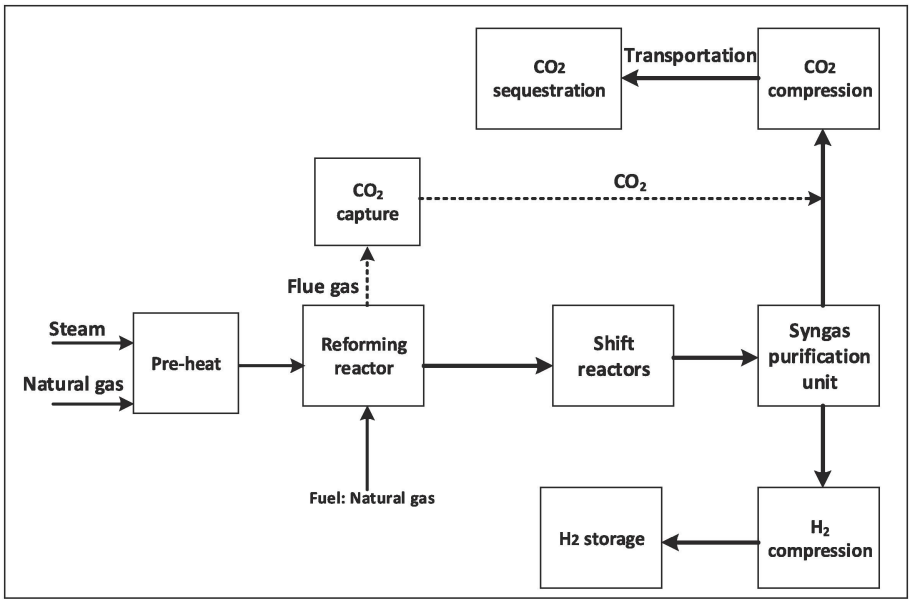


Figure 3.1: Simplified process flow diagram of steam methane reforming with carbon capture and storage (Oni et al., 2022)

3.2 Autothermal Reforming

Autothermal Reforming (ATR) process is a combination of steam methane reforming with partial oxidation. ATR process contains an autothermal reactor, water-gas shift reactors, syngas purification unit, compressors, transportation, sequenstration, hydrogen storage, and an air separation unit (Figure 3.2). First, steam and natural gas are preheated before entering the autothermal reactor. In the autothermal reactor, oxygen extracted from the air by the air separation unit is added to cause the partial oxidation reaction and reforming reaction to occur simultaneously (Damen et al., 2006). The partial oxidation reaction is exothermic, meaning that it produces

heat to drive the endothermic steam reforming reaction (Ritter and Ebner, 2007). Hence, external heat is not required in ATR systems, leading to higher energy efficiency than SMR systems. The produced syngas contains hydrogen, carbon oxide, and steam, entering the WGS to produce more hydrogen. The shifted gas is sent to the syngas purification and PSA where hydrogen is recovered at a high purity while the remaining gases (fuel gases) are used as fuel in the boiler or furnace.

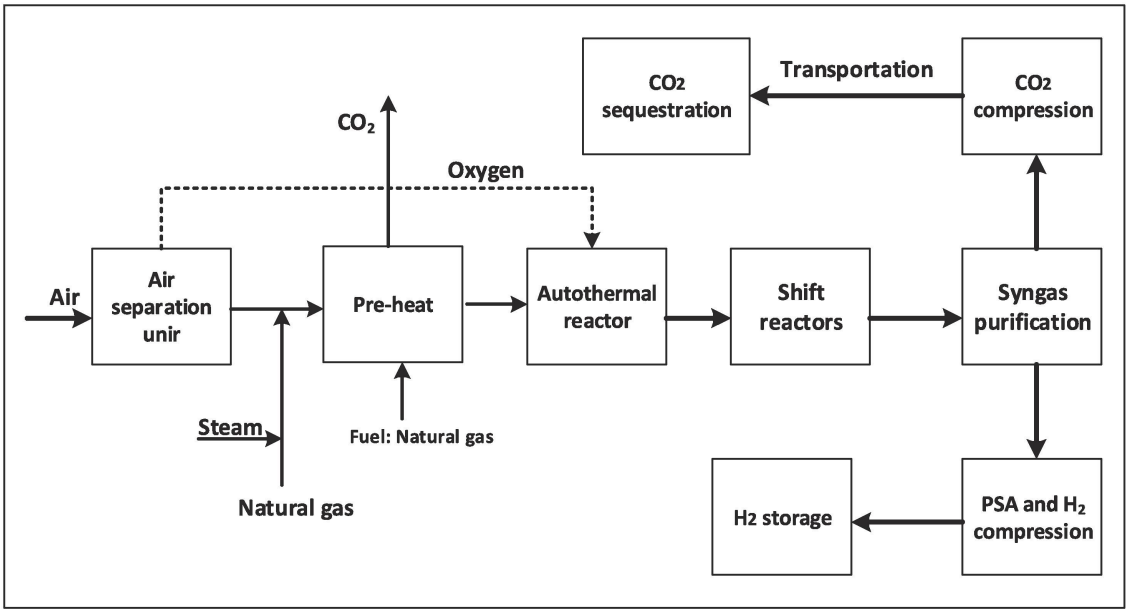


Figure 3.2: Simplified process flow diagram of autothermal reforming with carbon capture and storage (Oni et al., 2022)

3.3 Hydrogen Pressure Swing Adsorption

Four different hydrogen purification technologies in the industry are absorption, both chemical and physical, adsorption, membranes, and cryogenic processes. In terms of the adsorption method, PSA is widely used because of the ability to produce high purity of hydrogen (99.99 vol% H_2) (Stöcker et al., 1998). Adsorbents in PSA processes are silica gel/alumina for water removal, activated carbon for CO_2 removal, and 5A zeolite for CH_4, CO, and N_2 removal (Ritter and Ebner, 2007). Commercial implementation of PSA for large-scale hydrogen purification made a major breakthrough in the early 1970s with the development of a 4-bed, multi-layer PSA process (Stöcker et al., 1998). Since then, more beds have been added to the PSA system, as many as 16 beds (Luberti et al., 2014; Luberti and Ahn, 2022). This leads to the complex arrangement of flows, co- and counter-current depressurization, pressure equalization, and repressurisation. Simple sequences of operation for a 4-bed PSA system are shown in Chapter 4. The advantage of having multiple adsorbers is twofold: (1) having more than one adsorber on the adsorption

mode or regeneration modes, (2) bypassing a malfunctioning valve or instrument by switching the configuration of the system such that it can be operated with fewer adsorbers (Stöcker et al., 1998). This can help to achieve higher throughputs but make the cycle sequence more complex. Different combinations of PSA process steps are numerically investigated in Waldron (2000). Sometimes, tanks could be installed to store intermediate process streams between cycle stages, resulting in a bed reduction in PSA configurations Zhou et al. (2002). A summary of hydrogen PSA technology revolution is represented in Stöcker et al. (1998); Ritter and Ebner (2007).

In a PSA system, adsorbers are operating in a staggered sequence, making the process to be continuously producing hydrogen. Figure 3.3 shows the basic steps of all PSA units regardless of the number of adsorber vessels. A simple pressure-swing cycle consists of the following five steps:

a) **Adsorption**:

Feed gas at a high adsorption pressure (P^F) is passed through the adsorption bed in the upward direction. Impurities are adsorbed in the different layers of the bed, and a stream of high-purity hydrogen at pressure P^F is withdrawn at the product end as a product. A part of this gas is used for repressurisation of a companion column. This step ends when it has reached its adsorption capacity, and the feed is automatically switched to a fresh adsorber.

b) **Co-current depressurisation**:

The adsorption step is followed by co-current depressurization. The column is depressurized (same direction as the feed flow) from P^F to P^I to recover the hydrogen trapped in the adsorbent void spaces in the adsorber. The hydrogen is used to repressurise and purge other adsorbers.

c) **Counter-current depressurisation (dump/blowdown)**:

The column is countercurrently depressurised from P^I to P^D to remove the impurity fronts. The desorbed impurities are rejected to the offgas. This step is sometimes called "blowdown".

d) **Purge at low pressure**:

The column is countercurrently purged at pressure P^D using a stream of hydrogen from another column undergoing step b. This step further regenerates the adsorber, giving the remaining portion of the desorbed impurities at the feed end.

e) **Repressurisation**:

The column is countercurrently repressurised from P^D to P^F by introducing hydrogen product gas from other columns undergoing step a and step b. The column is now ready for a new cycle

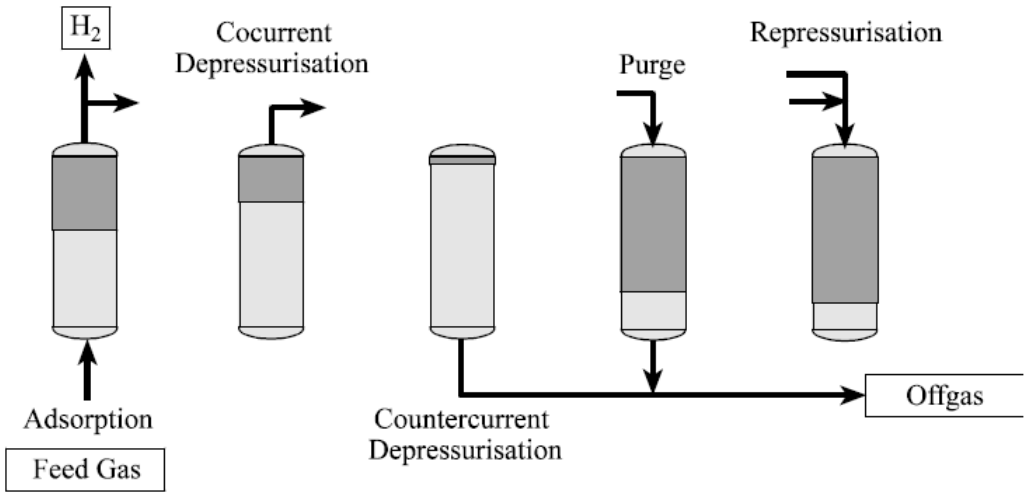


Figure 3.3: PSA process steps (Stöcker et al., 1998)

Other configurations of PSA processes can have an additional providing purge step and multiple (de)pressurising pressure equalisation steps. Providing purge is a cocurrently depressurised step, providing a hydrogen-rich stream to another adsorber undergoing the purge step. The term pressure equalization refers to the process where the pressures in two interconnected beds are equalized. (De)pressurising pressure equalisation steps help to (de)pressurise to an intermediate pressure level between the high and low-pressure (Cassidy, 1980). The idea of pressure equalisation was first introduced in William D Marsh (1961) where an empty tank was required to store a portion of the compressed gas from a saturated bed, and the gas was used to purge the same bed later.

Yavary et al. (2015) investigates three 6-bed PSA cycles with one, two and three pressure equalisation steps for purifying hydrogen from a refinery off-gas. This is claimed to reduce the initial pressure of the counter-current depressurisation step, thus minimising the hydrogen losses (Waldron, 2000; Luberti et al., 2014; Luberti and Ahn, 2022). More pressure equalisations, however, increase the cost of a PSA unit (Stöcker et al., 1998). Also, idle steps with no gas streams flowing into or out of the bed can be introduced to the system for the bed synchronization purpose (Luberti and Ahn, 2022). To investigate and optimize the proper numbers of beds and idle steps with respect to a specified step sequence, scheduling methods are usually used (Mehrotra et al., 2010).

Chapter 4

Case Study on Production Availability of a 4-bed PSA System

In what follows, we consider one of the most common PSA systems in hydrogen purification in the industry. The purpose is to model and calculate the production availability of a 4-bed PSA system. To this end, different modelling tools, namely Markov process, discrete event simulation in Python, and MIRIAM RAM Studio, are used to build models for the system. The input data and results from these models are shown in detail in this chapter.

4.1 4-bed PSA System Description

The 4-bed system was developed by Batta in 1971 (Batta, 1971). As the name implies it has four parallel adsorbers operating continuously with constant feed flow and product gas withdrawal (Figure 4.1). A 4-bed PSA system is usually used for production plants that require a small capacity of PSA units, e.g., few hundreds normal cubic metres per hour, Nm^3/h. In this study, let consider a PSA unit with a capacity of 900 $Nm^3 H_2/h$. The total cycle time for the process can be 16 minutes as reported in Stöcker et al. (1998) or 24 minutes as shown in Batta (1971). A cycle corresponding to a 16-minute program of a 4-bed PSA is chosen to be studied in this research (See Figure 4.2). Figure 4.2 illustrates that the 4 beds have a phase lag between them, and they change their functions after every 4 minutes. This makes sure that the four beds (adsorbers) of the PSA unit perform all the functional modes at any time. Hereafter, we call this property as a cycle of 4-minute functional modes. Let study these functional modes in the first four minutes of the process in Figure 4.2:

Bed A : **High-pressure adsorption (ADS)**: producing 25 Nm^3 H_2 per minute (1500 Nm^3/h), part of the product is used for repressurising in bed B

Bed B : **Pressure equalisation and repressurisation (E1 R)**: Pressure equalisation with bed D is

followed by a repressurisation, consuming 10 Nm^3 H_2 per minute (600 Nm^3/h) from bed A

Bed C : **Countercurrent blowdown and purge (D P)**: consuming 5 Nm^3 H_2 per minute (300 Nm^3/h) from bed D

Bed D : **Pressure equalisation and providing purge (E1 PP)**: Pressure equalisation with bed B is followed by a cocurrent depressurisation producing 5 Nm^3 H_2 per minute (300 Nm^3/h), the effluent gas is used to purge bed C

This sequence highlights the staggered fashion as mentioned in Chapter 3. The purposes of having this staggered sequence are: (1) to maintain continuous production of hydrogen by having at least one bed undergoing the adsorption step every 4 minutes (products oscillations flattening), (2) to utilize the pressure (hydrogen) from the pressure adsorption step and cocurrent depressurization step to supply for other beds that require hydrogen to repressurise and purge, respectively, and (3) to enable pressure equalisation. In other words, the 4 beds are dependent, meaning that the failure of one bed can affect the functionality of others and break the whole cycle. Such dependence on adsorbers in a 4-bed PSA system defines the characteristics of reliability models and will be discussed further in this chapter.

A configuration of the 4-bed hydrogen PSA system under study is shown in Figure 4.1. The PSA unit is directed by a system of 20 switch valves. Four adsorbers are connected in parallel flow relation between the feed manifold and product manifold. Four automatic valves 1A, 1B, 1C, and 1D direct feed gas flow to bed A, bed B, bed C, and bed B, respectively. Valves 2A, 2B, 2C, and 2D direct production flow from these beds to the production manifold. Adsorbers are joined at their inlet ends to the depressurization and purge effluent manifold through valves 3A, 3B, 3C, and 3D. Valves 4A, 4B, 4C, and 4D are provided at the discharged ends of the four adsorbers to pass the cocurrent depressurization gas from one bed for use as pressure equalisation and purge gas in another bed. Similarly, valves 5A, 5B, 5C, and 5D connect the discharged ends of the four adsorbers in the product pressurization manifold in order to direct high pressure to the adsorber that is undergoing the repressurisation step. Notice that the suffix in the name of valves shows which adsorber the valve belongs to. Hereafter, we will consider an adsorber and its five associated valves as a subsystem, so-called a line, in the PSA unit. Failure of a valve can lead to the interrupted flow coming in or out of an adsorber. Failure modes and failure rates are discussed in detail later in this chapter.

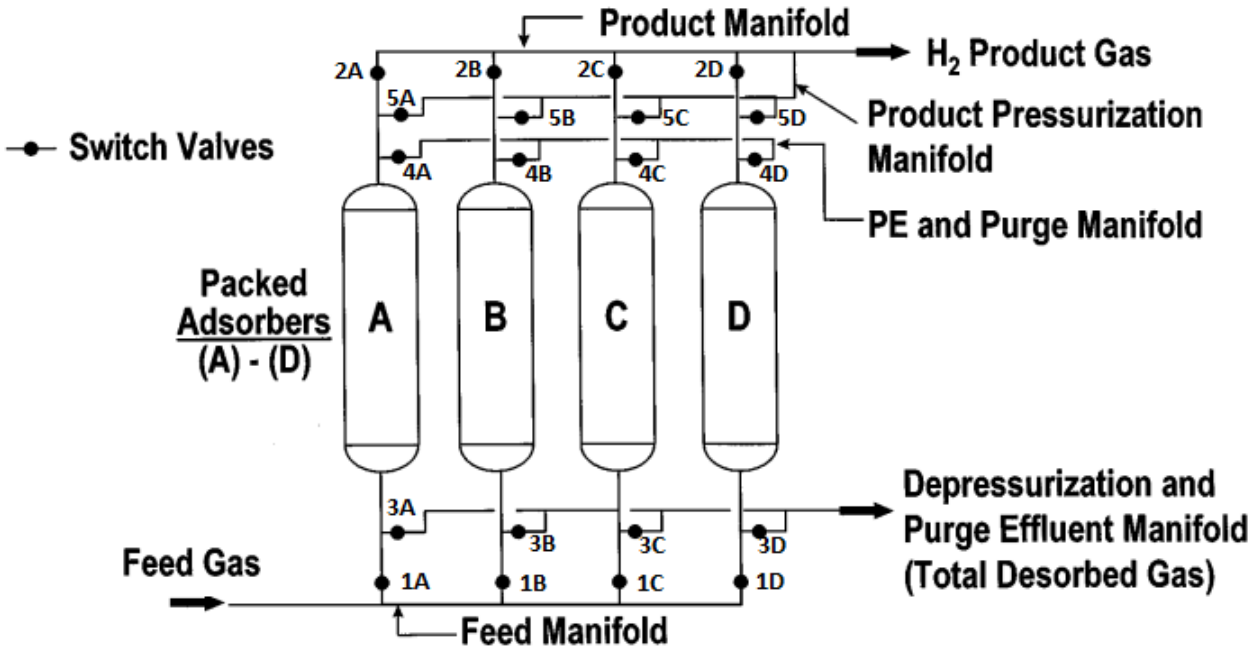


Figure 4.1: Process flowsheet of a 4-bed PSA system (Waldron, 2000)

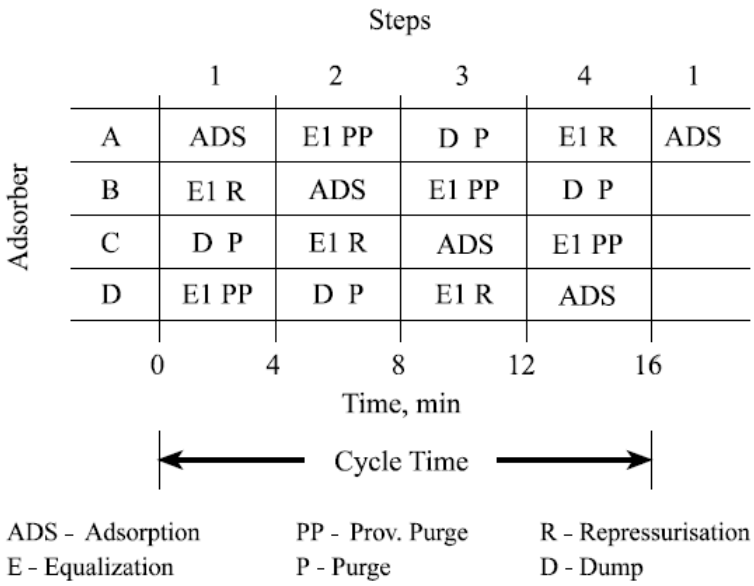


Figure 4.2: Standard 4-bed cycle sequence (Stöcker et al., 1998)

4.2 Reliability Data

Reliability data of PSA systems was first collected in 1988. This survey was performed on 10 randomly different PSA units consisting of 4-, 6-, and 10- adsorber systems. Regardless of the type of failures from different components, the number of total failures per year for 10 PSA systems is 42. Data on a component malfunction in this investigation can be found in Stöcker et al. (1998). Based on this reliability survey, the most unreliable component is the valve Stöcker et al. (1998). This is reasonable because of the large number of switches of valves per year during operation, e.g., few million switches per year. Therefore, in this study, reliability data of valves will be accessed and used in modelling. We further assume that the valves in a 4-bed PSA system are identical and share the same failure rates.

To carry out the study, a database for reliability data of valves is required and OREDA (Offshore and onshore REliability DAta project) is the most well-known resource (SINTEF, 2009). Since the 1980s, OREDA has produced comprehensive equipment reliability data for the oil & gas industry. One benefit of using data from OREDA is that failure rates are collected according to failure modes. It considers four types of failures known as critical, degraded, incipient, and unknown failure. Only critical failure modes and their active repair hours are identified in this work (Table 4.1). The data belong to topside equipment, or more specifically, control and safety equipment. Note that OREDA provides three values of failure rate, including the mean of the estimated constant failure rate, lower and upper 90% uncertainty level for the estimated constant failure rate (SINTEF, 2009). Here 90% uncertainty level means that the probability of the true failure rate falling into the interval of the lower value and the higher value is 0.9. With respect to repair time, the OREDA database categorizes active repair hours in terms of mean and max. This work adopts the mean failure rate and mean active repair hours.

Ideally, to maintain continuous hydrogen production even when a failure happens at valves or adsorbers, the configuration of a 4-bed PSA system can be altered by switching valves in the system. This could, however, break the cycle sequence designed for a 4-bed PSA as shown in Figure 4.2. For the sake of simplicity, we assume that the 4-bed hydrogen PSA system in this research strictly follows the standard 4-bed cycle sequence in Figure 4.2. Then, any critical failure from valves can lead to the failure of the corresponding adsorber/line, which causes the system failure. From this assumption, one can compute the mean failure rate of one line, denoted by λ, by summing the mean failure rates of the five corresponding valves:

$$\lambda = 1.49 \times 10^{-5} h^{-1} \tag{4.1}$$

By assuming this failure rate remains constant over the simulation time, we can have that time to failure of one line in the PSA system is exponentially distributed.

