TÉCNICO LISBOA

A Benchmark Model to Generate Batch Process Data for Machine Learning Testing and Comparison

Margarida Dos Santos Louçada Coelho Vicente

Thesis to obtain the Master of Science Degree in

Chemical Engineering

Supervisors: Dr. Franz David Bähner
Dr. José Filipe Oliveira Granjo

Examination Committee

Chairperson: Prof. Maria de Fátima Grilo da Costa Montemor
Supervisor: Dr. Franz David Bähner
Members of the Committee: Prof. Carla Isabel Costa Pinheiro
Dr. Roumu Tan

September 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

The completion of the dissertation would not have been possible without the help, support, and dedication of several people whom I hereby want to acknowledge.

Foremost, I want to show my gratitude to both my supervisors. First, to Dr. Franz Bähner, for allowing and giving me the chance to work with him and with the most talented group of engineers at Bayer on the Process Modeling and Design department; for his tireless support, care, and more upfront availability. Secondly, to Dr. José Granjo for accepting me as one of his master thesis students, for encouraging me to pursue even greater achievements on my academic and professional curriculum. To both, I am truly indebted.

To all my friends and colleagues at Bayer, I want to express my gratitude for the utmost supportive environment. Even remotely, the way I was welcomed and treated from day one mirrors the incredible team I worked alongside.

My sincere thanks also go to Dr.ª Ruomu Tan, a research scientist from ABB (Asea Brown Boveri), for the insightful comments and discussion of future work endeavors.

I also want to thank Gregor Just, a student from the Technical University of Dresden, that shared the same passion and purpose on this project as I did.

Last but not least, I show my appreciation to the two people who most support me in every challenge and celebrate even the smaller milestones: my parents.

A todos, obrigada.

Abstract

During production, batch processes generate batch data and time-series values. The former contains start/end dates of production from the archived Manufacturing Execution or Batch Control systems. Furthermore, it may contain several quality-related or custom attributes. The time-series of the process are gathered by sensors and controllers containing information about production delays/variability. These data are valuable to build process models to support decision-making in planning and scheduling.

Due to the repetitiveness nature of batch processes, there are reproducible behaviors among the data. For that reason, Machine Learning (ML) algorithms have been exploited for generating batch statistics. However, the practical use of these techniques is complicated by the noise in the data (transient, stochastic, and discrete), scarcity of quality data to train the algorithm, or process disturbances.

The main ambition is to develop a benchmark model of a batch process to generate data where active challenges, disturbances, and noise are fully controlled. The process is viewed at the unit level with filling, processing, draining, and cleaning operations; the raw-materials loads are valve-controlled; and, served by hot and cold utilities. The simulation is limited to mass balances with chemical reactions and kinetics being omitted (not relevant for the intended application of ML). Finally, several scenarios were generated where 21 disturbances of different types, causes, or fault origins are injected (isolated or combined) in the implementation for testing. The scenarios were grouped into six benchmark cycles with increasing levels of complexity in terms of intensity, duration, and probability tackling some major challenges in ML.

Keywords

Batch process, MATLAB Simulink Stateflow, Hybrid dynamic simulation, Process Modelling, Machine Learning, Benchmark Model

Resumo

Durante a produção, os processos descontínuos geram informação acerca de eventos e valores de série temporal. O primeiro contém as datas de início/fim de produção provenientes do sistema de manufatura e controlo, além de poder conter detalhes acerca da qualidade de produção. Os dados temporais provêm de controladores e sensores contendo informação sobre atrasos/variabilidade do processo. Estes dados são pertinentes para a otimização de decisões no planeamento e programação de lotes.

Visto que, os dados de eventos nem sempre são gerados durante a produção, as ferramentas de aprendizagem de máquina (AM) podem ajudar na sua geração. Embora haja repetibilidade em processos descontínuos, o uso destas técnicas é complexo devido ao ruído existente (transiente, estocástico e discreto), falta de qualidade nos dados ou perturbações no processo.

O principal objetivo da dissertação é desenvolver um processo descontínuo como caso de estudo em que os desafios, perturbações e ruídos no processo são totalmente controlados. As operações processuais e de limpeza ocorrem num único tanque; as cargas de matérias-primas são controladas por válvulas; e, são utilizadas utilidades quente e fria. A simulação é limitada a balanços de massa, omitindo reações químicas e cinética (não é relevante para a aplicação de AM pretendida). Finalmente, vários cenários foram gerados em que 21 perturbações de diferentes classes, causas e origens são injetadas (isoladas ou combinadas) na implementação do processo. Os cenários foram agrupados em seis ciclos de referência com níveis crescentes de complexidade em termos de intensidade, duração e probabilidade com foco nos principais desafios de AM.

Palavras Chave

Processo descontínuo, MATLAB Simulink Stateflow, Simulação Dinâmica Híbrida, Modelação de Processos, Aprendizagem de Máquina, Caso de Estudo

Contents

List of Figures

List of Tables

List of Publications and Communications

Part of the work contained in this thesis has been:

- Preliminarily accepted for a poster communication entitled "A Benchmark Model to Generate Batch Process Data for Machine Learning Testing and Comparison" at the "32nd European Symposium on Computer-Aided Process Engineering" that will take place in Toulouse from June 12nd and 15th, 2022.
- Incorporated in a manuscript of book chapter for the book series "Computer Aided Chemical Engineering" that is to be submitted for peer review no later than November 15th, 2021.

1

Introduction

Contents

Batch processes are more expensive, harder to operate, and less efficient than continuous processes. They also have a greater environmental and economic impact than a continuous process in a single product scenario [14]. Nonetheless, several reasons make them attractive for producing fine and specialty chemicals and biochemicals, being frequently used in these industries. Among them, the flexibility of the process is exploited by producing different products in the same plant [15]. Production in batch allows for the production in campaigns (small amount of product with high economical value) where is not feasible to operate in continuous.

A batch process is characterized by a cyclic operation of one or several units that are filled with material, perform their desired task for a given duration, shut down, drained, and cleaned before the cycle is repeated [6]. The sequence and parameters of operation follow a specific recipe or *production schedule*, typically based on heuristics and experience [16, 17]. In practice, the execution of the production schedule is subjected to disturbances and noise that distresses the start and end points of the production schedule [9]. Therefore, *production planning* needs to make **robust production plans** which allow for **variability** without having delays at the customer end [13]. Such can be achieved by introducing time or material buffers (time constraint for the production to be complete two days before the customer's deadline) [18]. Production scheduling reacts to these disturbances and assures the deadlines of the robust production plan are met by allocating time-slots to pieces of equipment and re-adjusting the schedule [18]. These decisions can be: "if Unit1 is broken down, produce Product1 on Unit2, even though it leads to a delay in Product2, which is acceptable".

Decisions in planning and scheduling are based on batch data (also designated as event-data) which generally contains start and end dates of production, and may furthermore contain several quality-related or custom attributes (such as product content). The time-series values (gathered by the sensors and controllers during production) of the process contain information about the start/end/delays/variability of the process. For that reason, time-series values are interesting for the planner, although not ordinarily used in planning without previous processing and analysis [14, 18].

Batch data are needed to evaluate and optimize production planning, scheduling, and a wide range of retrofit optimization projects which can lead to better economic and environmental results. This can be done in several ways (very often by experts and Microsoft Excel calculations) but **process modelling and discrete-event simulation** is one method that can be used [16, 19]. The value of data presents increasing value in the industry of chemical engineering for process control, recognition of bottlenecks, efficient technology transfer and process fitting [14, 18].

1.1 Hybrid Simulation

The development and application of mechanistic models are not economically worth/practical for complex batch processes especially if involving a large number of equations with a significant number of process variables. For that principle, it is required several experiments for model parameterization, and the time necessary to develop this model can be quite long. As such, systems that ordinarily contain non-linear continuous equations and need a discrete-event handler require a specialized solver [13].

Since decisions in scheduling and planning are made by looking at batch data, it is acceptable to make the abstraction to only contemplate material hold-ups and transfer in model building. Hybrid simulation coupled with empirical models allows to train the model in an easier way (parameter estimation); to update/adapt more frequently; and, to solve it in real-time. Combining both time-driven and event-time environments still allows for fast execution if beyond the simple mass balance (reduced to a single component)—no further heat, or mass transfer phenomena, or reaction kinetics are included. The implementation of a continuous-discrete simulation can be modeled by combining Stateflow® and Simulink® [20].