Table 4.1: Reliability data of valves in control and safety equipment with respect to critical failure modes (SINTEF, 2009)

Failure mode	Failure rate (per 10^6 hours)			Active rep. hours	
	Lower	Mean	Upper	Mean	Max
Critical	3E-4	2.98	13.63	6.2	100
Delayed operation	-	0.14	0.74	3.0	6.0
External leakage - Process medium	-	0.36	1.88	32	100
External leakage - Utility medium	4E-4	0.04	0.14	8.0	8.0
Fail to close on demand	-	0.97	5.01	3.8	23
Fail to open on demand	2E-4	0.93	4.17	5.9	17
Fail to regulate	0.03	0.43	1.2	2.4	4.0
Internal leakage	5E-4	0.11	0.42	6.3	8.0
Low output	9E-4	0.04	0.13	2.0	2.0
Spurious operation	-	0.08	0.44	6.0	6.0
Structural deficiency	-	0.17	0.93	5.0	9.0
Valve leakage in closed position	3E-4	0.14	0.55	10.0	10.0
Other	9E-4	0.04	0.13	-	-

4.3 Maintenance Policy

A maintenance policy applied in this study is corrective maintenance in which the valves are always repaired to an "as-good-as-new" state from failed state. We consider neither preventive maintenance nor a degraded state in this study. The Mean Time to Repair (MTTR) of the valve is assessed based on the mean active repair hours from OREDA (see Table 4.1). Mean active repair time implies the time requires to repair the failure and restore the function of the failed component, meaning that time to shut down the system, issue work orders, and wait for spare parts is excluded (SINTEF, 2009). MTTR is a weighted average of the active repair times for the valve with the weighting factor being the failure rate of each means of active repair time. Hence, we can calculate MTTR and get MTTR = 7.615 h. Repair rate, denoted by μ, is then calculated as:

$$\mu = \frac{1}{MTTR} = \frac{1}{7.615} h^{-1} \tag{4.2}$$

We assume that all the repair times are exponentially distributed. Furthermore, we assume that there is one repairman only who can carry out the repair work on one single valve at once. There is no priority in corrective maintenance of valves, meaning that repairs are implemented on a first-come-first-served basis. As a result of that assumption, repairs are initiated right after failure when the repair man is available, and there is a delay in repair completion when failures occur during repair time.

4.4 Scenario 1: PSA without a buffer

In the first scenario, a 4-bed PSA system with a normal configuration as shown in Figure 4.1 is studied. Modelling tools that are used to model the PSA without a buffer are Markov process, discrete event simulation in Python, and MIRIAM RAM Studio. Due to the dependency of the four lines in the cycle of 4-minute functional modes as presented in Section 4.1, the system can be considered as a 4-o-o-4 system, and production capacity is either 100% or null. Put it differently, failure in one line leads to the failure of the PSA system. Then corrective maintenance is performed immediately right after the system fails to bring the system back to its normal state. Time to detect failures is negligible.

Markov Model

With constant failure and repair rates, the PSA system satisfies the Markov property. This means that information regarding the process in the past does not affect the state of the system in the future. Accordingly, time to failure and time to repair are distributed exponentially. It also means that there is no ageing in the system. Hence, there are no degradation states between the normal state and the failed state in the system.

The Markov model of the PSA under studying has five states, with the number in each state showing the number of adsorbers that are working (Table 4.2). For example, state 4 indicates that 4 adsorbers are working normally, meaning that the system produces 100% capacity. From state 3 to state 0, the system fails and does not produce any hydrogen. Transitions between states are failure and repair rates. All states and their transition rates are represented graphically in a Markov transition diagram in Figure 4.3. Accordingly, transition matrix A is obtained from the Markov transition diagram:

$$\mathrm{A} = \begin{bmatrix} -\mu & \mu & 0 & 0 & 0 \\ \lambda & -(\lambda+\mu) & \mu & 0 & 0 \\ 0 & 2\lambda & -(2\lambda+\mu) & \mu & 0 \\ 0 & 0 & 3\lambda & -(3\lambda+\mu) & \mu \\ 0 & 0 & 0 & 4\lambda & -4\lambda \end{bmatrix} \tag{4.3}$$

Table 4.2: System states

State	Description	Production capacity
4	4 adsorbers in perfect state	100%
3	3 adsorbers in perfect state and 1 failed	0%
2	2 adsorbers in perfect state and 2 failed	0%
1	1 adsorbers in perfect state and 3 failed	0%
0	4 adsorbers failed	0%

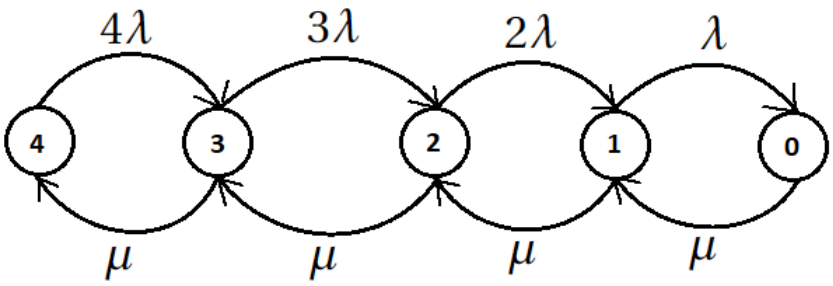


Figure 4.3: Markov diagram in the first scenario

The steady-state probabilities P_0, P_1, P_2, P_3, P_4 are calculated by solving the following equations:

$$[P_0, P_1, P_2, P_3, P_4] \cdot \begin{bmatrix} -\mu & \mu & 0 & 0 & 0 \\ \lambda & -(\lambda+\mu) & \mu & 0 & 0 \\ 0 & 2\lambda & -(2\lambda+\mu) & \mu & 0 \\ 0 & 0 & 3\lambda & -(3\lambda+\mu) & \mu \\ 0 & 0 & 0 & 4\lambda & -4\lambda \end{bmatrix} = [0,0,0,0,0] \tag{4.4}$$

and

$$P_0 + P_1 + P_2 + P_3 + P_4 = 1 \tag{4.5}$$

Solving equations 4.4 and 4.5 yields the steady-state probability of the perfect state (state 4):

$$P_4 = \frac{\mu^4}{24\lambda^4 + 24\lambda^3\mu + 12\lambda^2\mu^2 + 4\lambda\mu^3 + \mu^4} = 0.999546198 \tag{4.6}$$

Here, we assume that the planned production rate of the PSA system, denoted by $D_0(t)$, is a constant over a 50-year period, and production rate at any time t when the system is on the perfect state, $D(t)$, is equal to the planned production rate. The production availability of the

PSA system over a 50-year period is then computed according to Equation 2.1 as:

$$A_{11}(t_1, t_2) = \frac{D_0(t) \times P_4(t_2) \times (t_2 - t_1)}{D_0(t) \times (t_2 - t_1)} = P_4(t_2) \tag{4.7}$$

where $t_1 = 0$ and $t_2 = 50$ years. It is argued that a steady state is achieved after three times the shortest expected transition time, which is $\frac{3}{\mu} = 22.85$ h in this case. This means that the system can reach a steady state after one day. Hence, the state probabilities of the PSA unit after 50 years can be used as steady states. That is,

$$P_4(t_2 = 50 \text{ year}) \approx P_4 \tag{4.8}$$

Thus, the production availability of the PSA system without a buffer after 50 years calculated from the Markov model is:

$$A_{11} = P_4 = 0.999546198 \tag{4.9}$$

Using Python to find the time-dependent solution at $t_2 = 50$ years gives us the same production availability (See Appendix C.1.1).

Discrete Event Simulation in Python with Simpy

In this section, the reliability model of the 4-bed PSA system without a buffer is established by using Python. The Python code of this model (See Appendix C.2.1) comprises two main user-define functions. The first function uses SimPy in Python (SimPy, 2013) to simulate the operation of the PSA system over a period of 50 years while the second user-define function deals with the calculation of production availability.

SimPy is used to record the operational profile of individual line in the PSA system. The system changes its state when (1) a critical failure of a valve happens and (2) a repair is done. Accordingly, two types of "timeout" events whose time delays are Time-To-Failure (TTF) and Time-To-Repair (TTR) occur throughout the time of simulation. In other words, the system continuously produces hydrogen until a failure event takes place. The time at which this event occurs is recorded. Simultaneously, the process requests a repairman in the resource class to do the maintenance. The number of repairmen in this case study is limited to one person, which is implemented in SimPy by a resource of limited capacity, e.g., the capacity of one. If this repairman is occupied, the process waits until the repairman becomes available again. Simply put, the repairman is resumed and released by processes from time to time. If the repairman is available, Time-To-Repair is yielded and recorded. After being repaired, the system is brought back to production mode.

After the operational profile of each line is gained, their dependence is taken into account to produce an operational profile of the PSA system as a whole. In particularly, a 4-o-o-4 relation

is established among the four lines in order to calculate the up-time of the PSA system. This coincides with the assumption that the PSA system without a buffer is down when one or more lines fail.

Run-length and number of replications are 50 years and 50000 replications, respectively. Convergence graph plotting the cumulative mean of production availability against the number of replications is shown in Figure 4.4. The discrete event simulation model of the 4-bed PSA without a buffer yields the average production probability as:

$$A_{12} = 0.999546511 \tag{4.10}$$

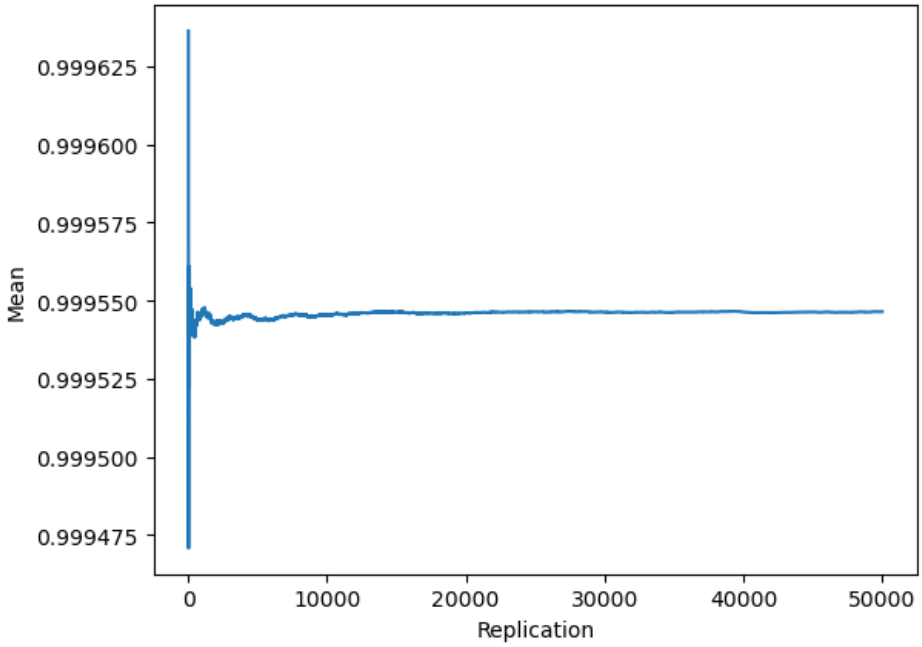


Figure 4.4: Convergence graph of the PSA model without a buffer in Python

MIRIAM RAM Studio

The reliability model of the 4-bed PSA system without a buffer is rather simple in MIRIAM RAM Studio. The flow network contains one entry point, one discharge point, and one process stage (Figure 4.5). There are three types of throughputs in the system, namely syngas, hydrogen, and offgas. Syngas is the input/supply flow of the PSA system. The process stage converts syngas into hydrogen which is collected at the discharge point. Part of the hydrogen production is used to clean the adsorbers, producing offgas which is also collected at discharge point.

The process stage is a subsystem of four identical streams representing four lines in the 4-bed PSA unit (Figure 4.6). Each stream is a series of one adsorber and five valves following the configuration in Figure 4.2. Reliability data collected in the previous sections are the input data

of the valve. The number of streams normally running is 4, while number of streams required to run is 1. This together with the min flow at discharge point, which is equal to demand flow, will define the how the process stage fulfills the hydrogen demand. Particularly, all four adsorbers have to work to meet the hydrogen demand at discharge point. This corresponds to state 4 in the Markov Model presented above. Zero production is the consequence of any failure in the process stage.

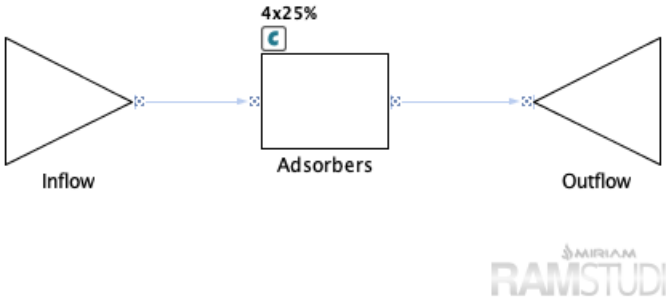


Figure 4.5: Model of the 4-bed PSA system without buffer in MIRIAM RAM Studio

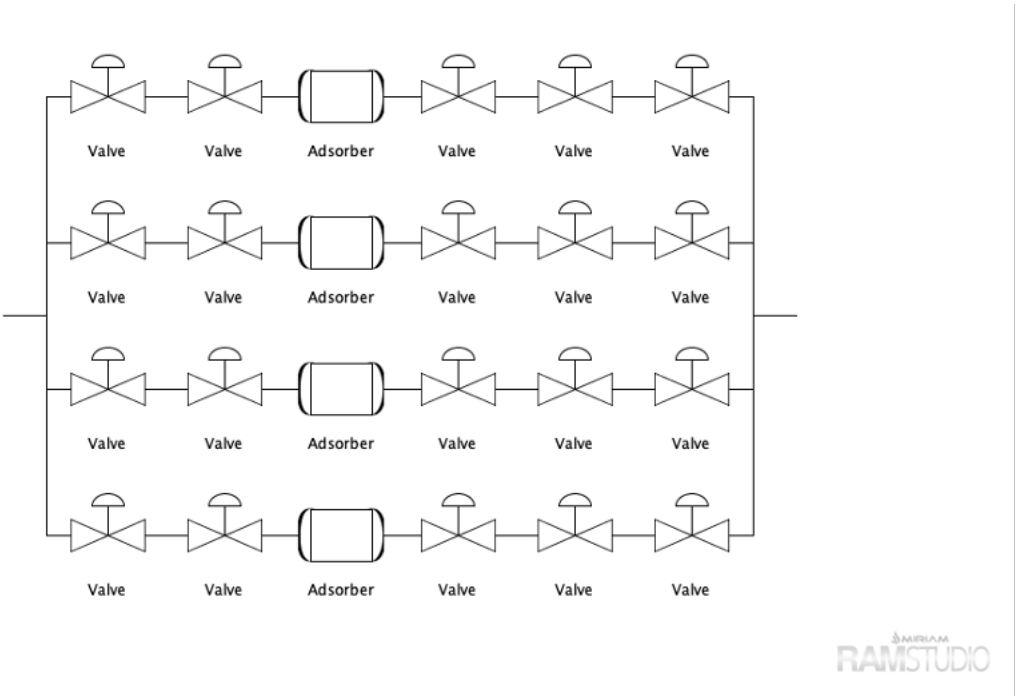


Figure 4.6: Process stage of the PSA model without a buffer

The flow algorithm strives to maximize the flow through the network in order to meet demand flow at the Boundary Point (discharge point). The flow going into the system is syngas whose supply flow is set to be constant, i.e., 1800 m^3/h. Regardless of the four different phases in

the 16-minute program of the PSA system (Figure 4.1), the throughput of syngas at each stream in process stage is modelled as a constant flow as well. In other words, each stream produces hydrogen from syngas with a capacity of 225 m^3/h. The yielding ratio is assumed to be 0.5. The demand flow at discharge point is 900 m^3 hydrogen per hour, which equals to the maximum capacity of the process stage.

Run-length of one simulation in MIRIAM is 50 years. Number of replications is 50000 which is equal to that of the discrete event simulation model. Figure 4.7 shows the average production availability as well as standard deviation and error of the mean after running 50000 replications. Average production availability in this case is:

$$A_{13} = 0.999546359 \tag{4.11}$$

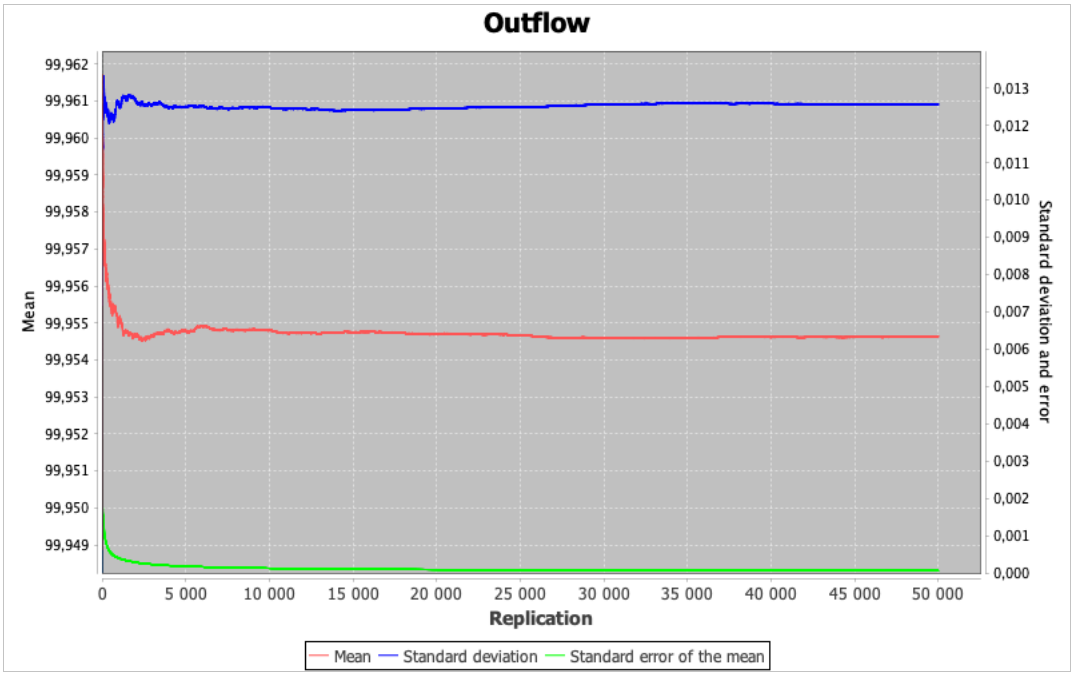


Figure 4.7: Convergence graph of the PSA model without a buffer in MIRIAM

4.5 Scenario 2: PSA with a buffer

Adapting the idea of having an additional tank in the PSA unit from William D Marsh (1961), in this second scenario, we introduce a buffer into the 4-bed PSA system.

The intention of installing a buffer is to store and provide high-pressure hydrogen to the system when one adsorber fails. Accordingly, the buffer can temporarily take over the role of adsorbers that are supposed to produce high-pressure hydrogen but fail to do so. For example, assume that an adsorber fails at the high-pressure adsorption mode (ADS), while this adsorber

is being repaired, the buffer takes over and provides hydrogen for the system. This helps to keep three other adsorbers working while repairing one adsorber. In short, the buffer is withdrawn when the failed adsorber is supposed to perform two modes, namely high-pressure adsorption (ADS) and pressure equalisation and providing purge (E1 PP). This means that the buffer is not withdrawn continuously. At two other modes, E1 R and D P, where the failed adsorber is supposed to consume hydrogen for repressurising and purging, respectively, the buffer is not withdrawn but refilled with such amount of hydrogen. We refer to this kind of hydrogen as "cleaning" hydrogen. In case the buffer is full and cannot take in more cleaning hydrogen, we send this cleaning hydrogen to the output. As a result, failures at E1 R and D P modes are more beneficial for the buffer to get refilled than failures at ADS and E1 PP modes. Furthermore, the limited capacity of the buffer leads to a problem that the buffer can get empty after running for a period of time. If the buffer is empty and repairs are not finished yet, the whole system fails to produce hydrogen. From these aspects, we can see that installing a buffer to the PSA makes the system more complex and dynamic.

Assumptions related to the buffer are made as the followings:

1. Initialization: The buffer has a limited capacity which is 900 Nm^3 hydrogen, and it is empty at the beginning of the production process

2. Refilling policy: The buffer is refilled either by 1% production from adsorbers undergoing the high-pressure adsorption mode (ADS) at perfect state or (cleaning) hydrogen that is supposed to be consumed by a failed adsorber at pressure equalisation and repressurisation mode (E1 R) and countercurrent blowdown and purge mode (D P). Refill rate is then 9, 600, and 300 m^3 hydrogen per hour, respectively. If buffer is full, cleaning hydrogen is sent to the production

3. Withdrawing policy: When one adsorber fails (three others working) and buffer is not empty, the buffer takes over at times this adsorber is supposed to perform the high-pressure adsorption (ADS) and pressure equalisation and providing purge (E1 PP) modes. Withdraw rate is thus 1500 and 300 m^3 hydrogen per hour, respectively.

4. Maintenance policy: The buffer is connected with adsorbers by valves which are inspected by the repairman when he is idle such that these valves are always in the perfect state. Additionally, the inspection time is negligible.

The difference between the second scenario and the first one is that the model in the second scenario is addressed in a more dynamic way. More specifically, the PSA system with a buffer should be modeled over each 4-minute period during the life time. By doing this, it facilitates the refilling and withdrawing policy (RWP) of the buffer and makes sure that functional modes of the PSA strictly follow the standard cycle sequence as defined in Figure 4.2. Keeping track of the

production mode of adsorbers after every four minutes leads to the need of establishing time-driven models (Cassandras and Lafortune, 2010), which will challenge and test the flexibility of the modelling tools under study. Models that are both time-driven and event-driven are called hybrid models/systems.

Markov Model

The prevalent Markov modelling approach faces with challenges when modelling the dependability between the buffer and the PSA. As defined above, refill/withdraw rate of the buffer is not a constant during the lifetime of the system but a dynamic, changeable value depending on states of the system, functional modes of the adsorbers, and current level of the buffer. Refilling time and withdrawing time are then not exponentially distributed. For instance, at the beginning of the production, if we assume that time to the first failure is much larger than time to continuously fill up the buffer for the first time (T), then we have:

$$T = \frac{900}{9} = 100h. \tag{4.12}$$

After the first failure and the first repair, to determine the next event associated with the buffer, we need to know the next failure and the current level of the buffer. The latter depends on the previous uptime and downtime, which is associated with two events in the past. Whereas, Markov property requires that the probability of each event depends only on the state attained in the previous event (Limnios and Oprişan, 2001; Rausand and Høyland, 2004). Moreover, to keep track of the buffer's level, the buffer should be modeled as a Continuous-Variable Dynamic System (CVDS) with a time-driven state mechanism (Cassandras and Lafortune, 2010). In other words, the buffer's state should be governed after every 4 minutes by means of a continuous variable, which takes on any real number in a subset of the real plane R. That is out of the scope of this study since the study only focuses on discrete-state event-driven Markov processes. Shortly, Markov process is not suitable for modelling the dynamic behaviors of the buffer.

Some assumptions are needed for the Markov process to approximately model the PSA with a buffer. Firstly, refilling takes place only at the perfect stage (four adsorbers working), and the buffer is always empty when we start refilling it. Plus, refill rate is constant over time, which is 9 m^3 per hour (1% of the total production). Secondly, the buffer is always full when we start withdrawing it, and withdraw rate is constant which is 225 m^3 per hour.