The functioning of a continuous system (with the process, actuators, sensors, and a control system - designated as *regulatory control*) is of no relevance here. From a production planning perspective, it is not economic to consider first-principles models with detailed reaction kinetics, as model-building is expensive, execution time is long, and little or no information would be gained from it. Thus, the primary focus is on **Measurement and Actuation with the Inventory Control System** (Level 1 of Figure 1.1) since measurement devices (sensors) and actuators (for example, valve positions) are used to measure the process variables [1,2]. The same output is then used to train the Machine Learning (ML) algorithm.

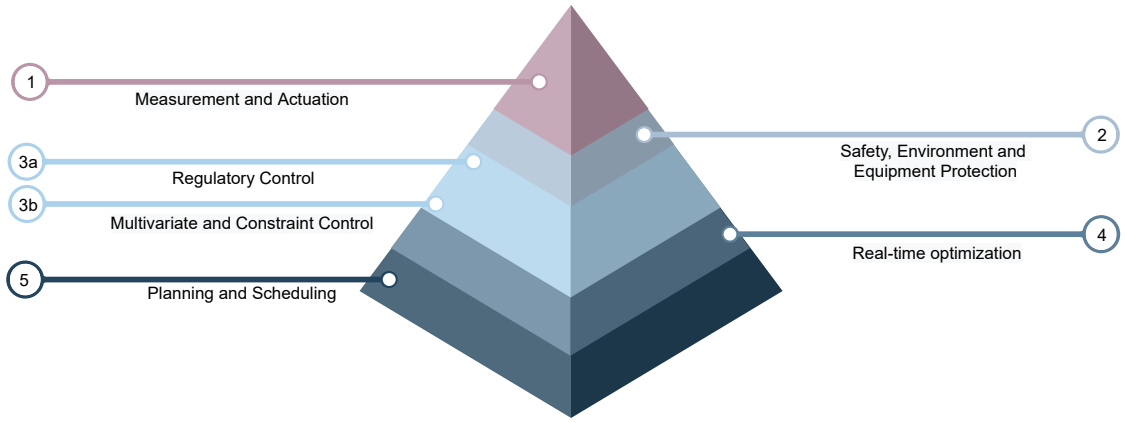


Figure 1.1: Five steps of process control and optimization in manufacturing. Adapted from [1,2].

Figure 1.2 highlights the two types of batch data: archived as planned that corresponds to the ideal plan and extracted from the Manufacturing Execution System (MES); and, the real data which is an

actual precise value of start and end points after the process being subjected to disturbances and noise. Process optimization is achieved by continuously looking at all gathered data (batch data statistics) and change the production planning.