Based on the two assumptions above, we can establish a Semi-Markov model of the PSA system with a buffer. More specifically, the first assumption allows us to model the refilling time by a deterministic distribution, denoted by $DET(T)$, with T calculated in Equation 4.12. Similarly, thanks to the second assumption, withdrawing time can be governed by a deterministic distribution, denoted by $DET(L)$, with parameter L = 4 h which is the rundown time of buffer

when withdraw rate is constantly 225 m^3 per hour. System states are presented in Table 4.3. The corresponding Semi-Markov transition diagram is shown in Figure 4.8. Compared to the first scenario, the number of system states in this Semi-Markov model is double as system states are now associated with the state of the buffer, i.e., empty (E) and full (F). There is no transitions between states 2^E and 2^F or between states 1^E and 1^F because we assume that the buffer is withdrawn only at state 3 and it is filled up only at perfect state and from zero level. Note that the system is at state 4^E (4 adsorbers work, buffer is empty) at the beginning of the production process. At this state, we spend 1% of production refilling the buffer. Thus, the corresponding production capacity of the system is 99%.

Table 4.3: System states in Semi-Markov model of the PSA with a buffer

State	Description	Production capacity
4^E	4 adsorbers work, buffer is empty	99%
4^F	4 adsorbers working, buffer is full	100%
3^F	3 adsorbers working, 1 adsorber failed, and buffer is full	100%
3^E	3 adsorbers working, 1 adsorber failed, and buffer is empty	0%
2^F	2 adsorbers working, 2 adsorber failed, and buffer is full	0%
2^E	2 adsorbers working, 2 adsorber failed, and buffer is empty	0%
1^F	1 adsorbers working, 3 adsorber failed, and buffer is full	0%
1^E	1 adsorbers working, 3 adsorber failed, and buffer is empty	0%
0^F	4 adsorber failed, buffer is full	0%
0^E	4 adsorber failed, buffer is empty	0%

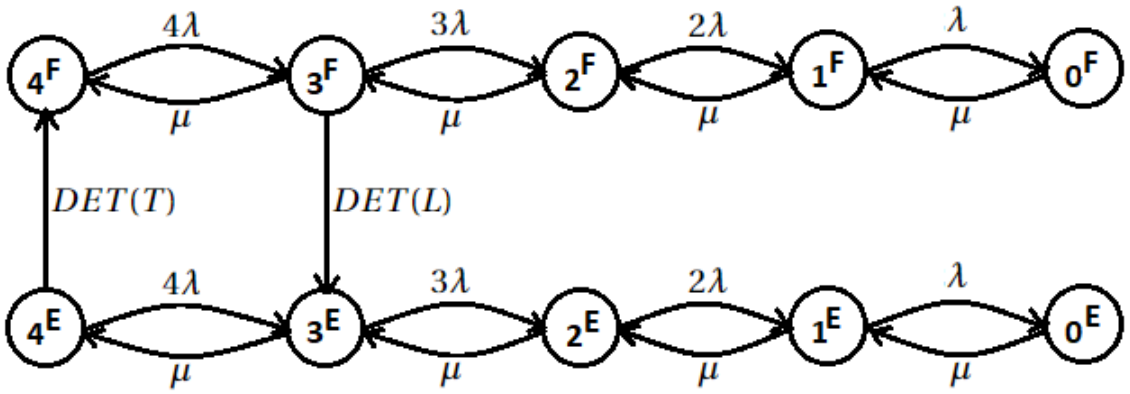


Figure 4.8: Semi-Markov diagram of the PSA with a buffer

Then, the two deterministic distributions, $DET(T)$ and $DET(L)$, in the Semi-Markov model are approximated and replaced by k-stage Erlang distributions, denoted as $E_k(\lambda^T)$ and $E_k(\lambda^L)$,

respectively. In this study, we choose $k = 5$. With respect to the approximation of $DET(T)$, each stage in the 5-stage Erlang distribution corresponds to an exponential distribution with rate λ^T. That is,

$$\lambda^T = \frac{k}{T} = \frac{5}{100} = 0.05 \text{ h}^{-1}. \tag{4.13}$$

Similarly, each stage in the 5-stage Erlang distribution associated with withdrawal corresponds to an exponential distribution with rate λ^L

$$\lambda^L = \frac{k}{L} = \frac{5}{4} = 1.25 \text{ h}^{-1}. \tag{4.14}$$

This results in a Markov process with 30 stages (Figure 4.9). The description of these stages is presented in Table 4.4. Notice that besides two original states, i.e., empty and full, the buffer or the system in the Markov model now has more intermediate, discrete levels (states).

Table 4.4: System states in Markov model of the PSA with a buffer

State	Description	Capacity
4^E	4 adsorbers working; buffer is empty	99%
4^{20}, 4^{40}, 4^{60}, 4^{80}	4 adsorbers working, buffer is 20%, 40%, 60%, and 80% full, respectively	99%
4^F	4 adsorbers working; buffer is full	100%
3^E	3 adsorbers working; buffer is empty	0%
3^{20}, 3^{40}, 3^{60}, 3^{80}	3 adsorbers working; buffer is 20%, 40%, 60%, and 80% full, respectively	100%
3^F	3 adsorbers working; buffer is full	100%
2^E	2 adsorbers working; buffer is empty	0%
2^{20}, 2^{40}, 2^{60}, 2^{80}	2 adsorbers working; buffer is 20%, 40%, 60%, and 80% full, respectively	0%
2^F	2 adsorbers working; buffer is full	0%
1^E	1 adsorbers working; buffer is empty	0%
1^{20}, 1^{40}, 1^{60}, 1^{80}	1 adsorbers working; buffer is 20%, 40%, 60%, and 80% full, respectively	0%
1^F	1 adsorbers working; buffer is full	0%
0^E	4 adsorbers failed; buffer is empty	0%
0^{20}, 0^{40}, 0^{60}, 0^{80}	4 adsorbers failed; buffer is 20%, 40%, 60%, and 80% full, respectively	0%
0^F	4 adsorbers failed; buffer is full	0%

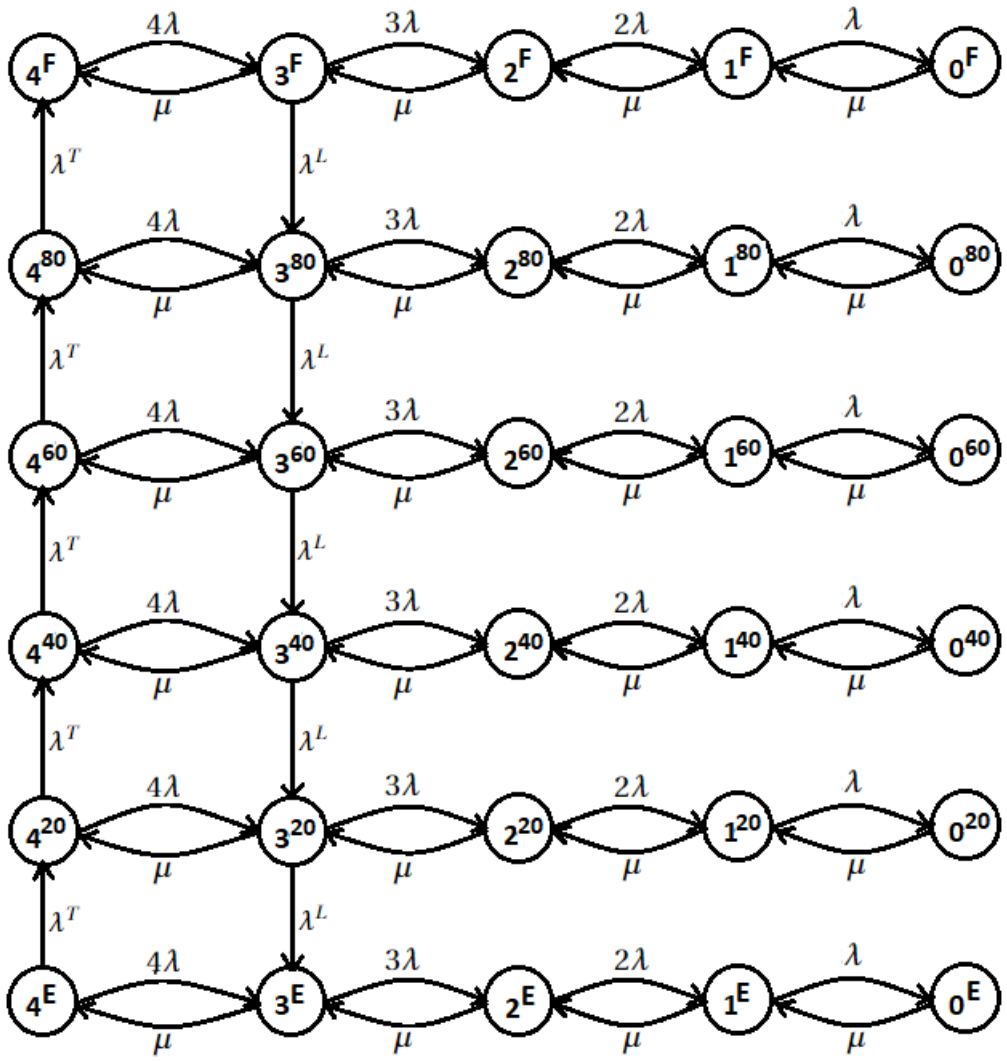


Figure 4.9: Markov diagram of the PSA with a buffer

Using Python to find the steady states of the system (See Appendix C.1.2) helps to calculate the production availability of the PSA unit with a buffer over 50 years:

$$\begin{aligned} A_{21} &= 0.99\left(P_{4^E} + P_{4^{20}} + P_{4^{40}} + P_{4^{60}} + P_{4^{80}}\right) + \left(P_{4^F} + P_{3^F} + P_{3^{80}} + P_{3^{60}} + P_{3^{40}} + P_{3^{20}}\right) \\ &= 0.999679653 \end{aligned} \tag{4.15}$$

Discrete Event Simulation in Python with SimPy

Complex model

We use the term "complex" to refer to the way of modelling in which the 4-minute functional modes of adsorbers are captured and simulated in the model. It is, in other words, a hybrid model because we take into account the time-driven aspect. In Python, this is leveraged by the

use of classes/objects and Try-Except statement. More specifically, a class named Line with two object methods is created (see Appendix C.2.2), representing a line in the PSA unit. The first method in the Line class, $working(self)$ method, executes the performance of the line and the buffer in every 4 minutes over the simulation time. Whereas, the second object method, $break\_valve(self)$, tries to interrupt this main process with failures from the adsorber. Similar to the model in the first scenario, such interruption is facilitated by the use of simpy.Interrupt method and takes place only when the adsorber is currently working. When a failure occurs, the Try statement in the $working(self)$ method is not executed, instead, the Except statement is performed. Regardless if the line is up or down, the 4-minute functional mode is still maintained such that it follows the 16-minute cycle of the PSA. Hence, functional mode of the line is always moved to the next mode before the $yield$ function shifts the simulation time to the next 4-minute.

In this model, the number of adsorbers working at a given time is recorded in a variable named "state". This variable initiates with a value of four, meaning that four adsorbers work in the beginning, and changes when a failure occurs and a repair is finished. The variable "state" along with the mode of adsorbers decide how the buffer is refilled and withdrawn in accordance to characteristics of the buffer. The buffer is created by the use of a common resource of the simulation environment. It is empty at the beginning of the process. The buffer has a limited capacity of 900 m^3 hydrogen which is equivalent to four-hour continuous withdrawal with a withdraw rate of 225 m^3/h. However, withdrawal in this complex model is not continuous but dependent upon the mode of adsorber, just similar to the refilling policy. For example, we only withdraw from the buffer when the failed adsorber is supposed to perform the high-pressure adsorption (ADS) mode and pressure equalisation and providing purge (E1 PP) mode but fails to do so. With SimPy, one can take resources from the buffer and refill it by using the function $buffer.get()$ and $buffer.put()$, respectively.

One issue with the complex model in Python is that the executive time is long. It takes roughly 120 seconds to run the model with run-length of 50 years. Thus, we run the model with 50 replications. The convergence graph from this run is shown in Figure 4.10. The resulting average production availability is

$$A_{22}^{*} = 0.999691577 \tag{4.16}$$

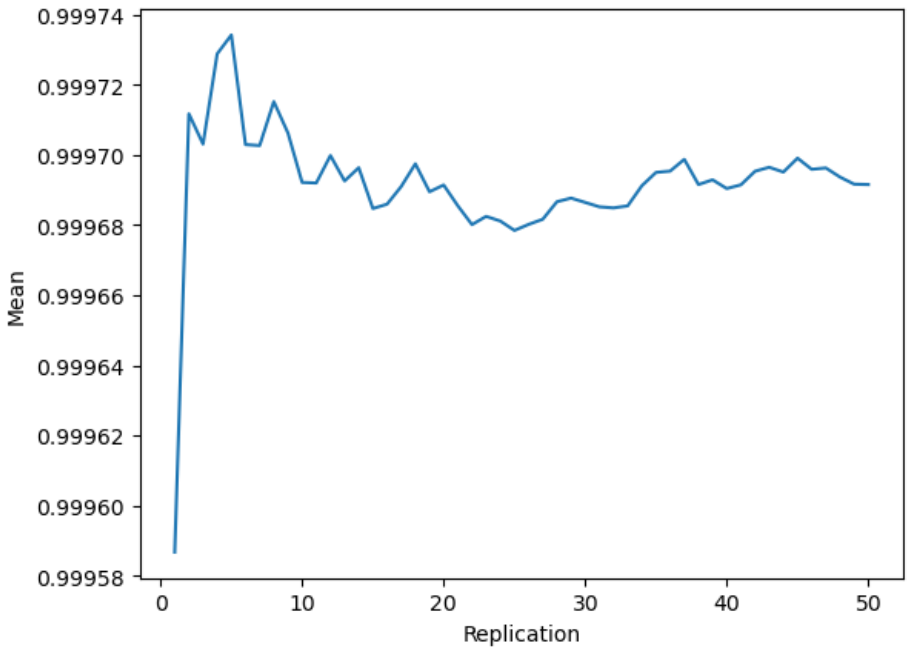


Figure 4.10: Convergence graph of the complex PSA model with a buffer in Python

Simple model

As it it time-consuming to run the complex model of the PSA with a buffer in Python, we try to simplify the case to reduce the running time and come up with a "simple" model.

The simple model is similar to the model of the PSA without a buffer in the first scenario. We use SimPy to simulate each line of the system and access their operational profiles over 50 years. Then, in the calculation of production availability of the simple Python model, we take into account the relation of the buffer to the model in the second scenario (see Appendix C.2.3).

Compared to the complex model, the cycle of 4-minute operational modes of adsorbers is ignored in this simple model. Hence, we consider the production rate of each line in the PSA to be constant over time, which is 225 m^3 hydrogen per hour. This is equal to the withdraw rate of the buffer. In total, the PSA unit with four beds/lines produces 900 m^3 hydrogen per hour. The buffer is only used when one adsorber fails and three other adsorbers are working. Filling rate of the buffer is 9 m^3 hydrogen per hour when the system is at state 4 (4 adsorbers working), and there is no refilling at state 3. The amount of hydrogen used to fill up the buffer is subtracted from the total production, resulting in a decrease in production rate to 891 m^3 hydrogen per hour. Production rate comes back to 900 m^3 hydrogen per hour when the buffer is full and the system is at perfect state. In short, refilling and withdrawing policy in this case is simpler than the complex model because we skip to model the operational modes.

After running 50000 replications (See convergence graph in Figure 4.11), each has a run-

length of 50 years, average production availability of the simple model of PSA with a buffer is:

$$A_{22} = 0.999683307 \tag{4.17}$$

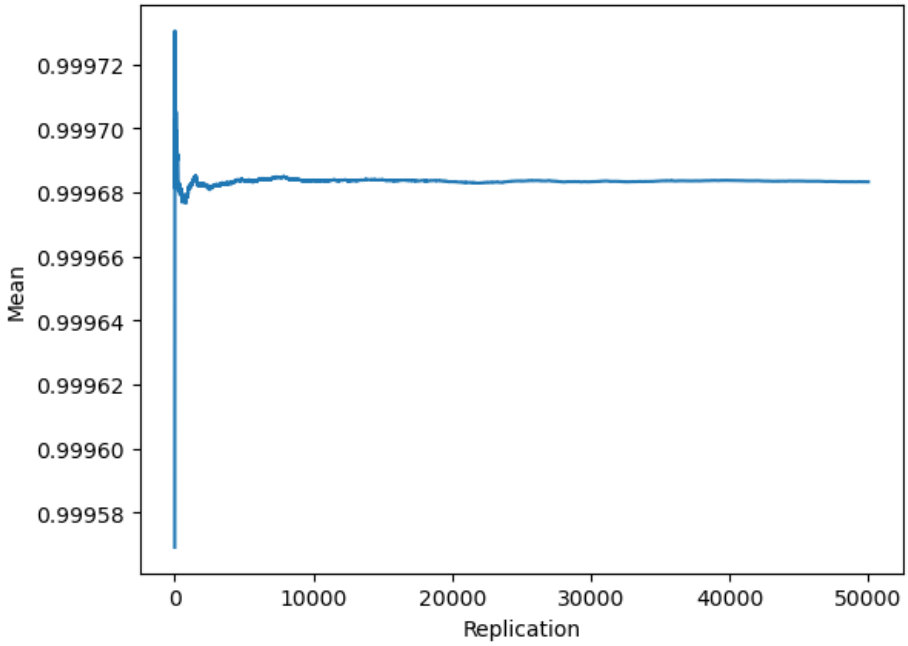


Figure 4.11: Convergence graph of the simple PSA model with a buffer in Python

MIRIAM RAM Studio

Complex model

Similar to the complex model in Python, we aim to model the 4-minute functional modes of the PSA with a buffer in MIRIAM RAM Studio.

The flow network of the PSA with a buffer is more sophisticated than that of the PSA without buffer. The flow network in this case has 4 entry points, 2 discharge points, 8 process stages, and one storage unit (Figure 4.12). Amongst 8 process stages, Line_A, Line_B, Line_C, and Line_D are the main process stages. Each of them is a series of five valves whose reliability data are collected previously. In the flow network, they are connected to each other, and each line has its own entry point. Four other process stages contain a dummy valve in each of them. They are elements in advanced operation rules (AOR) which help to control the flow in the network in order to achieve the refilling/withdrawing policy of the buffer. Advanced operation rules (AOR) are introduced further in this section. The storage unit is used as a buffer which allows hydrogen go through and stores hydrogen if possible. There exists three types of throughputs in the flow network, namely syngas, hydrogen, and offgas. Hydrogen and offgas produced from the four

main process stages are collected at the hydrogen discharge point and offgas discharge point, respectively.

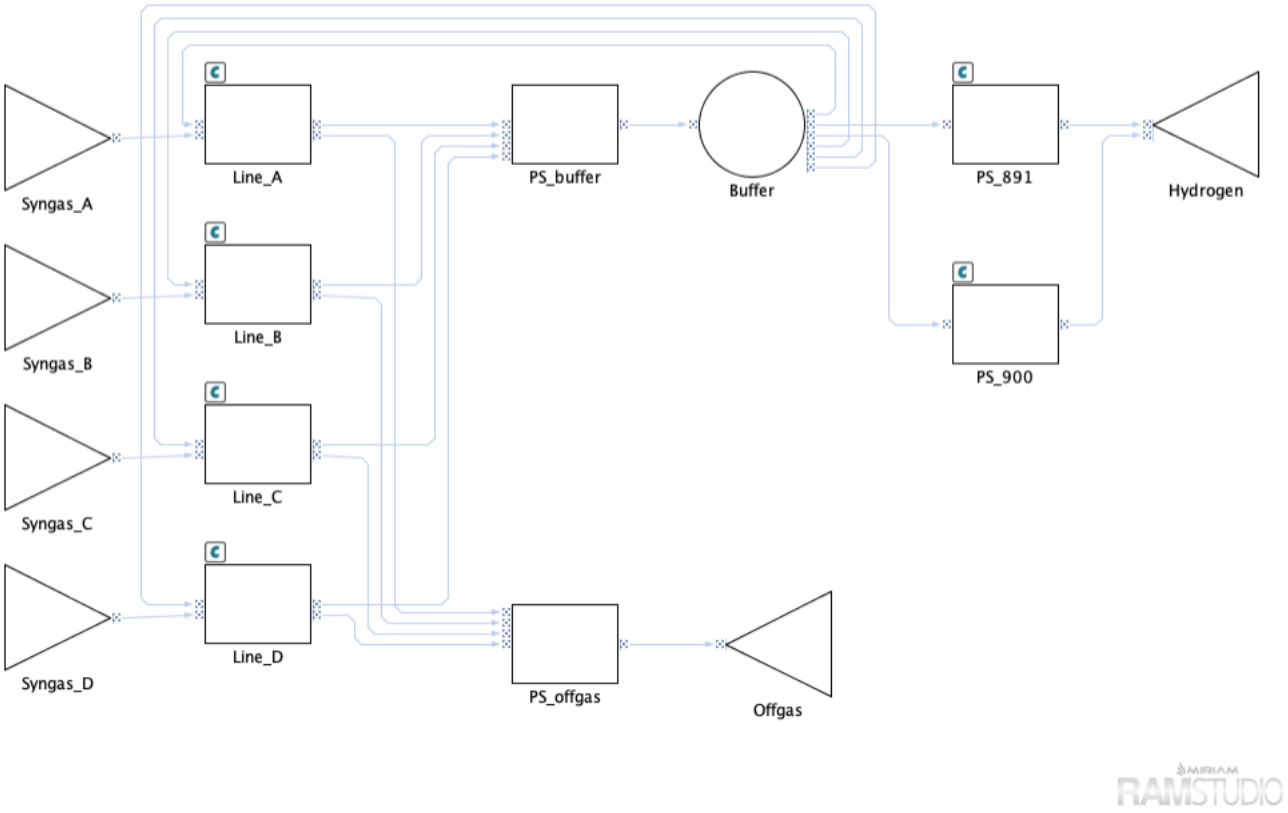


Figure 4.12: Complex model of the 4-bed PSA system with a buffer in MIRIAM, simulation of 4-minute operational modes included

There are three main challenges in modelling the complex model of PSA with a buffer in MIRIAM RAM Studio.

The first challenge is how to simulate the 4-minute operational modes of the 4-bed PSA system. Unlike models of the PSA without a buffer where production rate of the system is assumed to be constant over time, the complex model of the PSA with a buffer distinguishes between four production rates of the adsorber at four different operational modes. Moreover, each adsorber has to carry out distinct operational mode at any time. The modes or production rates have to shift amongst the four lines to satisfy the 4-minute operational modes. This is achieved by using calendar to allocate input flow to each entry points and flow design (capacity) to each main process stage differently after every 4 minutes (Appendix B.1 and B.2).

The second challenge is how to fulfill the refilling and withdrawing policy of the buffer. Similar to production rate of each line, refilling and withdrawing rates also change depending on system states and functional modes of each line. This is fulfilled by combining various proper-

ties and rules in MIRIAM RAM Studio. Basic property of the buffer is its capacity which is set to be 900 m^3 hydrogen. If withdraw rate is 225 m^3 hydrogen per hour and stays constant at state 3 (3 lines working), the capacity of 900 m^3 hydrogen is equivalent to 4 hours of withdrawal which is the assumption in Markov process. This is unfortunately not the case in MIRIAM when the 4-minute operational modes are taken into consideration. More specifically, withdraw rate could be either 1500 or 300 m^3 hydrogen per hour depending on the functional modes (Appendix B.3 and B.4). The refilling policy of the buffer is facilitated by three factors: (1) demand flow at discharge point, (2) two process stages, PS_891 and PS_900, and (3) two advanced operation rules (AOR) named "AOR_891" and "AOR_900" (see 4.13). When the conditions in AOR_891 are fulfilled, the process PS_891 allows a flow of 891 m^3 hydrogen per hour direct to the discharge point while the process PS_900 closes by forcing the dummy valve to be failed. This means that at perfect state, the PSA system provides 891 m^3 hydrogen per hour to the discharge point and saves 9 m^3 hydrogen per hour in the buffer (Appendix B.5). When the buffer is full or the system is not at perfect state, the advanced operation rule named AOR_900 is applied. By doing this, we stop the refilling procedure of the buffer and allow full production at the discharge point (Appendix B.8). In addition to this, the buffer can receive "cleaning" hydrogen that is supposed to use for the countercurrent blowdown and purge mode (D P) (300 m^3 per hour) and pressure equalisation and repressurisation mode (E1 R) (600 m^3 per hour) if the adsorber undergoing these modes is failed. Refill rates in this case are either 300 or 600 m^3 hydrogen per hour (Appendix B.6 and B.7). As a result, refilling sections can come between withdrawing sections when the buffer is in use at state 3. Put it differently, we can refill the buffer although the system is not at the perfect state.

Name	Condition	Rule
AOR_3A_empty	AND	IF **Line_A**[ANY, ANY, FAILURE] AND **Buffer**[LEVEL=0] THEN **PS_buffer**[ALL, ALL, FAILURE], **PS_offgas**[ALL, ALL, FAILURE]
AOR_3B_empty	AND	IF **Line_B**[ANY, ANY, FAILURE] AND **Buffer**[LEVEL=0] THEN **PS_buffer**[ALL, ALL, FAILURE], **PS_offgas**[ALL, ALL, FAILURE]
AOR_3C_empty	AND	IF **Line_C**[ANY, ANY, FAILURE] AND **Buffer**[LEVEL=0] THEN **PS_buffer**[ALL, ALL, FAILURE], **PS_offgas**[ALL, ALL, FAILURE]
AOR_3D_empty	AND	IF **Line_D**[ANY, ANY, FAILURE] AND **Buffer**[LEVEL=0] THEN **PS_buffer**[ALL, ALL, FAILURE], **PS_offgas**[ALL, ALL, FAILURE]
AOR_4	OR	IF **Line_A**[ANY, ANY, RESTORATION] OR **Line_B**[ANY, ANY, RESTORATION] OR **Line_C**[ANY, ANY, RESTORATION] OR **Line_D**[ANY, ANY, RESTORATION] THEN **PS_buffer**[ALL, ALL, RESTORATION], **PS_offgas**[ALL, ALL, RESTORATION]
AOR_891	AND	IF **Line_A**[ANY, FLOW>0] AND **Line_B**[ANY, FLOW>0] AND **Line_C**[ANY, FLOW>0] AND **Line_D**[ANY, FLOW>0] AND **Buffer**[LEVEL<900] THEN **PS_891**[ALL, ALL, RESTORATION], **PS_900**[ALL, ALL, FAILURE]
AOR_900	OR	IF **Line_A**[ANY, ANY, FAILURE] OR **Line_B**[ANY, ANY, FAILURE] OR **Line_C**[ANY, ANY, FAILURE] OR **Line_D**[ANY, ANY, FAILURE] OR **Buffer**[LEVEL=900] THEN **PS_891**[ALL, ALL, FAILURE], **PS_900**[ALL, ALL, RESTORATION]

Figure 4.13: Advanced operation rules of the complex model in MIRIAM

The last challenge is to control how the system works after emptying the buffer. When one line in the PSA system is down, we use the buffer to replace this line. The buffer can get empty after a while. When this case happens, the system fails to meet the demand at discharge point. However, it is able to provide some hydrogen to the buffer. Hence, we need other advanced operation rules shown in Appendix 4.13 to control this. These rules make sure that all lines are closed when the buffer gets empty at state 3 (Appendix B.9). This causes a shutdown of the system.

The execution time of the complex model in MIRIAM RAM Studio is large. It takes about 36 hours to run a simulation of the PSA with a buffer with a run-length of 50 years. Therefore, we just afford to run 50 replications and gain the average production availability:

$$A_{23}^{*} = 0.999676910 \tag{4.18}$$

Cumulative average value of production availability and its standard error after running 50 replications are shown in Figure 4.14.

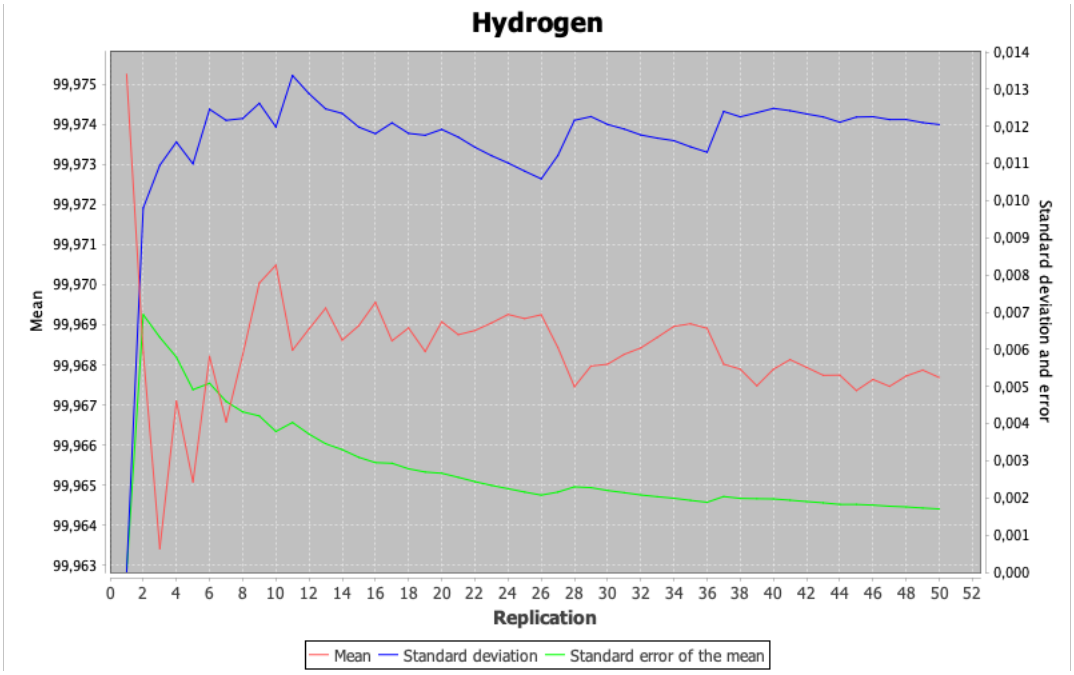


Figure 4.14: Convergence graph of the complex PSA model with a buffer in MIRIAM

Simple model

Simple model of the PSA with a buffer in MIRIAM RAM Studio will not model the 4-minute functional modes to shorten the running time.

We develop the simple model of the PSA with a buffer from the MIRIAM model in the first scenario. From the flow network of the PSA without a buffer, two adjustments are made. We

firstly add a storage unit between the process stage and the discharge point (Figure 4.15). This storage point plays the role of the buffer with capacity of 900 m^3 hydrogen. By establishing the withdraw rate to be 225 m^3 per hour, such capacity is equivalent to 4 hours of continuous withdrawal (Appendix B.11). The buffer is empty at the beginning of the process. To fill up the buffer, the second adjustment is required. Similar to the complex model of the PSA with a buffer, two process stages, PS_891 and PS_900, and three advance operational rules (see 4.16) are added to the simple model. Additionally, demand flow at the discharge point can vary from 891 m^3 per hour to 900 m^3 per hour. Combining this with the advanced operation rules allows the buffer to be refilled when it is empty at perfect state (Appendix B.10). Otherwise, the buffer is not refilled (see Appendix B.12).

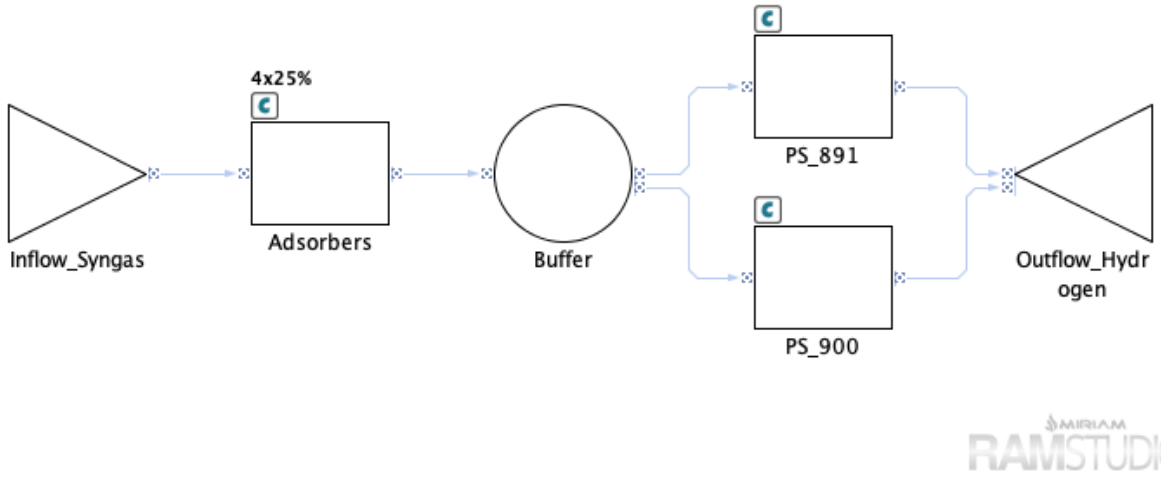


Figure 4.15: Simple model of the 4-bed PSA system with a buffer

Name	Condition	Rule
AOR_891	AND	IF **Adsorbers**[ALL, ALL, FULLFLOW] AND **Buffer**[LEVEL<900] THEN **PS_891**[ALL, ALL, RESTORATION], **PS_900**[ALL, ALL, FAILURE]
AOR_900	OR	IF **Adsorbers**[ANY, ANY, FAILURE] OR **Buffer**[LEVEL=900] THEN **PS_891**[ALL, ALL, FAILURE], **PS_900**[ALL, ALL, RESTORATION]
AOR_state3	OR	IF **Adsorbers**[ANY, ANY, FAILURE] THEN **PS_891**[ALL, ALL, FAILURE], **PS_900**[ALL, ALL, RESTORATION]

Figure 4.16: Advanced operation rules of the simple model in MIRIAM

The main difference between the two models is that we do not simulate the changes in flow after every 4 minutes. Hence, the 4 adsorbers are represented by one process stage instead of 4 different process stages and 4 entry points like in the complex model. Accordingly, simulation time over 50 years is faster. See Figure 4.17 for the convergence graph of 50000 replications. The

average production availability from this run is:

$$A_{23} = 0.999681380 \tag{4.19}$$

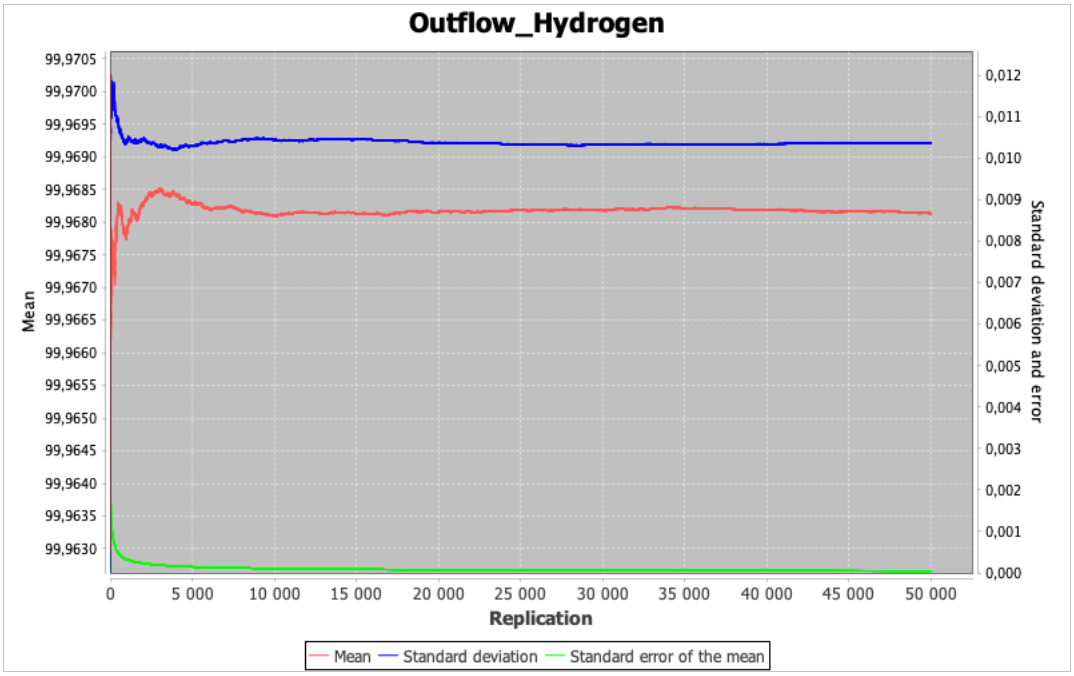


Figure 4.17: Convergence graph of the simple PSA model with a buffer in MIRIAM

Chapter 5

Results and Discussions

In this chapter, findings from the study, especially findings from the case study in Chapter 4, are summarised and discussed.

5.1 Results

In this research, production assurance, applied reliability modelling tools, and technology in blue hydrogen production are studied. Definition of production assurance and how to calculate production availability are reviewed based on two standards, i.e., NORSOK-Z016 (1998) and ISO-20815 (2018), and a number of studies in the literature. Production availability is the ratio of the mean actual production to the planned production over a period of time. Methods used to compute this value fall into two main categories: analytical methods and simulation-based methods. In this master's thesis, three different reliability modelling tools are studied and applied to calculate production availability of a common unit in Steam Methane Reforming (SMR) systems and Autothermal Reforming (ATR) systems.

In the case study, three modelling tools are employed to model a 4-bed pressure swing adsorption (PSA) in blue hydrogen plants. These modelling tools include Markov process, discrete event simulation in Python, and MIRIAM RAM Studio. The aim is to examine the flexibility of the three tools in terms of modelling and calculating production availability of the PSA with(out) a buffer. The 4-minute functional modes of the PSA and the refilling/withdrawing policy of the buffer are two main challenges to test such flexibility.

Eight models of the 4-bed PSA are constructed in this work. They are divided into two categories: PSA without buffer (3 models) and PSA with a buffer (5 models). With regards to models of the PSA with a buffer, we differentiate between complex models and simple ones. The former refers to the hybrid models of the PSA with a buffer where the 4-minute operational modes are simulated. Whereas, the latter skips these modes. There are two complex models made by discrete event simulation in Python and MIRIAM RAM Studio. The only model in Markov process

does not simulate the 4-minute functional modes, so it is considered as a simple model. Production availability is then computed from different reliability models of the PSA. As production availability values are close to 1, complement to 1 of production availability is calculated. The results are shown by a bar chart in Figure 5.1. What stands out in this chart is that compliment to 1 of production availability from models of the PSA with a buffer is smaller than that value in models without a buffer. This interesting result is further discussed in the next section.

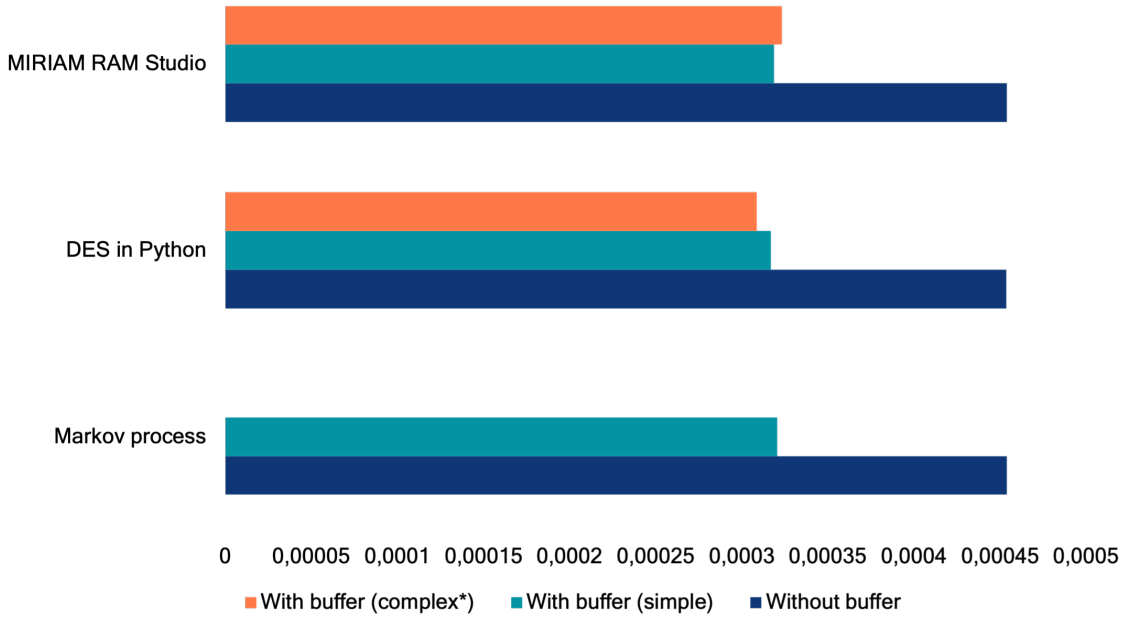


Figure 5.1: Complement to 1 of production availability from different models of the 4-bed PSA. The asterisk sign (*) means that complex models in MIRIAM and Python are run with a 50 replications; other simulation models are run with 50000 replications.

5.2 Discussions

Discussions in this section centre on the flexibility of three reliability modelling tools in modelling a complex and dynamic system. Prior studies have noted the less flexibility of Markov processes in modelling non-exponential distributions. Very little was found in the literature on the question of such flexibility in MIRIAM RAM Studio and discrete event simulation in Python. Findings from this study reveal some supporting points.

In the first scenario where the PSA without a buffer is modelled, Markov process, discrete event simulation with SimPy in Python, and MIRIAM RAM Studio seem comparable. They all successfully model the PSA system without a buffer and give similar results as shown in Figure 5.1. This could be attributed to the simplicity of the PSA system without a buffer. More specifi-

cally, without the buffer, production rate of the system can be either 900 m^3 hydrogen per hour at perfect state or zero otherwise. Two types of event affecting the hydrogen production's continuity are failure and reparation of valves in the 4-bed PSA unit. Time to failure and time to repair are stochastic variables and independent of the 4-minute functional modes. Hence, the 4-minute operational modes of the PSA can be ignored. This makes the 50000 replications in both MIRIAM RAM Studio and Python run quite fast. It is noticed in Figures 4.4 and 4.7 that these average values of production availability are stable after around 20000 replications with flat cumulative mean lines. To some extent, the mean values after 50000 replications are reliable. Also, there is minor difference between the average production availability from these runs and production availability from steady states in Markov model (see dark blue bars in Figure 5.1). Thus, three modelling tools does not differ from each other in terms of modelling simple systems like the PSA without a buffer.

When the system gets more complex in the second scenario (PSA with a buffer), three modelling tools under study show different capabilities.

Consistent with the literature, this research found that the regular Markov process is less flexible. Markov processes are limited to only exponential distributions while the capacity of the buffer is a deterministic variable. The buffer's level which is a continuous variable is also a challenge for discrete-state Markov processes. Plus, there is a dependability of the buffer's refilling and withdrawing sessions on both level of the buffer, previous system states, and the past 4-minute operational modes of the PSA. As a result, hydrogen production rate at the output, refilling rate, and withdrawing rate at the buffer keep changing over time. Hence, time to fill up the buffer and time to empty it cannot be exactly expressed by a specific exponential distribution.

To cope with the dynamic system of the PSA model with a buffer in Markov process, necessary assumptions and approximations are needed. We first assume that the buffer is always empty when we start refilling it; and it is always full when we start withdrawing from it. Put it simply, the buffer is either full or empty; no intermediate levels of the buffer are allowed. This is in fact not the real-life property of the buffer. Yet, it is reasonable in case we want to simplify the situation such that Markov process can be utilized to approximately model a dynamic system. More importantly, the assumptions make it possible to employ Semi-Markov process. In the Semi-Markov process, there are two deterministic distributions which are the 4-hour capacity of the buffer and 100-hour refilling time from the zero level. These deterministic distributions are further approximated by 5-stage Erlang distributions, turning the model into a Markov process. By doing this, intermediate-level states of the buffer are now introduced, allowing more states (real levels) of the buffer to be included into the model. In general, without necessary assumptions and the k-stage Erlang distribution, Markov process is rather limited in capturing real-life behaviors of the buffer.

The difference between production availability stemmed from Semi-Markov process utiliz-

ing k-stage Erlang distribution and production availability values from simple models in other tools is subtle (See light blue bars in Figure 5.1). A possible explanation for this could be that the PSA with a buffer is simplified and handled in the same way in these models. For example, the 4-minute operational modes are not modelled, refilling takes place only at perfect stage, refill rate and withdraw rate are constant over time, and so forth. The term "simple" itself already reflects the simplicity in these models.

Unlike Markov process, MIRIAM RAM Studio and discrete event simulation in Python are more flexible in modelling the 4-bed PSA with a buffer. They can model the 4-minute operational modes and stick to the refilling and withdrawing rules of the buffer. In Python, we can execute the 4-minute operational modes with a "while" loop. Inside this loop, we calculate the production and handle the buffer using user-defined functions. In MIRIAM RAM Studio, options for controlling throughput, capacity, and flow are integrated and available in established forms. Users just need to input data, make the rules, and let the flow algorithm do the rest. One advantage of MIRIAM is the flexibility in defining flow and capacity of all items in the flow network. Flow and capacity could be defined as constant or changeable according to calendar/ functions defined by users.