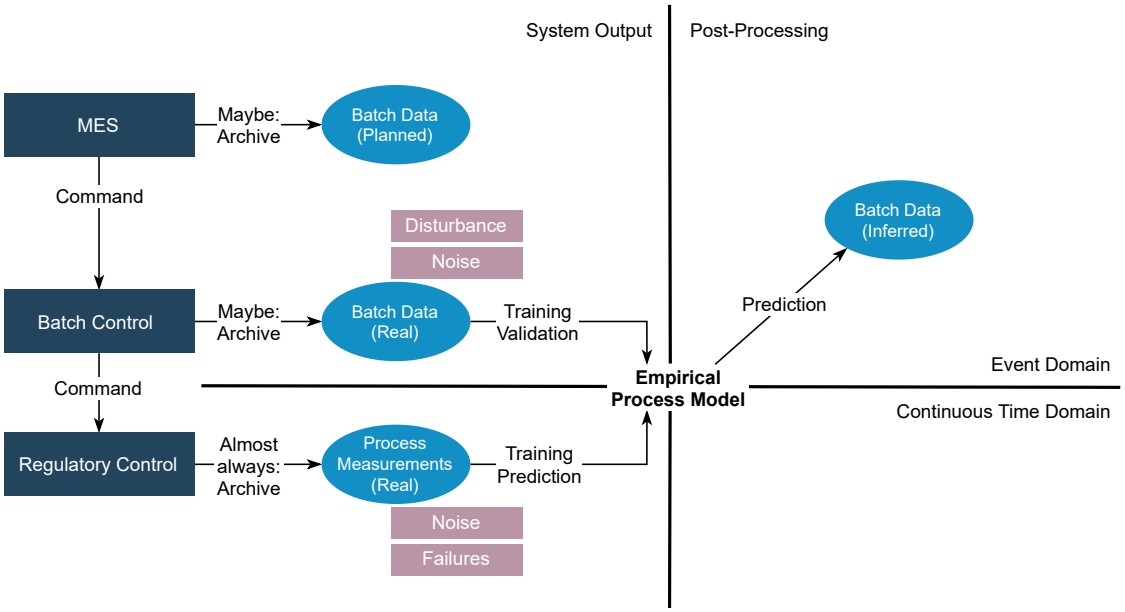


Figure 1.2: Overview of event and continuous time-domain simulations by comparing what type of batch data can be extracted.

Information about the start and end points of a production schedule is not always generated during production which turns into an obstacle to data analysis and process optimization (data-driven methods as well as mechanistic models). If these data are not generated during production, they need to be generated later on. One possibility to accomplish that goal is to rely on time-series data of process measurements which are much more frequently archived than batch data. Thus, ML algorithms can be trained using this information to make predictions and assist decision-making and process diagnostics.

1.2 Machine Learning

ML describes the ability of an algorithm to learn from data [3]. Due to the inherent repetitiveness of batch processes which are based on standardized recipes, there are reproducible behaviors among the data. In practice, they are often masked by process noise and disturbances. ML is expected to assist humans in generating predictions based on this time-series data which can be a very challenging task.

For the dissertation work, supervised methods of ML are used for the generation of interpretable batch data—the system learns by already being provided labels (check Figure 1.3).

ML performance is affected by several factors, decreasing when there is noise within the data (transient, stochastic, and discrete), low amount of quality data to train the algorithm, and disturbances in the

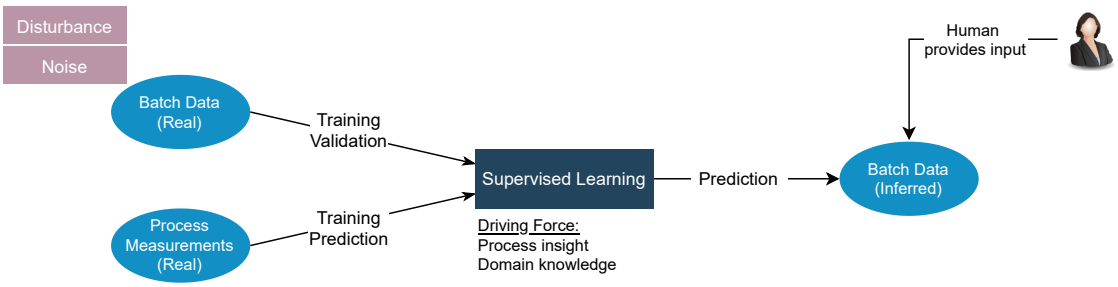


Figure 1.3: Overview of supervised learning approach to generate batch data.

process (wrong operator interventions, breakdowns, unknown specifications of raw materials, among others) [3].

1.3 Challenges

Besides all differences already presented between batch and continuous processes, a comparatively large effort is needed for model-based process optimization given the dynamic character of batch processes and their complexity (sequence of operations with possibly some occurring in parallel). As a result, the development and validation of dynamic models are often expensive or even impossible for batch processes in contrast with continuous processes, which have been plainly studied for several years [21]. For multi-product and multipurpose batch plants, due to the diversity of products, such an effort might not be economically feasible, especially if the life span of products in the market does not exceed a couple of years [14].

Other challenges might appear in model-building not just due to the complexity of batch process plants. Incomplete monitoring and some model-uncertainties and complex manual operations can create reasonable doubts about the quality of data acquired after implementation [14]. Batch systems are also computationally more expensive to study since it involves solving numerical differential equations due to the absence of steady state [22]. In this case, the process proceeds from an initial to a final (different) state. Along these lines, chemical processes are nonlinear which also makes it more difficult for an ML algorithm to predict behavior.