Amongst possible features, advanced operation rules in MIRIAM are powerful and they successfully fulfill the refilling/withdrawing rules of the buffer. Thanks to operation rules, the flow towards the hydrogen discharge point is changeable: 891 m^3 per hour when refilling the buffer at state 4 and 900 m^3 per hour when buffer is full or system is at state 3. Without these advanced operation rules, we have to face with production lost because the demand at discharge point is fixed at value of 891 to facilitate the refilling activity. Accordingly, we never reach 900 m^3 hydrogen per hour at the output even though the buffer is full. Not to mention that if demand at discharge point stays constant at 891 m^3 per hour, production availability value will be of the scale compared to that value from other tools. The reason is that in MIRIAM RAM Studio, the reference level for production availability is defined by the demand at discharge point. If the demand is higher than the capacity of the system, production availability can never reach 100% even the system produces full capacity all the time. In our case, without advanced operation rules, the demand at discharge point is smaller than the capacity ($891 < 900$). Hence, the corresponding total production or production availability is not exactly comparable to other cases where expected production is computed based on production rate of 900 m^3 per hour. Shortly, advanced operation rules make MIRIAM RAM Studio flexible and comparable to discrete event simulation in Python.

The executive time of complex models of the PSA with a buffer in MIRIAM RAM Studio and Python is however large, larger than that of simple models. It takes around two minutes to finish one simulation of complex model with a run-length of 50 years in Python, and that number in MIRIAM is more than 36 hours. In Python, long running time could be explained by multiple

"if" statements in the "while" loop and their repetition since we simulate the "while" loop with a 4-minute time step. In MIRIAM RAM Studio, the long running time could have been generated by the time-driven model and the number of various constraints, objectives, and advanced operation rules that we define in the software to acquire the desired model. The flow algorithm in MIRIAM is called whenever a change happens in the model. In the complex model of the PSA with a buffer, changes take place after every 4 minutes over a 50-year run-length. For these reasons, it is time-demanding to run simulations of such complex model. However, the simulation report from MIRIAM shows that the buffer is full most of the time over 50 years. This means that if the buffer is full and no failures take place, equations (constraints) which the flow algorithm has to solve will regularly repeat themselves. Recommendation for improvement associated to this matter is proposed in the next chapter. Due to the time-consuming property of the complex model, we can afford to run only 50 replications in both MIRIAM and Python. The data from these limited samples unfortunately cannot represent for production availability at steady state, which is indicated by fluctuated mean values in Figures 4.10 and 4.14. Figure 4.14 also indicates that standard error of the mean after 50 replications is not approaching 0 yet and higher than that of 50000 replications. In other words, with such small sample size, caution must be applied. In contrast, the simple models of the PSA with a buffer are less time-consuming than the complex models in both MIRIAM RAM Studio and Python. Short execution time makes it possible to run 50000 replications in both tools. Figures 4.11 and 4.17 illustrate that average production availability reaches stable state after around 20000 replications.

From an overall perspective, the complex models are expected to give higher production availability values than other models (Figure 5.1). The reason is that beside refilling the buffer at state 4 like the simple models, we also fill up the buffer at state 3 with "cleaning" hydrogen in complex models (see refilling policy in Section 4.5). Refilling rate in this case can be either 300 or 600 m^3 hydrogen per hour, which is significantly higher than that in state 4, i.e., 9 m^3 hydrogen per hour. Even though refilling with "cleaning" hydrogen takes place in short time, that is 4 minutes each session, it helps to prolong the time using the buffer. Arguably, it should slightly increase the production availability of the system in general. Nevertheless, such increase in production availability is only observed in the complex model in Python, not the complex model in MIRIAM (see orange bar in Figure 5.1). This could be firstly attributed to the small sample size of simulations, i.e., 50, as discussed above. Secondly, in Python, we allow the "cleaning" hydrogen to be sent to the production if the buffer is full at state 3 according to the refilling policy of the buffer. This means that the production rate could be larger than 900 m^3 per hour. Notice that the possibility of this case is rather small because at state 3, we withdraw from the buffer more than we put in, meaning that the buffer rarely gets full during this situation. Still, it is possible, especially if the system jumps from state 4^F to state 3^F, and vice verse, when the failed adsorber is performing mode E1 R or D P. Whereas, in MIRIAM, the maximum hydrogen

demand is 900 m^3 per hour, meaning that the flow algorithm will adjust the flow within the network to just meet that demand, even we can produce more than the defined demand flow.

An analysis of the bar chart in Figure 5.1 reveals that installing a buffer to the 4-bed PSA system helps to increase the production availability. Either simple models or complex models of the PSA with a buffer return higher production availability (smaller complement to 1) than that number in the PSA model without a buffer. However, it is uncertain to say that it is more economical to install a buffer. There are two reasons. Firstly, in this study, valves connected to the buffer are assumed to work perfectly all the time. This is controversial when we consider that valves connected between adsorbers can fail. Secondly, capital expenditures (CapEx) and operating expenses (OpEx) are not covered in this work. Investment and operation cost of the buffer can be high and overkill the improvement in production availability. Suggestion on how to improve this matter is proposed in the last chapter.

Chapter 6

Conclusions and Recommendations for Further Work

In this chapter, some conclusions from the research are drawn. More work and ideals are introduced in the end to plan for future work.

6.1 Conclusions

The aim of the present research was to perform a comparison of three different modelling tools by applying them to model and calculate production availability of a 4-bed PSA system. Taking the literature review and the case study into consideration, we arrive at the following conclusions.

The study gives an account of what production assurance is and how to compute production availability of a system. Production assurance refers to activities performed to achieve and maintain optimal performance of a system. In doing this, production availability is required. Two main approaches to accessing production availability are analytical methods and simulation-based methods. Markov process representing the analytical methods is scrutinized in this study. Then, a generalisation of the Markov process, Semi-Markov process is introduced to handle non-Markovian models. In terms of simulation-based methods, discrete event simulation in Python and MIRIAM RAM Studio are employed. There have been few previous attempts to compare modelling tools in the analytical approach with simulation-based method, but not specifically three tools in this study.

This study set out to test the flexibility of Markov process, Python, and MIRIAM RAM Studio in modelling a 4-bed PSA unit in blue hydrogen production plants. To that end, a buffer is added to the PSA system. The results from the case study reveal that Markov process is less flexible than MIRIAM and discrete event simulation in Python. Modelling the buffer requires that the modelling tool can handle non-exponential distributions and time-driven systems which is a limi-

tation of event-driven Markov modelling. To overcome this challenge, we utilize Semi-Markov process, k-stage Erlang distribution along with several assumptions. The resulting model shows the capability to handle multi-state systems. However, it is limited in capturing dynamic behaviours such as the 4-minute operational modes and advanced refilling and withdrawing activities of the buffer.

The findings reported in this study show that MIRIAM RAM Studio and discrete event simulation in Python have the ability to model highly complex and dynamic systems, but the running time is significant. The complex models of the PSA with a buffer indicate that both tools can handle the 4-minute operational modes of the PSA and fulfill all refilling/withdrawing rules of the buffer. In Python, SimPy is a useful library in discrete event simulation when it provides methods to control when the events are triggered and how resources in the process are treated. However, it does require a certain computational skill of the programmers to create necessary user-defined functions for operational rules. MIRIAM RAM Studio, on the other hand, is user-friendly when features needed to model complex systems are available in the software. For instant, advanced operation rules in MIRIAM can be made with few clicks, which helps to save time compared to coding in Python. The flow algorithm in MIRIAM can solve many constraints related to calendar, flow, and capacity. Yet, it is time-consuming to complete a run-length of several decades when constraints change on numerous occasions, for example, every 4 minutes of the run-length.

6.2 Recommendations for Further Work

Based on the work that has been completed in this study, possible extensions and recommendations for future work are defined.

First of all, we suggest a study similar to this one should be carried out on more modelling tools, e.g., BlockSim and AltaRica 3.0, in the future. We had an attempt to employ AltaRica 3.0 in this study but it did not work out due to some disagreements. We believe that challenges in modelling the 4-bed PSA system with a buffer can help to reveal advantages and disadvantages of these tools, and improvements can be made to achieve better performance.

Secondly, further studies also need to determine uncertainty in production availability calculation of the PSA. In this research, reliability data are collected from OREDA without further information about uncertainties. The number of replications in simulation-based methods are also determined based on a simple graphical method without providing any measured precision level. More information on uncertainty in production availability results of the PSA would help us to establish a greater degree of accuracy on this study.

Thirdly, the capacity and refilling/withdrawing policy of the buffer could be a starting point for a sensitivity analysis. A large capacity of the buffer can increase the availability but it does

not mean that production availability of the system is increased correspondingly. Repair time might have an important role to play in this issue. In this master thesis, the characteristics of the buffer are defined subjectively. So, an optimization problem can be established in further investigation to maximize the usage of the buffer.

Similarly, variable k in the k-stage Erlang distribution is subjectively chosen to be equal to 5. We mainly aimed to not make the Markov diagram (number of system states) too big. However, with the help from computer-based tools, it might not be a challenge if we increase the variable k in the Erlang approximation to, for example, 100 in future studies. Larger k means that more intermediate levels of the buffer are included. This could help to better approximate the deterministic distribution related to the buffer in Semi-Markov process.

Furthermore, research in life cycle cost of the PSA with a buffer is an essential next step in confirming the benefit of the buffer. Considering product price, operation cost, ect, can be helpful in making decisions involved in installation of the buffer. MIRIAM RAM Studio could be further studied in this case since cost optimization is one of the problems that the software focuses on.

The challenge is now to reduce the running time in MIRIAM RAM Studio when flow changes too often in time-driven models. The flow algorithm might solve the same set of equations multiple times if the flow changes in sequential and repetitive order. To save running time, one reasonable approach could be to retain repeated constraints and reuse the corresponding set of flows while running simulations of the same model.

Finally, the study in the 4-bed PSA can be further expanded for multi-bed PSA systems such as 6-bed and 10-bed PSA systems. The dynamic operation increases when the number of beds in the PSA unit increases. Therefore, this would be a fruitful area for further work.

Appendix A

Acronyms

RAMS Reliability, availability, maintainability, and safety

RAM Reliability, availability, and maintainability

PSA Pressure swing adsorption

GHG Greenhouse gas

SMR Steam methane reforming

ATR Autothermal reforming

GHR Gas heated reformer

GSR Gas switching reforming

WGS Water-gas shift

CCS Carbon capture and storage

OREDA Offshore and onshore reliability data project

LCC Life cycle cost

CTMC Continuous-time Markov chain

CVDS Continuous-variable dynamic system

DET Deterministic distribution

SMP Semi-Markov process

MRP Markov renewal process

DES Discrete event simulation

SEL Scheduled event list

TTF Time to failure

MTTF Mean time to failure

MTBF Mean time between failure

TTR Time to repair

MTTR Mean time to repair

RBD Reliability block diagram

FTA Fault tree analysis

FMECA Failure mode, effects & criticality analysis

PDC Probability distribution of throughput capacity

ADS High-pressure adsorption

E1 R Pressure equalisation and repressurisation

D P Countercurrent blowdown and purge

E1 PP Pressure equalisation and providing purge

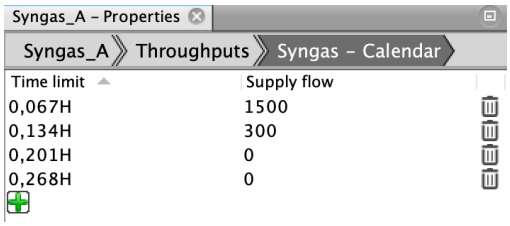
4-o-o-4 Four out of four

AOR Advanced operation rule

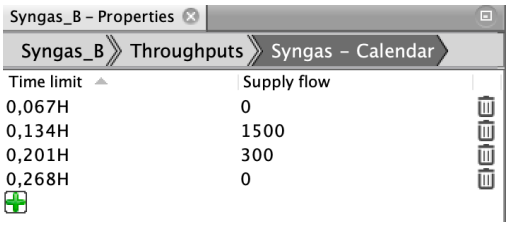
Appendix B

Figures in MIRIAM RAM Studio

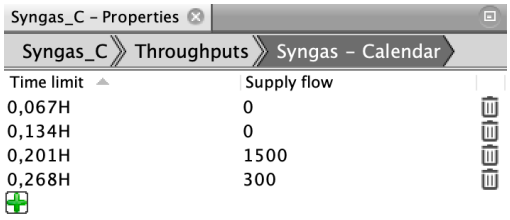
B.1 Complex Model of PSA With a Buffer



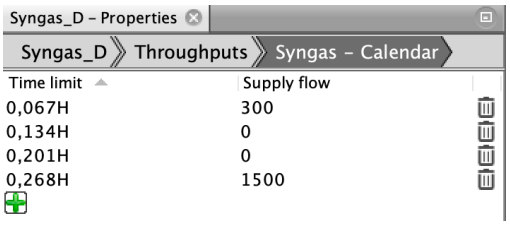
(a) Line A



(b) Line B

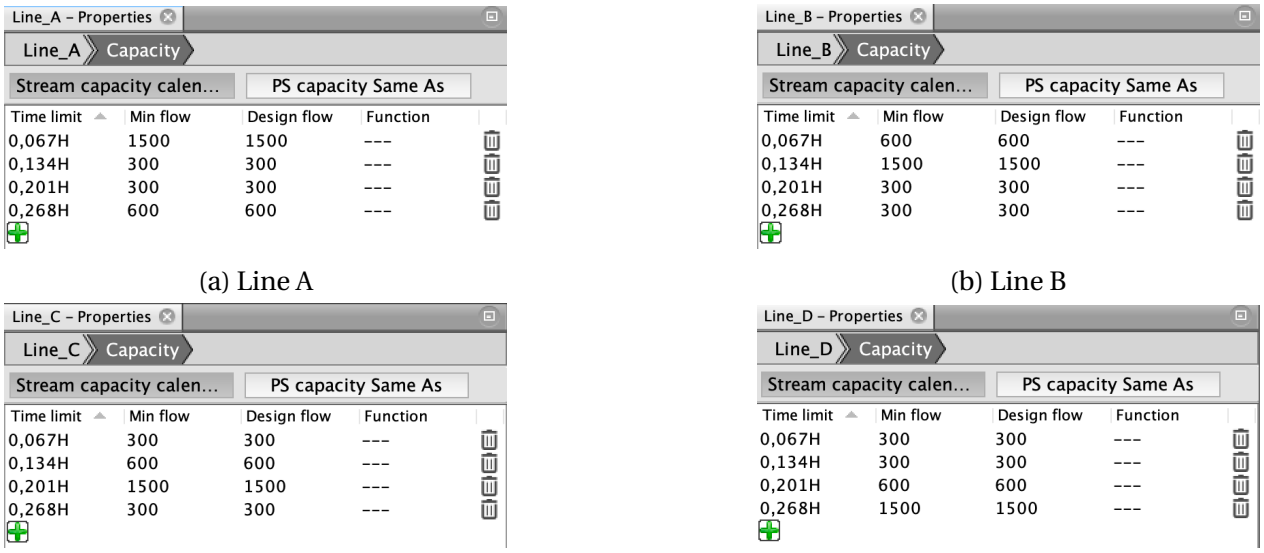


(c) Line C



(d) Line D

Figure B.1: Syngas supply flow calendar



(a) Line A

(b) Line B

(c) Line C

(d) Line D

Figure B.2: Stream capacity calendar

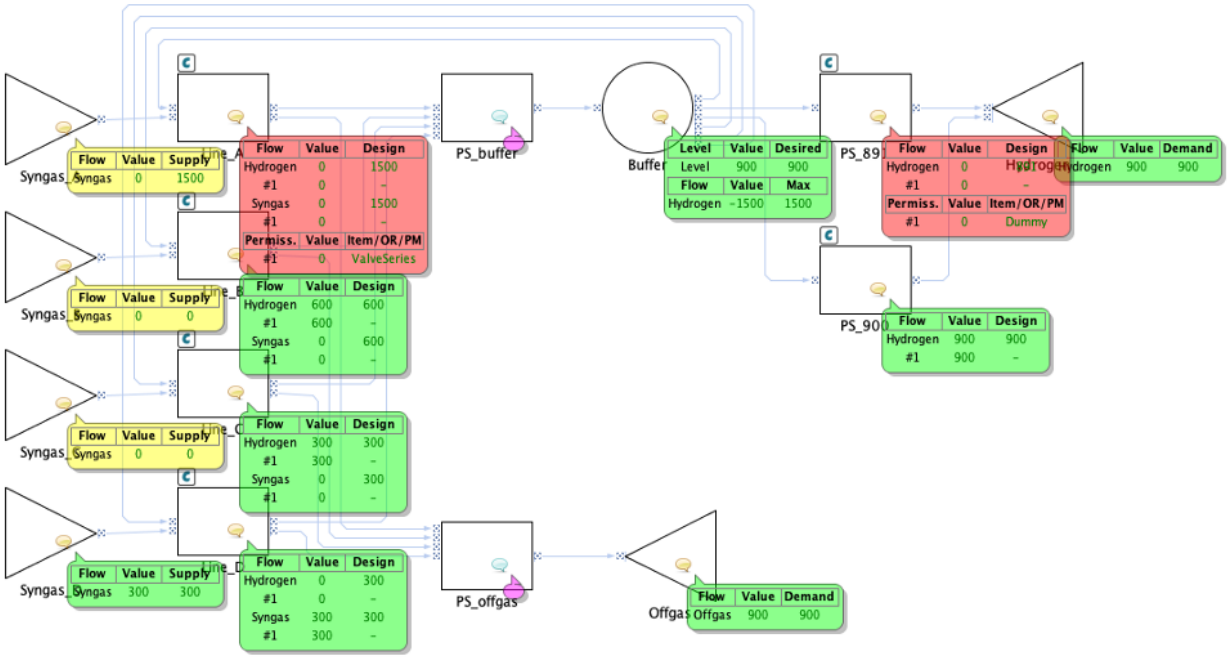


Figure B.3: Withdraw rate of 1500 in complex model of the PSA with a buffer

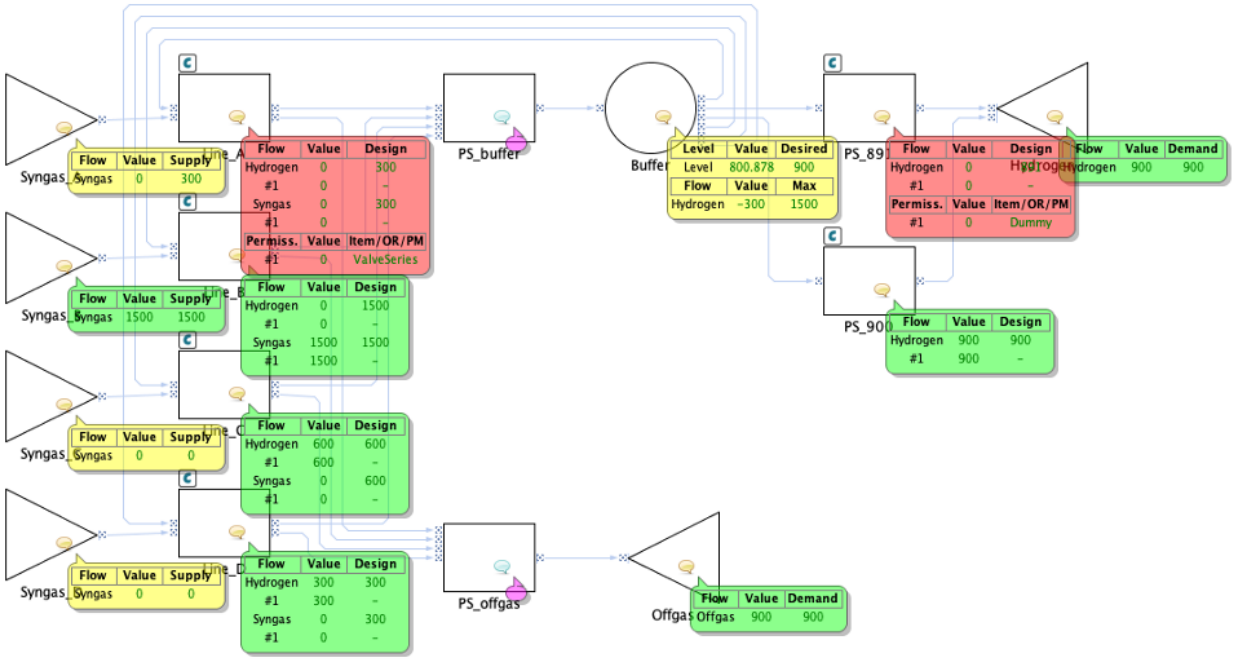


Figure B.4: Withdraw rate of 300 in complex model of the PSA with a buffer

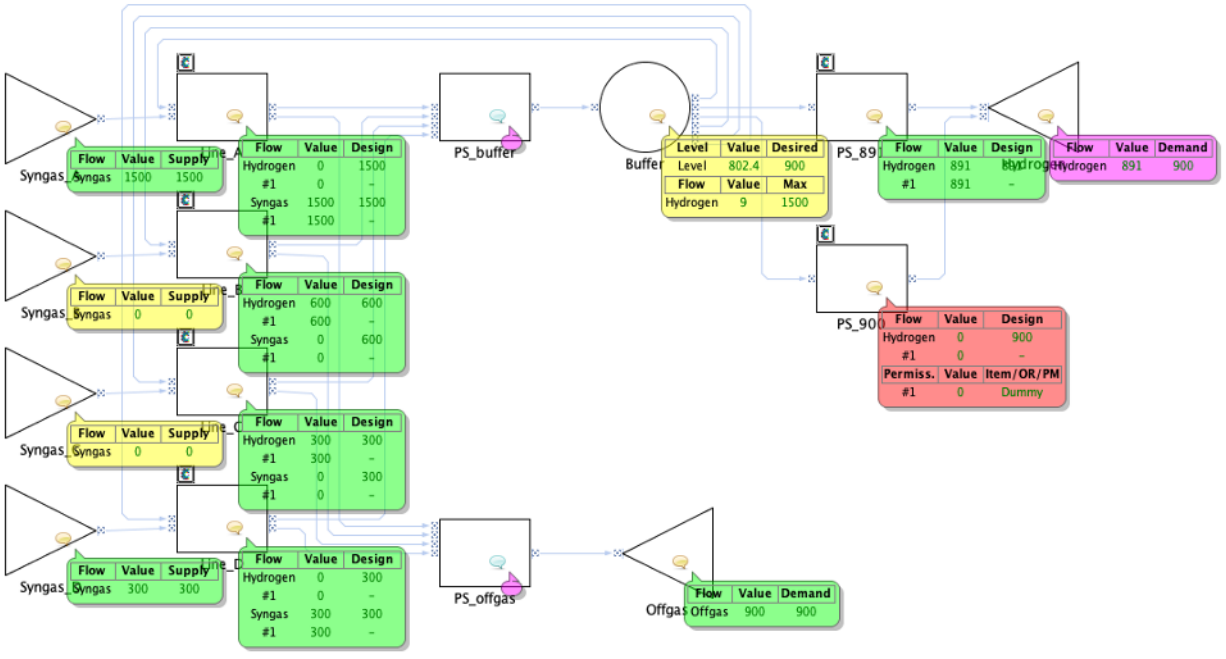


Figure B.5: Refill rate of 9 in complex model of the PSA with a buffer at perfect state

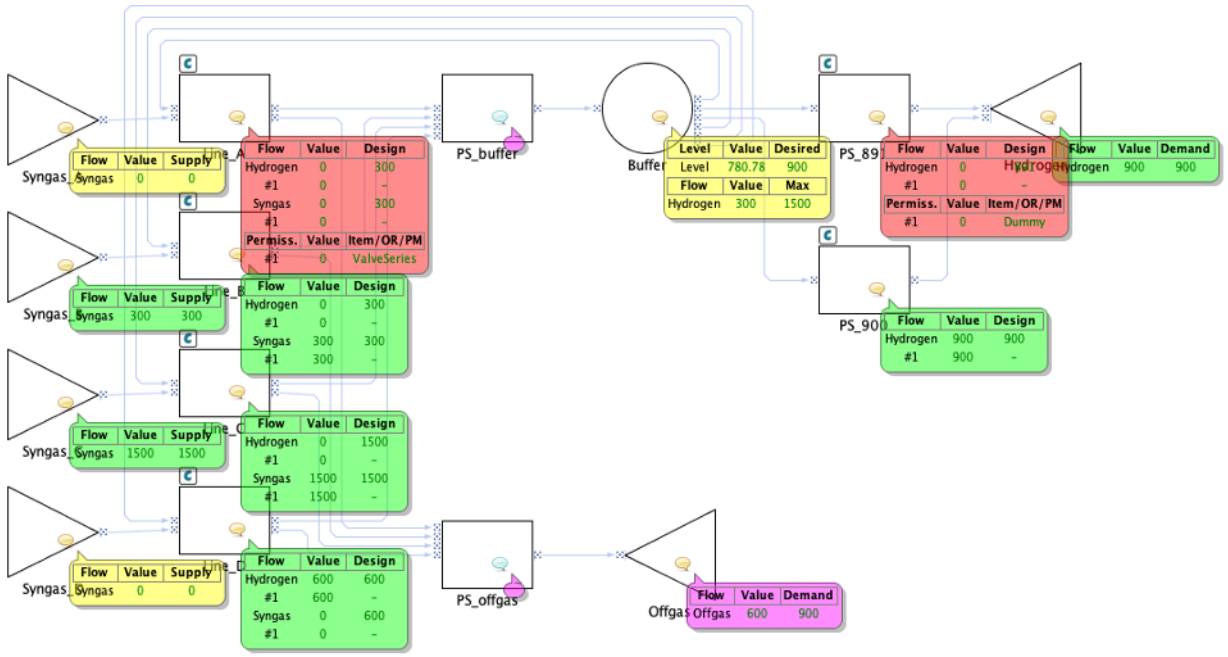


Figure B.6: Refill rate of 300 in complex model of the PSA with a buffer

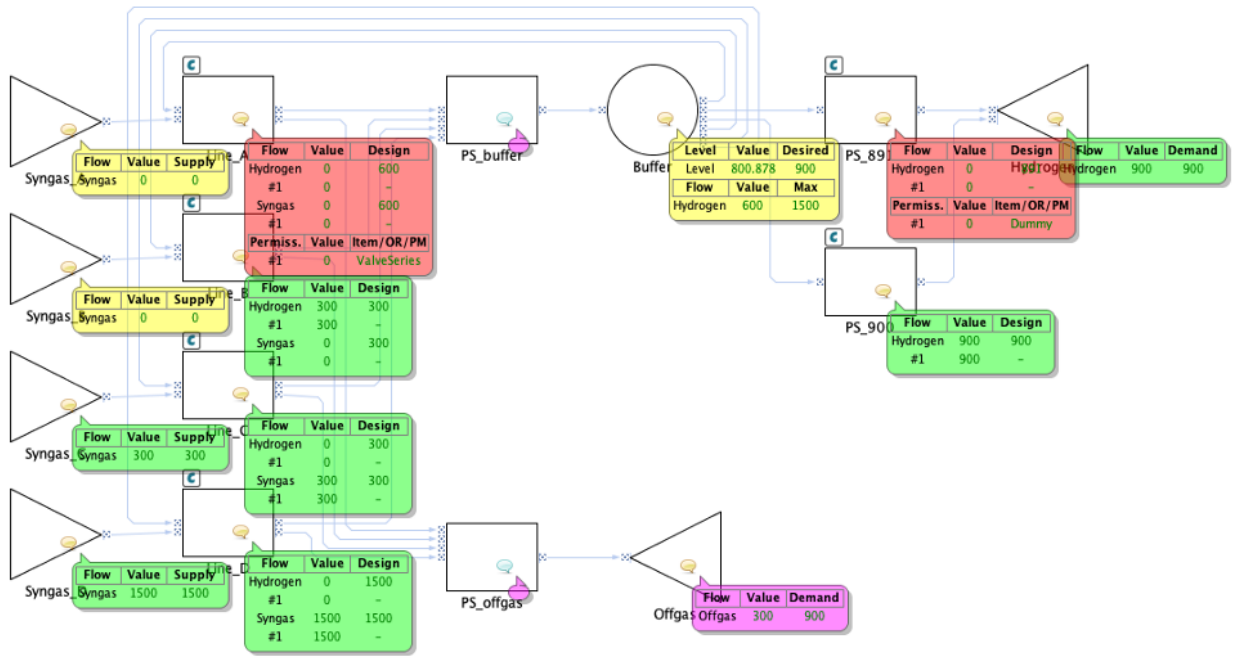


Figure B.7: Refill rate of 600 in complex model of the PSA with a buffer

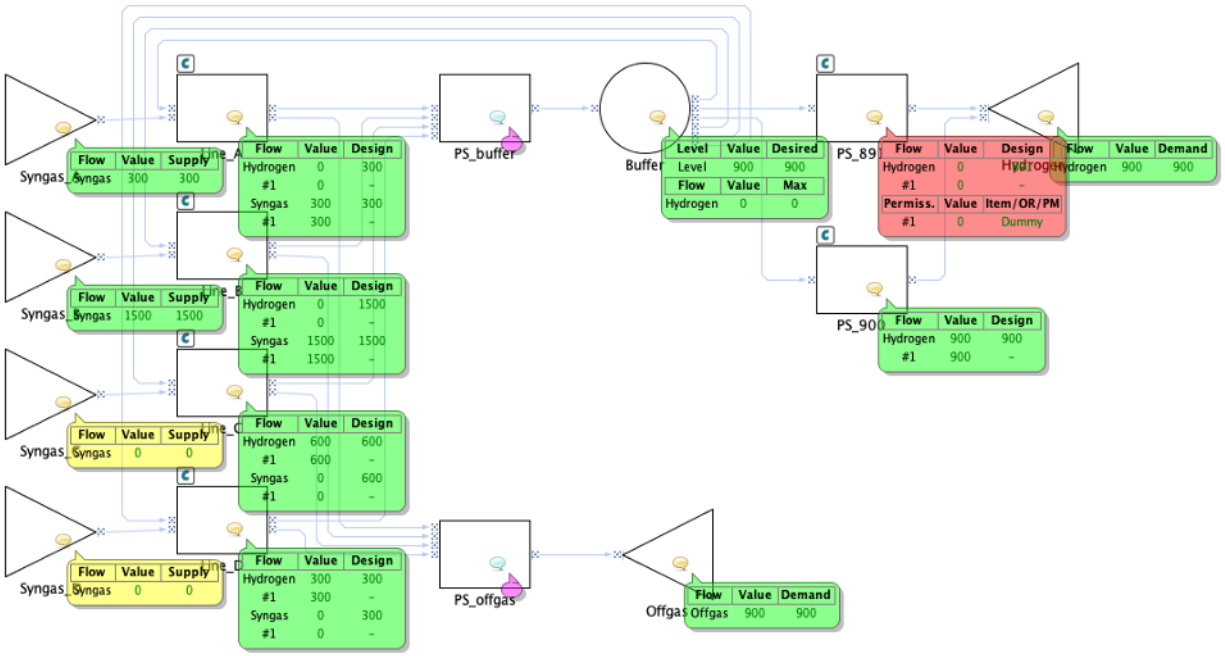


Figure B.8: Stop refilling when buffer is full, advanced operation rules are applied

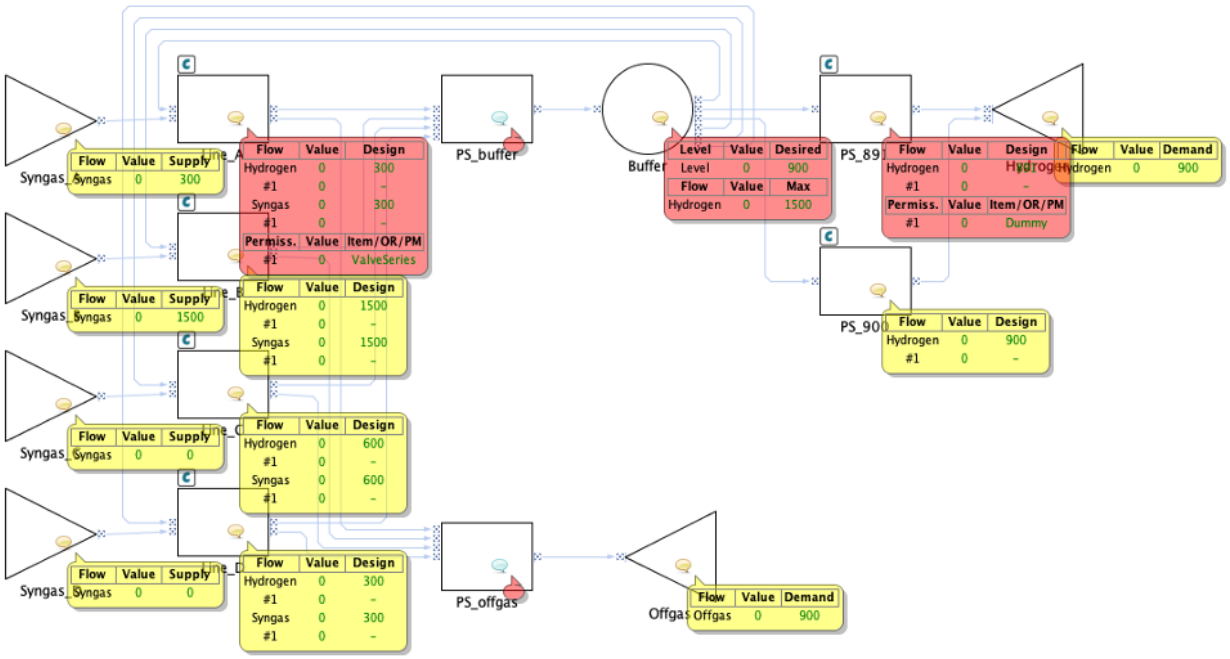


Figure B.9: System is shutdown when buffer gets empty at state 3

B.2 Simple Model of PSA With a Buffer

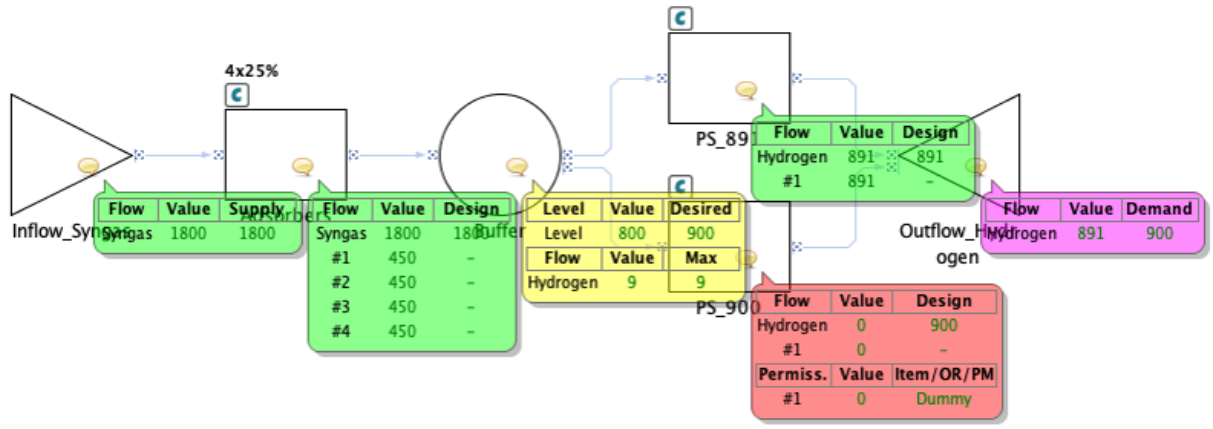


Figure B.10: Refill rate of 9 in simple model of the PSA with a buffer at perfect state

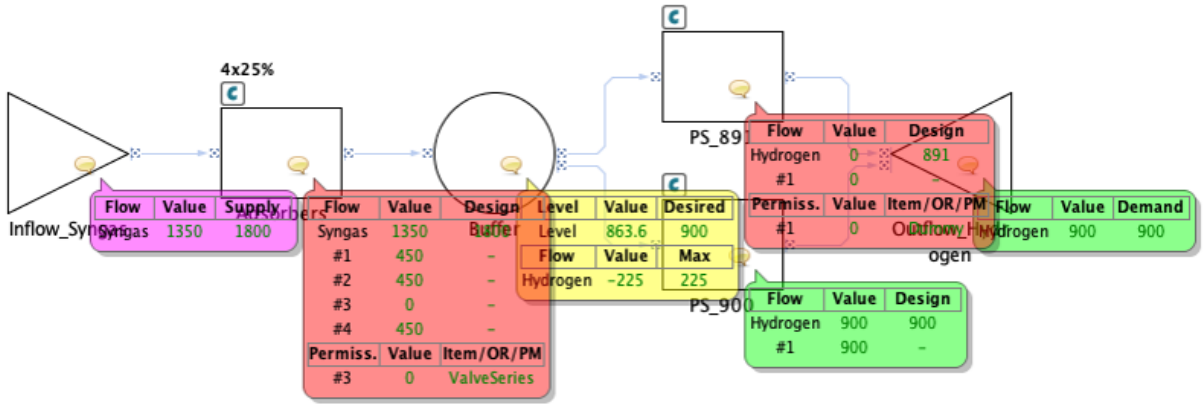


Figure B.11: Withdraw rate of 225 in simple model of the PSA with a buffer

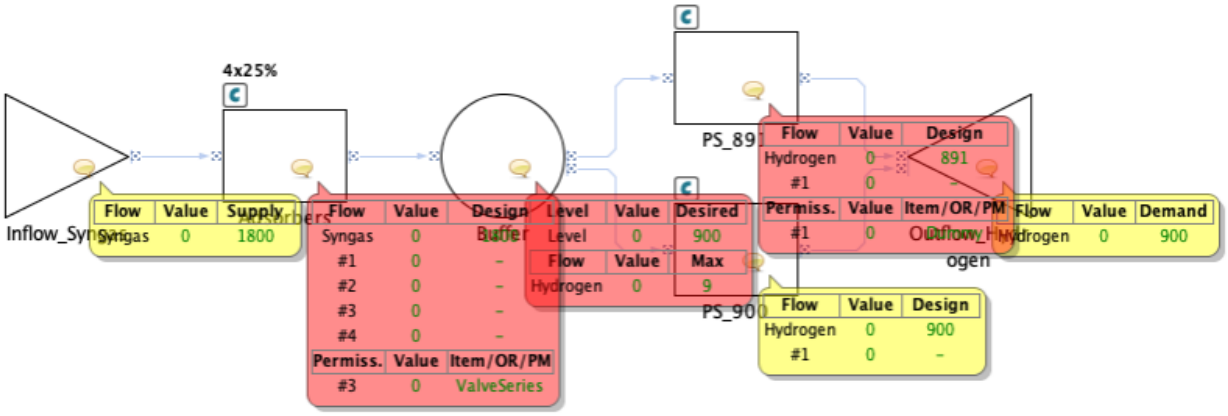


Figure B.12: System is shutdown when buffer gets empty at state 3

Appendix C

Python Code in Case Study

C.1 Markov Model

C.1.1 Time Dependent Solution in The First Scenario

```
from numpy import *
import numpy as np

lamb = 1.49e-5    # Failure rate
mu = 1/7.615      # Repair rate

# Transition rate
A = array ([[ -mu, mu, 0, 0, 0] ,
[lamb, -(lamb+mu), mu, 0, 0 ] ,
[0, 2*lamb, -(2*lamb+mu), mu, 0 ] ,
[0, 0, 3*lamb, -(3*lamb+mu), mu ] ,
[0, 0, 0, 4*lamb, -4*lamb]])

I = np.identity(5)
P = array ([0 , 0 , 0 , 0 , 1]) # Initial state
dt = 1
SIM_TIME = 8760*50            # 50 year simulation

for i in range(dt, SIM_TIME+dt, dt) :
    P = np.dot(P, I + A*dt)
print(P)
```

C.1.2 Steady-state Solution in The Second Scenario

```
import numpy as np
lamb = 1.49e-5
lambL = 1.25
lambT = 0.05
mu = 1/7.615
A = np.array([
    [1, lambT, 0, 0, 0, 0, 4*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #4^E
    [1, -(lambT+4*lamb), lambT, 0, 0, 0, 0, 4*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #4^20
    [1, 0, -(lambT+4*lamb), lambT, 0, 0, 0, 0, 4*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #4^40
    [1, 0, 0, -(lambT+4*lamb), lambT, 0, 0, 0, 0, 4*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #4^60
    [1, 0, 0, 0, -(lambT+4*lamb), lambT, 0, 0, 0, 0, 4*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #4^80
    [1, 0, 0, 0, 0, -(4*lamb), 0, 0, 0, 0, 0, 4*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #4^F

    [1, 0, 0, 0, 0, 0, -(mu+3*lamb), 0, 0, 0, 0, 0, 3*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #3^E
    [1, mu, 0, 0, 0, 0, lambL, -(mu+3*lamb+lambL), 0, 0, 0, 0, 0, 3*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #3^20
    [1, 0, mu, 0, 0, 0, 0, lambL, -(mu+3*lamb+lambL), 0, 0, 0, 0, 0, 3*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #3^40
    [1, 0, 0, mu, 0, 0, 0, 0, lambL, -(mu+3*lamb+lambL), 0, 0, 0, 0, 0, 3*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #3^60
    [1, 0, 0, 0, mu, 0, 0, 0, 0, lambL, -(mu+3*lamb+lambL), 0, 0, 0, 0, 0, 3*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #3^80
    [1, 0, 0, 0, 0, mu, 0, 0, 0, 0, lambL, -(mu+3*lamb+lambL), 0, 0, 0, 0, 0, 3*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #3^F

    [1, 0, 0, 0, 0, 0, mu, 0, 0, 0, 0, 0, -(mu+2*lamb), 0, 0, 0, 0, 0, 2*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #2^E
    [1, 0, 0, 0, 0, 0, 0, mu, 0, 0, 0, 0, 0, -(mu+2*lamb), 0, 0, 0, 0, 0, 2*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #2^20
    [1, 0, 0, 0, 0, 0, 0, 0, mu, 0, 0, 0, 0, 0, -(mu+2*lamb), 0, 0, 0, 0, 0, 2*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #2^40
    [1, 0, 0, 0, 0, 0, 0, 0, 0, mu, 0, 0, 0, 0, 0, -(mu+2*lamb), 0, 0, 0, 0, 0, 2*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #2^60
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, mu, 0, 0, 0, 0, 0, -(mu+2*lamb), 0, 0, 0, 0, 0, 2*lamb, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #2^80
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, mu, 0, 0, 0, 0, 0, -(mu+2*lamb), 0,
```

```
    0, 0, 0, 0, 2*lamb, 0, 0, 0, 0, 0, 0], #2^F

    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, mu, 0, 0, 0, 0, 0, -(mu+lamb), 0,
    0, 0, 0, 0, lamb, 0, 0, 0, 0, 0], #1^E
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, mu, 0, 0, 0, 0, 0, -(mu+lamb),
    0, 0, 0, 0, 0, lamb, 0, 0, 0, 0], #1^20
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, mu, 0, 0, 0, 0, 0, -(mu+
    lamb), 0, 0, 0, 0, 0, lamb, 0, 0, 0], #1^40
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, mu, 0, 0, 0, 0, 0, -(mu+
    lamb), 0, 0, 0, 0, 0, lamb, 0, 0], #1^60
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, mu, 0, 0, 0, 0, 0, -(
    mu+lamb), 0, 0, 0, 0, 0, lamb, 0], #1^80
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, mu, 0, 0, 0, 0, 0,
    -(mu+lamb), 0, 0, 0, 0, 0, lamb], #1^F

    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, mu, 0, 0, 0, 0,
    0, -(mu), 0, 0, 0, 0, 0], #0^E
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, mu, 0, 0, 0,
    0, 0, -(mu), 0, 0, 0, 0], #0^20
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, mu, 0, 0,
    0, 0, 0, -(mu), 0, 0, 0], #0^40
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, mu, 0,
    0, 0, 0, 0, -(mu), 0, 0], #0^60
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, mu,
    0, 0, 0, 0, 0, -(mu), 0], #0^80
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    mu, 0, 0, 0, 0, 0, -(mu)]  #0^F
])
B = np.array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
x = np.matmul(B, np.linalg.inv(A))
print(x)
print(f"Production Availability: {(x[0]+x[1]+x[2]+x[3]+x[4])*0.99 + (x[5]+
    x[7]+x[8]+x[9]+x[10]+x[11])*1}")
```

C.2 Discrete Event Simulation in SimPy

C.2.1 PSA Model Without a Buffer

```
"""
Scenario: PSA without buffer
    1: Simulate individual adsorbers based on failure and repair rates
    2: Apply configuration rules to calculate production availability
"""

!pip install simpy
import simpy
import matplotlib.pyplot as plt
from statistics import mean
import random
import numpy as np

SIM_TIME = 8760*50

class Line(object):
    def __init__(self, env, name, repairman):
        self.env = env
        self.name = name
        self.time = [0]   # Record the time
        self.mode = [1]   # Record the state, 1 = UP, 0 = DOWN

        self.process = env.process(self.working(repairman))

    def working(self, repairman):
        while True:
            yield self.env.timeout(time_to_failure())  # Generate falure
   events
            t_broken = env.now
            self.time.extend([t_broken])
            self.mode.extend([0])

            with repairman.request() as req:    # Generate repair events
                yield req
                yield self.env.timeout(time_to_repair())
                t_repair = env.now
                self.time.extend([t_repair])
                self.mode.extend([1])
                t_down = t_repair - t_broken

def time_to_failure():
```

```
    return random.expovariate(1.49e-5)

def time_to_repair():
    return random.expovariate(1/7.615)

"""
    From here to the end, calculate production availability based on
    adsorber profiles
"""

def calculation(LineA, LineB, LineC, LineD, SIM_TIME,):

    last_mode = min(LineA.mode[-1],LineB.mode[-1],LineC.mode[-1],LineD.
    mode[-1])

    # Create and customise component profiles
    timeA = LineA.time
    timeA.pop(0)
    timeA.append(SIM_TIME)
    modeA = LineA.mode
    modeA.append(LineA.mode[-1])
    modeA.pop(0)

    timeB = LineB.time
    timeB.pop(0)
    timeB.append(SIM_TIME)
    modeB = LineB.mode
    modeB.append(LineB.mode[-1])
    modeB.pop(0)

    timeC = LineC.time
    timeC.pop(0)
    timeC.append(SIM_TIME)
    modeC = LineC.mode
    modeC.append(LineC.mode[-1])
    modeC.pop(0)

    timeD = LineD.time
    timeD.pop(0)
    timeD.append(SIM_TIME)
    modeD = LineD.mode
    modeD.append(LineD.mode[-1])
    modeD.pop(0)


```

```
# Create system profile based on system configuration
    time = [0]
    mode = [1]
    count_0 = 0
    while not (len(timeA)==1 and len(timeB)==1 and len(timeC)==1 and len(timeD)==1):
        x = min(min(timeA),min(timeB), min(timeC), min(timeD))
        if timeA.count(x)>0:
            ind = timeA.index(x)
            y = modeA[ind]
            if y==0:
                count_0 += 1
            timeA.pop(0)
            modeA.pop(0)
        elif timeB.count(x)>0:
            ind = timeB.index(x)
            y = modeB[ind]
            if y==0:
                count_0 += 1
            timeB.pop(0)
            modeB.pop(0)
        elif timeC.count(x)>0:
            ind = timeC.index(x)
            y = modeC[ind]
            if y==0:
                count_0 += 1
            timeC.pop(0)
            modeC.pop(0)
        else:
            ind = timeD.index(x)
            y = modeD[ind]
            if y==0:
                count_0 += 1
            timeD.pop(0)
            modeD.pop(0)

        time.append(x)

        if count_0>1:
            if y == 0:
                mode.append(0)
            else:
                mode.append(0)
                count_0 -= 1
        else:
```

```
            if y == 0:
                mode.append(y)
            else:
                mode.append(y)
                count_0 = 0
    time.append(SIM_TIME)
    mode.append(last_mode)

    # Justify the data to plot the system profile better
    final_time = []
    final_mode = []
    for i in range(1,len(time)-1):
        final_time.extend([time[i],time[i]])
        if mode[i]==0:
            if mode[i-1]==0:
                final_mode.extend([0,0])
            else:
                final_mode.extend([1,0])
        else:
            if mode[i-1]==1:
                final_mode.extend([1,1])
            else:
                final_mode.extend([0,1])
    final_time.insert(0,0)
    final_time.append(SIM_TIME)
    final_mode.insert(0,1)
    final_mode.append(mode[-1])

    # Calculate Production Availability
    hydrogen = 0
    up = 0
    buffer_level = 0
    for i in range(len(mode)-1):
        if mode[i] == 1:
            state_4 = time[i+1] - time[i]
            hydrogen += 900*state_4
            filling = min(80-buffer_level, up*0.0)
        elif (mode[i] == 0 and mode[i+1] == 1) or (mode[i] == 0 and mode[i
   +1] == 0 and mode[i-1] == 1):
            state_3 = time[i+1] - time[i]
            hydrogen += min(state_3, buffer_level/20)*0
            buffer_level -= min(state_3, buffer_level/20)*0

    return(hydrogen/(900*SIM_TIME))

```

```
# Execute
rep = []
for k in range(50000):          # 50000 replications
    env = simpy.Environment()
    repairman = simpy.Resource(env, capacity=1)
    LineA, LineB, LineC, LineD = [Line(env, 'Vessel %d' % i, repairman,)
   for i in range(4)]
    env.run(until=SIM_TIME)
    rep.append((calculation(LineA, LineB, LineC, LineD, SIM_TIME)))

print(mean(rep))
y = np.cumsum(rep)/range(1,len(rep)+1)
z = range(1, len(y)+1)
plt.plot(z,y)
plt.xlabel("Replication")
plt.ylabel("Mean")
plt.show()
```

C.2.2 Complex Model of PSA With a Buffer

```
"""
Scenario:
    1: H2 from individual adsorber varies according to operational modes:
            mode 0 = Adsorption       = +25 H2 per minute
            mode 1 = Depressurisation = +5  H2 per minute
            mode 2 = Purging          = -5  H2 per minute
            mode 3 = Repressurisation = -10 H2 per minute
        That is, one adsorber produces 225 H2 per hour
        4 adsorbers produce 900 H2 per hour
    2: Each mode lasts 4 minutes
    3: Buffer capacity = 900 H2
    4: Buffer withdrawing
            mode 0 = Adsorption       = -25 H2 per minute
            mode 1 = Depressurisation = -5  H2 per minute
    5: Buffer refilling
            mode 0 = +0.15 H2 per minute
            mode 2 = +5 H2 per minute
            mode 3 = +10 H2 per minute
"""

!pip install simpy
import random
import simpy
from statistics import mean
```

```
import matplotlib.pyplot as plt
import numpy as np

FR = (1.49e-5)           # Failure rate of one line (1 adsober and 5
   valves), in minutes
MTTR = 7.615             # Mean Time To Repair, in minutes
SIM_TIME = 8760*60*50    # Simulation time, 50 years, in minutes
FILL_RATE = 0.15         # m3 per minute
NUM_MACHINES = 4
count_down = 0

# Hydrogen production according to operational modes
class production():
    def mode(duration, mode):
        H2 = 0
        if mode % 4 == 0:
            H2 = duration*25
        elif mode % 4 == 1:
            H2 = duration*5
        elif mode % 4 == 2:
            H2 = duration*(-5)
        else:
            H2 = duration*(-10)
        return H2

# Refilling and withdrawing policy
class my_buffer():
    def withdraw(duration, mode, state):
        H2 = 0
        if state == 3 and mode % 4 == 0:
            H2 = duration*25
        elif state == 3 and mode % 4 == 1:
            H2 = duration*5
        elif state == 3 and mode % 4 == 2:
            H2 = duration*(-5)
        elif state == 3 and mode % 4 == 3:
            H2 = duration*(-10)
        return H2
    def refill_4(duration, mode):
        H2_refill = 0
        if mode % 4 == 0:
            H2_refill = duration*FILL_RATE
        return H2_refill

class Line(object):
```

```
    def __init__(self, env, name, repairman, buffer, state, mode):
        self.env = env
        self.name = name
        self.production = 0
        self.broken = False
        self.ideal = 0
        self.mode = mode
        self.start = 0

        self.process = env.process(self.working(repairman, buffer, state,
mode))
        env.process(self.break_valve())

    # Simulate 4-minute run-length
    def working(self, repairman, buffer, state, mode):
        global count_down
        while True:
            done_in = 4
            self.ideal += production.mode(done_in, self.mode)
            while done_in:
                self.start = self.env.now
                try:
                    # Perfect state, refill buffer if it is not full
                    if state.level == 4:
                        self.production += production.mode(done_in, self.
mode)
                        refill = min(buffer.capacity - buffer.level,
my_buffer.refill_4(done_in, self.mode))
                        if refill > 0:
                            buffer.put(refill)
                            self.production -= refill
                        self.mode += 1
                        yield self.env.timeout(done_in)
                        done_in = 0

                    # Line is under repair
                    elif count_down > done_in:
                        if buffer.level == 0:        # when buffer is
empty, we just move to the next mode
                            if self.broken:
                                count_down -= done_in
                        if state.level == 3 and buffer.level > 0:
                            if not self.broken:      # Working adsorbers
                                self.production += production.mode(done_in
, self.mode)
```

```
109                             else:                       # Failed adsorber
110                                 demand = my_buffer.withdraw(done_in, self.
    mode, state.level)
111                                 if demand > 0:          # Withdrawing at
    state 3
112                                     offer = min(demand, buffer.level)
113                                     buffer.get(offer)
114                                     self.production += offer
115                                 else:                   # Refilling at state
    3
116                                     refill = min(buffer.capacity - buffer.
    level, abs(demand))
117                                     if refill > 0:
118                                         buffer.put(refill)
119                                         self.production -= refill
120                                 count_down -= done_in
121                         self.mode += 1
122                         yield self.env.timeout(done_in)
123                         done_in = 0
124 
125                     # Line is under repair, but will be fixed within this
    mode
126                     elif count_down > 0 and count_down <= done_in:
127                         if buffer.level == 0:
128                             if self.broken:
129                                 self.broken = False
130                                 state.put(1)
131 
132                         if state.level == 3 and buffer.level > 0:
133                             if not self.broken:
134                                 self.production += production.mode(
    count_down, self.mode)
135                             else:
136                                 demand = my_buffer.withdraw(count_down,
    self.mode, state.level)
137                                 if demand > 0:
138                                     offer = min(demand, buffer.level)
139                                     buffer.get(offer)
140                                     self.production += offer
141                                 else:
142                                     refill = min(buffer.capacity - buffer.
    level, abs(demand))
143                                     if refill > 0:
144                                         buffer.put(refill)
145                                         self.production -= refill
```

```
146
147                                 # Repair is done
148                                 self.broken = False
149                                 state.put(1)
150
151                         done_in -= count_down
152                         yield self.env.timeout(count_down)
153                         count_down = 0
154
155                 # Interrupt the 4-minute mode at if failure takes place
156                 except simpy.Interrupt:
157                     t_up = self.env.now - self.start
158                     t_down = 4 - t_up
159                     t_jump = t_down
160
161                     #PRODUCTION BEFORE FAILURE
162                     if state.level == 3:  # It means it was 4 before
163                         self.production += production.mode(t_up, self.mode
    )
164
165                         refill = min(buffer.capacity - buffer.level,
    my_buffer.refill_4(t_up, self.mode))
166                         if refill > 0:
167                             buffer.put(refill)
168                             self.production -= refill
169
170                     if state.level == 2 and buffer.level > 0:
171                         demand = my_buffer.withdraw(t_up, self.mode, state
    .level)
172                         if demand > 0:
173                             offer = min(demand, buffer.level)
174                             buffer.get(offer)
175                             self.production += offer
176                         else:
177                             refill = min(buffer.capacity - buffer.level,
    abs(demand))
178                             if refill > 0:
179                                 buffer.put(refill)
180                                 self.production -= refill
181
182                     # Request a repairman
183                     with repairman.request() as req:
184                         yield req
185                         count_down += (random.expovariate(1/MTTR))*60
186
```

```
187                         if count_down < t_down:             # Faiulure and
    repair take place within 1 mode
188                             t_up = t_down - count_down      # New t_up
189                             t_down = count_down             # New t_down
190 
191                             if state.level == 3 and buffer.level > 0:
192                                 demand = my_buffer.withdraw(t_down, self.
    mode, state.level)
193                                 if demand > 0:
194                                     offer = min(demand, buffer.level)
195                                     buffer.get(offer)
196                                     self.production += offer
197                                 else:
198                                     refill = min(buffer.capacity - buffer.
    level, abs(demand))
199                                     if refill > 0:
200                                         buffer.put(refill)
201                                         self.production -= refill
202                             # Repair is done
203                             self.broken = False
204                             state.put(1)
205 
206                             if state.level == 4:
207                                 self.production += production.mode(t_up,
    self.mode)
208                                 refill = min(buffer.capacity - buffer.
    level, my_buffer.refill_4(t_up, self.mode))
209                                 if refill > 0:
210                                     buffer.put(refill)
211                                     self.production -= refill
212 
213                         #PRODUCTION AFTER FAILURE
214                         elif state.level == 3 and buffer.level > 0:
215                             demand = my_buffer.withdraw(t_down, self.mode,
     state.level)
216                             if demand > 0:
217                                 offer = min(demand, buffer.level)
218                                 buffer.get(offer)
219                                 self.production += offer
220                             else:
221                                 refill = min(buffer.capacity - buffer.
    level, abs(demand))
222                                 if refill > 0:
223                                     buffer.put(refill)
224                                     self.production -= refill
```

```
225
226                         done_in = 0
227                         count_down -= t_jump
228                         yield self.env.timeout(t_jump)
229
230     def break_valve(self):
231         """Break the machine every now and then."""
232         while True:
233             yield self.env.timeout((random.expovariate(FR))*60)
234             if not self.broken:
235                 # Only break the machine if it is currently working
236                 self.broken = True
237                 state.get(1)
238                 self.process.interrupt()   # This will interrupt the "
    working" process
239
240 # Execute
241 rep = []
242 for k in range(10):
243     env = simpy.Environment()
244     repairman = simpy.Resource(env, capacity=1)       # Define repair
    resource
245     state = simpy.Container(env, 4, init=4)           # State of the system
    , 4 means 4 adsorbers working
246     buffer = simpy.Container(env, 900, init=0)        # Define a container/
    buffer with capacity and initial level
247
248     lines = [Line(env, 'Line %d' % i, repairman, buffer, state, i) for i
    in range(NUM_MACHINES)]
249     env.run(until=SIM_TIME)
250     total_production = 0
251     ideal_production = 0
252     extra_hydrogen = 0
253     for line in lines:
254         total_production += line.production
255         ideal_production += line.ideal
256     A = total_production/ideal_production
257     rep.append(A)
258     print(f"{A}")
259
260 print(mean(rep))
261 y = np.cumsum(rep)/range(1,len(rep)+1)
262 z = range(1, len(y)+1)
263 plt.plot(z,y)
264 plt.xlabel("Replication")
```

```
plt.ylabel("Mean")
plt.show()
```

C.2.3 Simple Model of PSA With a Buffer

```
"""
Scenario: simple PSA model with buffer
    1: Simulate individual adsorbers based on failure and repair rates
    2: Apply operation rules to perform the refilling/withdarawing policy
   and calculate production availability
"""

!pip install simpy
import simpy
import matplotlib.pyplot as plt
from statistics import mean
import random
import numpy as np

SIM_TIME = 8760*50
BUFFER_CAPACITY = 900

class Line(object):
    def __init__(self, env, name, repairman):
        self.env = env
        self.name = name
        self.time = [0]   # Record the time
        self.mode = [1]   # Record the state, 1 = UP, 0 = DOWN

        self.process = env.process(self.working(repairman))

    def working(self, repairman):
        while True:
            yield self.env.timeout(time_to_failure())
            t_broken = env.now
            self.time.extend([t_broken])
            self.mode.extend([0])

            with repairman.request() as req:
                yield req
                yield self.env.timeout(time_to_repair())
                t_repair = env.now
                self.time.extend([t_repair])
                self.mode.extend([1])
```

```
                t_down = t_repair - t_broken

def time_to_failure():
    return random.expovariate(1.49e-5)

def time_to_repair():
    return random.expovariate(1/7.615)

"""
    From here to the end, calculate production availability based on
    adsorber profiles
"""

def calculation(LineA, LineB, LineC, LineD, SIM_TIME,):

    last_mode = min(LineA.mode[-1],LineB.mode[-1],LineC.mode[-1],LineD.
    mode[-1])

    # Create and customise component profiles
    timeA = LineA.time
    timeA.pop(0)
    timeA.append(SIM_TIME)
    modeA = LineA.mode
    modeA.append(LineA.mode[-1])
    modeA.pop(0)

    timeB = LineB.time
    timeB.pop(0)
    timeB.append(SIM_TIME)
    modeB = LineB.mode
    modeB.append(LineB.mode[-1])
    modeB.pop(0)

    timeC = LineC.time
    timeC.pop(0)
    timeC.append(SIM_TIME)
    modeC = LineC.mode
    modeC.append(LineC.mode[-1])
    modeC.pop(0)

    timeD = LineD.time
    timeD.pop(0)
    timeD.append(SIM_TIME)
    modeD = LineD.mode
    modeD.append(LineD.mode[-1])
```

```
    modeD.pop(0)


# Create system profile based on system configuration
    time = [0]
    mode = [1]
    count_0 = 0
    while not (len(timeA)==1 and len(timeB)==1 and len(timeC)==1 and len(
   timeD)==1):
        x = min(min(timeA),min(timeB), min(timeC), min(timeD))
        if timeA.count(x)>0:
            ind = timeA.index(x)
            y = modeA[ind]
            if y==0:
                count_0 += 1
            timeA.pop(0)
            modeA.pop(0)
        elif timeB.count(x)>0:
            ind = timeB.index(x)
            y = modeB[ind]
            if y==0:
                count_0 += 1
            timeB.pop(0)
            modeB.pop(0)
        elif timeC.count(x)>0:
            ind = timeC.index(x)
            y = modeC[ind]
            if y==0:
                count_0 += 1
            timeC.pop(0)
            modeC.pop(0)
        else:
            ind = timeD.index(x)
            y = modeD[ind]
            if y==0:
                count_0 += 1
            timeD.pop(0)
            modeD.pop(0)

        time.append(x)

        if count_0>1:
            if y == 0:
                mode.append(0)
            else:
```

```
                mode.append(0)
                count_0 -= 1
        else:
            if y == 0:
                mode.append(y)
            else:
                mode.append(y)
                count_0 = 0
    time.append(SIM_TIME)
    mode.append(last_mode)

    # Justify the data to plot the system profile better
    final_time = []
    final_mode = []
    for i in range(1,len(time)-1):
        final_time.extend([time[i],time[i]])
        if mode[i]==0:
            if mode[i-1]==0:
                final_mode.extend([0,0])
            else:
                final_mode.extend([1,0])
        else:
            if mode[i-1]==1:
                final_mode.extend([1,1])
            else:
                final_mode.extend([0,1])
    final_time.insert(0,0)
    final_time.append(SIM_TIME)
    final_mode.insert(0,1)
    final_mode.append(mode[-1])

    # Calculate Production Availability
    hydrogen = 0
    up = 0
    buffer_level = 0
    for i in range(len(mode)-1):
        if mode[i] == 1:
            state_4 = time[i+1] - time[i]
            hydrogen += 900*state_4
            filling = min(BUFFER_CAPACITY-buffer_level, state_4*
  refill_amount)
            buffer_level += filling
            hydrogen -= filling
            up += state_4
        elif (mode[i] == 0 and mode[i+1] == 1) or (mode[i] == 0 and mode[i
```

```
   +1] == 0 and mode[i-1] == 1):
170             state_3 = time[i+1] - time[i]
171             capability = min(state_3, buffer_level/225)
172             hydrogen += capability*900
173             buffer_level -= capability*225
174             up += capability
175 
176     return(hydrogen/(900*SIM_TIME))
177 
178 refill_amount = 9
179 rep = []
180 for k in range(50000):
181     env = simpy.Environment()
182     repairman = simpy.Resource(env, capacity=1)
183     LineA, LineB, LineC, LineD = [Line(env, 'Vessel %d' % i, repairman,)
    for i in range(4)]
184     env.run(until=SIM_TIME)
185     rep.append((calculation(LineA, LineB, LineC, LineD, SIM_TIME)))
186 
187 print(mean(rep))
188 y = np.cumsum(rep)/range(1,len(rep)+1)
189 z = range(1, len(y)+1)
190 plt.plot(z,y)
191 plt.xlabel("Replication")
192 plt.ylabel("Mean")
193 plt.show()
```

Bibliography

Terje Aven. 1987. Availability evaluation of oil/gas production and transportation systems. *Reliability Engineering* 18, 1 (1987), 35–44. https://doi.org/10.1016/0143-8174(87)90050-3

Terje Aven and Linda Martens Pedersen. 2014. On how to understand and present the uncertainties in production assurance analyses, with a case study related to a subsea production system. *Reliability Engineering System Safety* 124 (2014), 165–170. https://doi.org/10.1016/j.ress.2013.12.003

Jerry Banks, John S. Carson, Barry L. Nelson, and David M. Nicol. 2005. *Discrete-Event System Simulation* (4 ed.). Prentice-Hall.

Javad Barabady, Tore Markeset, and Uday Kumar. 2010. Review and discussion of production assurance program. *International Journal of Quality Reliability Management* 27 (06 2010), 702–720. https://doi.org/10.1108/02656711011054560

Louis Bela Batta. 1971. Selective adsorption process. (1971).

Mokhtar S. Bazaraa, John J. Jarvis, and Hanif D. Sherali. 2004. *Linear Programming and Network Flows*. Wiley-Interscience, USA.

Prakash Biswas, S Agrawal, and Shishir Sinha. 2010. Modeling and Simulation for Pressure Swing Adsorption System for Hydrogen Purification. *Chemical and Biochemical Engineering Quarterly* 24 (01 2010).

M. Boiteau, Y. Dutuit, A. Rauzy, and J.-P. Signoret. 2006. The AltaRica data-flow language in use: modeling of production availability of a multi-state system. *Reliability Engineering System Safety* 91, 7 (2006), 747–755. https://doi.org/10.1016/j.ress.2004.12.004

E. Borgonovo, M. Marseguerra, and E. Zio. 2000. A Monte Carlo methodological approach to plant availability modeling with maintenance, aging and obsolescence. *Reliability Engineering System Safety* 67, 1 (2000), 61–73. https://doi.org/10.1016/S0951-8320(99)00046-0

Radim Briš. 2013. Evaluation of the production availability of an offshore installation by stochastic Petri nets modeling. In *The International Conference on Digital Technologies 2013*. 147–155. https://doi.org/10.1109/DT.2013.6566303

C.G. Cassandras and Stephane Lafortune. 2010. *Introduction to Discrete Event Systems*. 800. https://doi.org/10.1007/978-0-387-68612-7

R. T. Cassidy. 1980. *Polybed Pressure-Swing Adsorption Hydrogen Processing*. Chapter 13, 247–259. https://doi.org/10.1021/bk-1980-0135.ch013 arXiv:https://pubs.acs.org/doi/pdf/10.1021/bk-1980-0135.ch013

Schalk Cloete, Carlos Arnaiz del Pozo, and Ángel Jiménez Álvaro. 2022. System-friendly process design: Optimizing blue hydrogen production for future energy systems. *Energy* 259 (2022), 124954. https://doi.org/10.1016/j.energy.2022.124954

Peter T. Clough, Matthew E. Boot-Handford, Liya Zheng, Zili Zhang, and Paul S. Fennell. 2018. Hydrogen Production by Sorption Enhanced Steam Reforming (SESR) of Biomass in a Fluidised-Bed Reactor Using Combined Multifunctional Particles. *Materials* 11, 5 (2018). https://doi.org/10.3390/ma11050859

Kay Damen, Martijn van Troost, André Faaij, and Wim Turkenburg. 2006. A comparison of electricity and hydrogen production systems with CO2 capture and storage. Part A: Review and selection of promising conversion and capture technologies. *Progress in Energy and Combustion Science* 32, 2 (2006), 215–246. https://doi.org/10.1016/j.pecs.2005.11.005

William Feller. 1964. ON SEMI-MARKOV PROCESSES[*]. *Proceedings of the National Academy of Sciences* 51, 4 (1964), 653–659. https://doi.org/10.1073/pnas.51.4.653 arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.51.4.653

Ricardo Fricks, Antonio Puliafito, and Kishor Trivedi. 2001. Markov Renewal Theory Applied To Performability Evaluation. (03 2001).

Carl-Erik Fröberg. 1972. *Introduction to Numerical Analysis*. Addison-Wesley. https://books.google.no/books?id=fTNpnQEACAAJ

Paul Gagniuc. 2017. *Markov Chains: From Theory to Implementation and Experimentation*. https://doi.org/10.1002/9781119387596

Franciszek Grabski. 2016. Concept of Semi -Markov Process. *Maritime Technical Journal* 206, 3 (2016), 25–36. https://doi.org/doi:10.5604/0860889x.1224743

Kathryn Hoad, Stewart Robinson, and Ruth Davies. 2007. Automating des output analysis: How many replications to run. In *2007 Winter Simulation Conference*. 505–512. https://doi.org/10.1109/WSC.2007.4419641

Per Hokstad. 1988. Assessment of production regularity for subsea oil/gas production systems. *Reliability Engineering & System Safety* 20 (1988), 127–146.

IEA IEA. 2019. The future of hydrogen. *International Energy Agency report* (2019).

International Electrotechnical Commission IEC. 2003. Dependability management – Part 3-1: Application guide – Analysis techniques for dependability – Guide on methodology. (2003).

IEC-61165. 2006. NEK IEC 61165:2006 Application of Markov techniques. (2006).

Hydrogen Irena. 2019. A renewable energy perspective. *International Renewable Energy Agency, Abu Dhabi* (2019).

ISO-20815. 2018. ISO 20815 Petroleum, petrochemical and natural gas industries Production assurance and reliability management. (2018).

Zhihong Jin, Katsuhisa Ohno, and Jiali Du. 2004. AN EFFICIENT APPROACH FOR THE THREE-DIMENSIONAL CONTAINER PACKING PROBLEM WITH PRACTICAL CONSTRAINTS. *Asia-Pacific Journal of Operational Research* 21, 03 (2004), 279–295. https://doi.org/10.1142/S0217595904000254 arXiv:https://doi.org/10.1142/S0217595904000254

Yoshio Kawauchi and Marvin Rausand. 2002. A new approach to production regularity assessment in the oil and chemical industries. *Reliability Engineering System Safety* 75 (03 2002), 379–388. https://doi.org/10.1016/S0951-8320(01)00130-2

Akram Khaleghei and Viliam Makis. 2015. Model parameter estimation and residual life prediction for a partially observable failing system. *Naval Research Logistics (NRL)* 62, 3 (2015), 190–205. https://doi.org/10.1002/nav.21622 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.21622

Yaser Khojasteh Salkuyeh, Bradley A. Saville, and Heather L. MacLean. 2017. Techno-economic analysis and life cycle assessment of hydrogen production from natural gas using current and emerging technologies. *International Journal of Hydrogen Energy* 42, 30 (2017), 18894–18909. https://doi.org/10.1016/j.ijhydene.2017.05.219

Leïla Kloul and Antoine Rauzy. 2017. Production trees: A new modeling methodology for production availability analyses. *Reliability Engineering System Safety* 167 (2017), 561–571. https://doi.org/10.1016/j.ress.2017.06.017 Special Section: Applications of Probabilistic Graphical Models in Dependability, Diagnosis and Prognosis.

Vo S Korolyuk, SM Brodi, and AF Turbin. 1975. Semi-Markov processes and their applications. *Journal of Soviet Mathematics* 4, 3 (1975), 244–280.

Averill M. Law and Michael G. McComas. 1991. Secrets of successful simulation studies. *1991 Winter Simulation Conference Proceedings.* (1991), 21–27.

Paul Levy. 1954. Processus semi-markoviens. In *Proc. Int. Congress. Math. III, Amsterdam, 1954.*

Nikolaos Limnios and Vlad Stefan Barbu. 2009. Semi-Markov Chains and Hidden Semi-Markov Models toward Applications: Their Use in Reliability and DNA Analysis by BARBU, V. S. and LIMNIOS, N. *Biometrics* 65, 4 (2009), 1312–1312. https://doi.org/10.1111/j.1541-0420.2009.01343_8.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1541-0420.2009.01343$_8$.x

N. Limnios and G. Oprişan. 2001. *Introduction to Stochastic Processes and the Renewal Process.* Birkhäuser Boston, Boston, MA, 1–29. https://doi.org/10.1007/978-1-4612-0161-8_1

N. Limnios, B. Ouhbi, and A. Sadek. 2005. Empirical Estimator of Stationary Distribution for Semi-Markov Processes. *Communications in Statistics - Theory and Methods* 34, 4 (2005), 987–995. https://doi.org/10.1081/STA-200054441 arXiv:https://doi.org/10.1081/STA-200054441

Anatoly Lisnianski and Gregory Levitin. 2003. *Multi-State System Reliability.* WORLD SCIENTIFIC. https://doi.org/10.1142/5221 arXiv:https://www.worldscientific.com/doi/pdf/10.1142/5221

Mauro Luberti and Hyungwoong Ahn. 2022. Review of Polybed pressure swing adsorption for hydrogen purification. *International Journal of Hydrogen Energy* 47, 20 (2022), 10911–10933. https://doi.org/10.1016/j.ijhydene.2022.01.147

Mauro Luberti, Daniel Friedrich, Stefano Brandani, and Hyungwoong Ahn. 2014. Design of a H-2 PSA for cogeneration of ultrapure hydrogen and power at an advanced integrated gasification combined cycle with pre-combustion capture. *ADSORPTION-JOURNAL OF THE INTERNATIONAL ADSORPTION SOCIETY* 20, 2-3, SI (FEB 2014), 511–524. https://doi.org/10.1007/s10450-013-9598-0

Ming Luo, Yang Yi, Shuzhong Wang, Zhuliang Wang, Min Du, Jianfeng Pan, and Qian Wang. 2018. Review of hydrogen production using chemical-looping technology. *Renewable and Sustainable Energy Reviews* 81 (2018), 3186–3214. https://doi.org/10.1016/j.rser.2017.07.007

I. Martínez, M.C. Romano, J.R. Fernández, P. Chiesa, R. Murillo, and J.C. Abanades. 2014. Process design of a hydrogen production plant from natural gas with CO2 capture based on a novel Ca/Cu chemical loop. *Applied Energy* 114 (2014), 192–208. https://doi.org/10.1016/j.apenergy.2013.09.026

Mario Martínez, Jesse Rumbo Morales, G. Ortiz-Torres, Salvador Paredes, Sebastián Reyes, Felipe Sorcia-Vázquez, Alan Perez Vidal, Jorge Valdez Martínez, Ricardo Pérez-Zúñiga, and Erasmo Vargas. 2022. Simulation and State Feedback Control of a Pressure Swing Adsorption Process to Produce Hydrogen. *Mathematics* 10 (05 2022), 1762. https://doi.org/10.3390/math10101762

J. Medhi. 2003. CHAPTER 1 - Stochastic Processes. In *Stochastic Models in Queueing Theory (Second Edition)* (second edition ed.), J. Medhi (Ed.). Academic Press, Burlington, 1–46. https://doi.org/10.1016/B978-012487462-6/50001-1

Amal Mehrotra, Armin D. Ebner, and James A. Ritter. 2010. Arithmetic approach for complex PSA cycle scheduling. *Adsorption* 16 (2010), 113–126.

Shareq Mohd Nazir, Jan Hendrik Cloete, Schalk Cloete, and Shahriar Amini. 2019. Efficient hydrogen production with CO2 capture using gas switching reforming. *Energy* 185 (2019), 372–385. https://doi.org/10.1016/j.energy.2019.07.072

NORSOK-Z016. 1998. Z-016 Regularity Management and Reliability Technology. (1998).

Michel Noussan, Pier Paolo Raimondi, Rossana Scita, and Manfred Hafner. 2021. The Role of Green and Blue Hydrogen in the Energy Transition—A Technological and Geopolitical Perspective. *Sustainability* 13, 1 (2021). https://doi.org/10.3390/su13010298

Walter R Nunn and Anthony M Desiderio. 1977. *Semi-markov processes: an introduction.* Technical Report. CENTER FOR NAVAL ANALYSES ARLINGTON VA OPERATIONS EVALUATION GROUP.

A. O. Oni, K. Anaya, T. Giwa, G. Di Lullo, and A. Kumar. 2022. Comparative assessment of blue hydrogen from steam methane reforming, autothermal reforming, and natural gas decomposition technologies for natural gas-producing regions. *ENERGY CONVERSION AND MANAGEMENT* 254 (FEB 15 2022). https://doi.org/10.1016/j.enconman.2022.115245

C. Persskog-Lundtofte, S. Isaksen, and L. M. Pedersen. 2015. How to present and understand uncertainty of production assurance analyses with respect to data quality. In *SAFETY AND RELIABILITY: METHODOLOGY AND APPLICATIONS*, T Nowakowski, M Mlynczak, A JodejkoPietruczuk, and S WerbinskaWojciechowska (Eds.). Wroclaw Univ Technol; Polish Safety & Reliabil Assoc; European Safety & Reliabil Assoc, 1857–1864. PROCEEDINGS OF THE EUROPEAN SAFETY AND RELIABILITY CONFERENCE (ESREL), Wroclaw, POLAND, SEP 14-18, 2014.

Ronald Pyke. 1961. Markov Renewal Processes: Definitions and Preliminary Properties. *The Annals of Mathematical Statistics* 32, 4 (1961), 1231–1242. http://www.jstor.org/stable/2237923

Marvin Rausand and Arnljot Høyland. 2004. *System Reliability Theory: Models, Statistical Methods, and Applications* (2nd ed.). Wiley, Hoboken, NJ.

Ana M. Ribeiro, João C. Santos, Alírio E. Rodrigues, and Sébastien Rifflart. 2012. Syngas Stoichiometric Adjustment for Methanol Production and Co-Capture of Carbon Dioxide by Pressure Swing Adsorption. *Separation Science and Technology* 47, 6 (2012), 850–866. https://doi.org/10.1080/01496395.2011.637282 arXiv:https://doi.org/10.1080/01496395.2011.637282

James A. Ritter and Armin D. Ebner. 2007. State-of-the-Art Adsorption and Membrane Separation Processes for Hydrogen Production in the Chemical and Petrochemical Industries. *Separation Science and Technology* 42, 6 (2007), 1123–1193. https://doi.org/10.1080/01496390701242194 arXiv:https://doi.org/10.1080/01496390701242194

Stewart Robinson. 2004. Simulation: The Practice of Model Development and Use.

S.M. Ross. 1996. *Stochastic Processes*. Wiley. https://books.google.no/books?id=ImUPAQAAMAAJ

Sheldon M. Ross. 2019a. 4 - Markov Chains. In *Introduction to Probability Models (Twelfth Edition)* (twelfth edition ed.), Sheldon M. Ross (Ed.). Academic Press, 193–291. https://doi.org/10.1016/B978-0-12-814346-9.00009-3

Sheldon M. Ross. 2019b. 6 - Continuous-Time Markov Chains. In *Introduction to Probability Models (Twelfth Edition)* (twelfth edition ed.), Sheldon M. Ross (Ed.). Academic Press, 375–429. https://doi.org/10.1016/B978-0-12-814346-9.00011-1

S. Ruthven, D.M. Farooq and K.S Knaebel. 1994. *Pressure swing adsorption*. VCH, New York.

Magnus Rydén, Anders Lyngfelt, and Tobias Mattisson. 2006. Synthesis gas generation by chemical-looping reforming in a continuously operating laboratory reactor. *Fuel* 85, 12 (2006), 1631–1641. https://doi.org/10.1016/j.fuel.2006.02.004

Stanley Santos, Guido Collodi, Giuliana Azzaro, and Noemi Ferrari. 2017. Techno-Economic Evaluation of SMR Based Standalone (Merchant) Plant with CCS. (02 2017).

Team SimPy. 2013. *Documentation for SimPy*. https://simpy.readthedocs.io/en/3.0/contents.html

SINTEF. 2009. *OREDA: Offshore Reliability Data Fifth Edition*.

W. Smith. 1955. Regenerative stochastic processes. In *Proc. Roy. Soc., ser. A, 232, 1955*.

R. Soltani, M.A. Rosen, and I. Dincer. 2014. Assessment of CO2 capture options from various points in steam methane reforming for hydrogen production. *International Journal of Hydrogen Energy* 39, 35 (2014), 20266–20275. https://doi.org/10.1016/j.ijhydene.2014.09.161

Chunfeng Song, Qingling Liu, Na Ji, Yasuki Kansha, and Atsushi Tsutsumi. 2015. Optimization of steam methane reforming coupled with pressure swing adsorption hydrogen production process by heat integration. *Applied Energy* 154 (2015), 392–401. https://doi.org/10.1016/j.apenergy.2015.05.038

J. Stöcker, M. Whysall, Antwerp, and G.Q. Miller. 1998. 30 years of PSA technology for Hydrogen production. (1998).

Tianqi Sun. 2017. Production Availability Analysis: Implications on Modelling due to Subsea Conditions. (2017).

Kishor S. Trivedi, Kalyanaraman Vaidyanathan, and Dharmaraja Selvamuthu. 2015. Chapter 13 - Markov chain models and applications. In *Modeling and Simulation of Computer Networks and Systems*, Mohammad S. Obaidat, Petros Nicopolitidis, and Faouzi Zarai (Eds.). Morgan Kaufmann, Boston, 393–421. https://doi.org/10.1016/B978-0-12-800887-4.00013-4

Lucas Van Cappellen, HJ Croezen, and FJ Rooijers. 2018. *Feasibility Study into Blue Hydrogen: Technical, Economic & Sustainability Analysis*. CE Delft.

Astrid Hetland Vesteraas. 2008. Comparision of methods and software tools for availability assessment of production systems. (2008).

W.E. Waldron. 2000. Parametric Study of a Pressure Swing Adsorption Process. *Adsorption* 6 (06 2000), 179–188. https://doi.org/10.1023/A:1008925703871

Liaoyi Wang. 2012. Production Assurance and Life Cycle Cost Evaluation of Offshore Development Projects in the Conceptual Design Phase. (2012).

Mona Wappler, Dilek Unguder, Xing Lu, Hendrik Ohlmeyer, Hannah Teschke, and Wiebke Lueke. 2022. Building the green hydrogen market – Current state and outlook on green hydrogen demand and electrolyzer manufacturing. *International Journal of Hydrogen Energy* 47, 79 (2022), 33551–33570. https://doi.org/10.1016/j.ijhydene.2022.07.253

Ronald C Hoke Charles W Skarstrom William D Marsh, Francis S Pramuk. 1961. Pressure equalization depressuring in heatless adsorption. (1961).

Jinsheng Xiao, Liang Fang, Pierre Bénard, and Richard Chahine. 2018. Parametric study of pressure swing adsorption cycle for hydrogen purification using Cu-BTC. *International Journal of Hydrogen Energy* 43, 30 (2018), 13962–13974. https://doi.org/10.1016/j.ijhydene.2018.05.054 Special Issue on The 8th International Conference on Hydrogen Production (ICH2P-2017).

Jinsheng Xiao, Ruipu Li, Pierre Bénard, and Richard Chahine. 2015. Heat and mass transfer model of multicomponent adsorption system for hydrogen purification. *International Journal of Hydrogen Energy* 40, 14 (2015), 4794–4803. https://doi.org/10.1016/j.ijhydene.2015.02.042

Yongliang Yan, Dhinesh Thanganadar, Peter T. Clough, Sanjay Mukherjee, Kumar Patchigolla, Vasilije Manovic, and Edward J. Anthony. 2020. Process simulations of blue hydrogen production by upgraded sorption enhanced steam methane reforming (SE-SMR) processes. *Energy Conversion and Management* 222 (2020), 113144. https://doi.org/10.1016/j.enconman.2020.113144

Ralph T Yang. 1987. *Gas Separation by Adsorption Processes*. Butterworths, Boston. arXiv:https://www.worldscientific.com/doi/pdf/10.1142/p037 https://www.worldscientific.com/doi/abs/10.1142/p037

R. T. Yang and S. J. Doong. 1985. Gas separation by pressure swing adsorption: A pore-diffusion model for bulk separation. *AIChE Journal* 31, 11 (1985), 1829–1842. https://doi.org/10.1002/aic.690311109 arXiv:https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.690311109

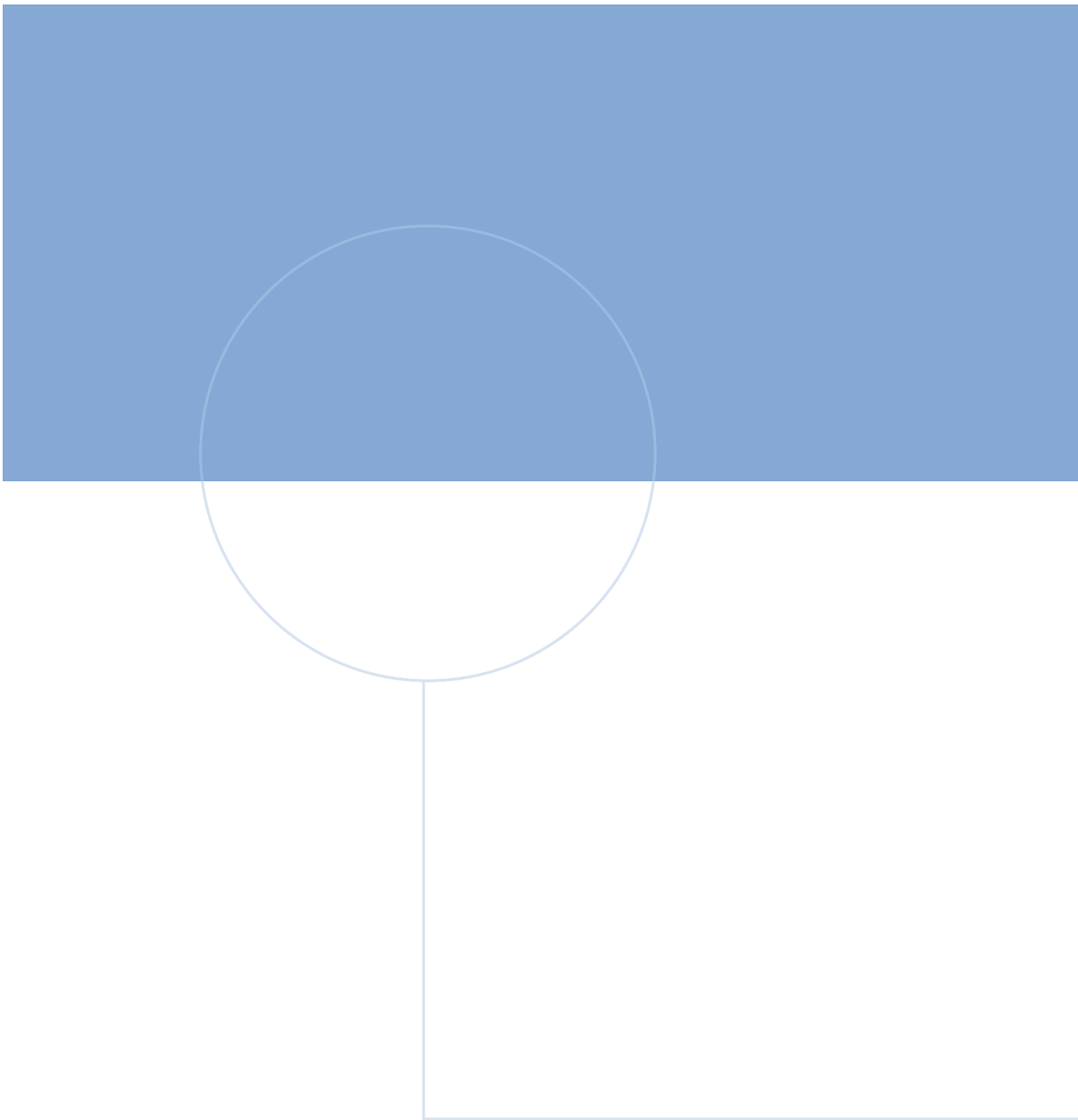
Milad Yavary, Habib Ale Ebrahim, and Cavus Falamaki. 2015. The effect of number of pressure equalization steps on the performance of pressure swing adsorption process. *Chemical Engineering and Processing: Process Intensification* 87 (2015), 35–44. https://doi.org/10.1016/j.cep.2014.11.003

Liang Yin, R.M. Fricks, and K.S. Trivedi. 2002. Application of semi-Markov process and CTMC to evaluation of UPS system availability. In *Annual Reliability and Maintainability Symposium. 2002 Proceedings (Cat. No.02CH37318)*. 584–591. https://doi.org/10.1109/RAMS.2002.981706

Li Zhou, Chang-Zhong Lü, Shou-Jun Bian, and Ya-Ping Zhou. 2002. Pure Hydrogen from the Dry Gas of Refineries via a Novel Pressure Swing Adsorption Process. *Industrial & Engineering Chemistry Research* 41, 21 (2002), 5290–5297. https://doi.org/10.1021/ie020043j arXiv:https://doi.org/10.1021/ie020043j

Enrico Zio, Piero Baraldi, and Edoardo Patelli. 2006. Assessment of the availability of an offshore installation by Monte Carlo simulation. *International Journal of Pressure Vessels and Piping* 83, 4 (2006), 312–320. https://doi.org/10.1016/j.ijpvp.2006.02.010 The 16th European Safety and Reliability Conference.

Erhan Çinlar. 1969. Markov renewal theory. *Advances in Applied Probability* 1, 2 (1969), 123–187. https://doi.org/10.2307/1426216



NTNU

Norwegian University of
Science and Technology