In practice (in real production data), it is very difficult to know which challenges a data set contains. There are too many disturbances, noise in parallel, and the origin of the data is not fully known. Hence, the *main ambition of the thesis*: develop a batch process that serves as a benchmark model for the development and study of data-driven techniques, in particular, ML. Benchmark models have enabled the development and extensive testing of algorithms such as the Tennessee Eastman problem [23]. This motivates the contribution which introduces a benchmark model for the development and testing of ML algorithms for batch tracking and batch phase-detection problems. By working with simulated data

where there is full control of the active challenges, disturbances, and noise, a study is made focusing on what makes an ML algorithm function and fail.

1.4 Work Plan and Thesis Structure

The ambition of the thesis is to build a simple base knowledge of Machine Learning, Batch Processes and Process Modelling and Simulation. After a literature study, the practical works begins by elaborating a process recipe for the batch process serving as the benchmark model. Keeping in mind some of the challenges of the ML algorithms already discussed, implementation is such that, already on the disturbance-free model, some of the challenges are already tackled. On the scope of the Bayer project, it is requested to identify disturbance scenarios to model on both batch and time-series data using the frames of Simulink® and Stateflow®. While increasing the disturbance level and complexity, the results are compared with the reference model (with no disturbances). By the end, there should be a clear understanding of the phenomena of batch processes that hinders the trustability of an ML algorithm in predicting scheduling sequences.

The benchmark process is described in Chapter 3 after providing the base knowledge required for the understanding of the implementation in Chapter 2. The disturbance mapping and functionality of the simulation with and without disturbances are provided in Chapter 4 followed by a review of the ML algorithm feasibility study. Conclusions, challenges, and further suggestions for implementation strategies are discussed in Chapter 5, hereafter the work is concluded.

The primary benchmark model studied follows a single-path structure where material follows a unique path. The implementation in Stateflow® is provided in Appendix A.1.1. Nonetheless, a multiple-path structure Benchmark Model is introduced in Appendix A.2.1—although not the focus of the thesis, this new benchmark process model is fully described allowing to the user proceed to such implementation.

2

Background

Contents

This chapter provides a background on Batch Processes and ML, naturally restricted to applications from these topics made in the following chapters.

2.1 Batch Process Control

By definition, a batch process leads to the manufacturing of a finite amount of material during a finite amount of time, following a specific recipe or *production schedule* [6]. In practice, disturbances and noise provoke constant changes in scheduling, requiring the process variables and the sequence of operations to be closely controlled to ensure product compliance and customer satisfaction with cost-efficiency [13].

As batch processes and ML are two separate fields of research, precise language should be used. Thereby, relevant definitions issued by the ISA (International Society of Automation) are presented in the following. The ISA-88 norm for batch control distinguishes between a process-perspective (conversion, transport, or storage), a physical-perspective (equipment/machinery), and a procedural-perspective (temporal decomposition). The temporal hierarchy (also denoted as procedural-perspective) holds the majority of nomenclature used in the next chapters and incorporates [24]:

- *Procedures:* the highest level in the hierarchy and defines the strategy for carrying out major processing activities such as making a batch. A procedure consists of an ordered set of Unit Procedures;
- *Unit Procedures:* These are connected to one piece of equipment and consist of several sequential operations. In a Unit Procedure, only one operation is presumed to be active at any time. An operation is carried to completion in a single unit. However, multiple unit procedures of one procedure may run in parallel (each in different units);
- *Operations:* order set of phases that defines a major processing sequence—the material being processed moves from one state to another (usually involves a chemical or a physical change);
- *Phases:* the smallest element of the procedural control model. A phase can issue one or more commands or cause one or more actions, such as conducting operator authorization checks, reading process variables, setting or changing alarms, among others.

Typically, a batch passes through several physical units during processing. The sequence of units will be called a *path*, and the task of following a path through the system will be called **batch tracking** [24]. Two cases can be argued when discussing batch tracking: **single-path** or **multiple-path structure**. A *single-path structure* is a group of units through which a batch passes sequentially. Multiple raw materials are typically used and multiple finished materials may be generated. For a single-path structure, several batches may be in progress at the same time. On the other hand, for a *multiple-path structure*,

several fixed trajectories exist in parallel [24]. For this dissertation, the two cases will be discussed with particular attention given to a single-path structure.

Batch processes are controlled by hierarchical systems, with a general structure provided in Figure 2.1, where [1, 2]:

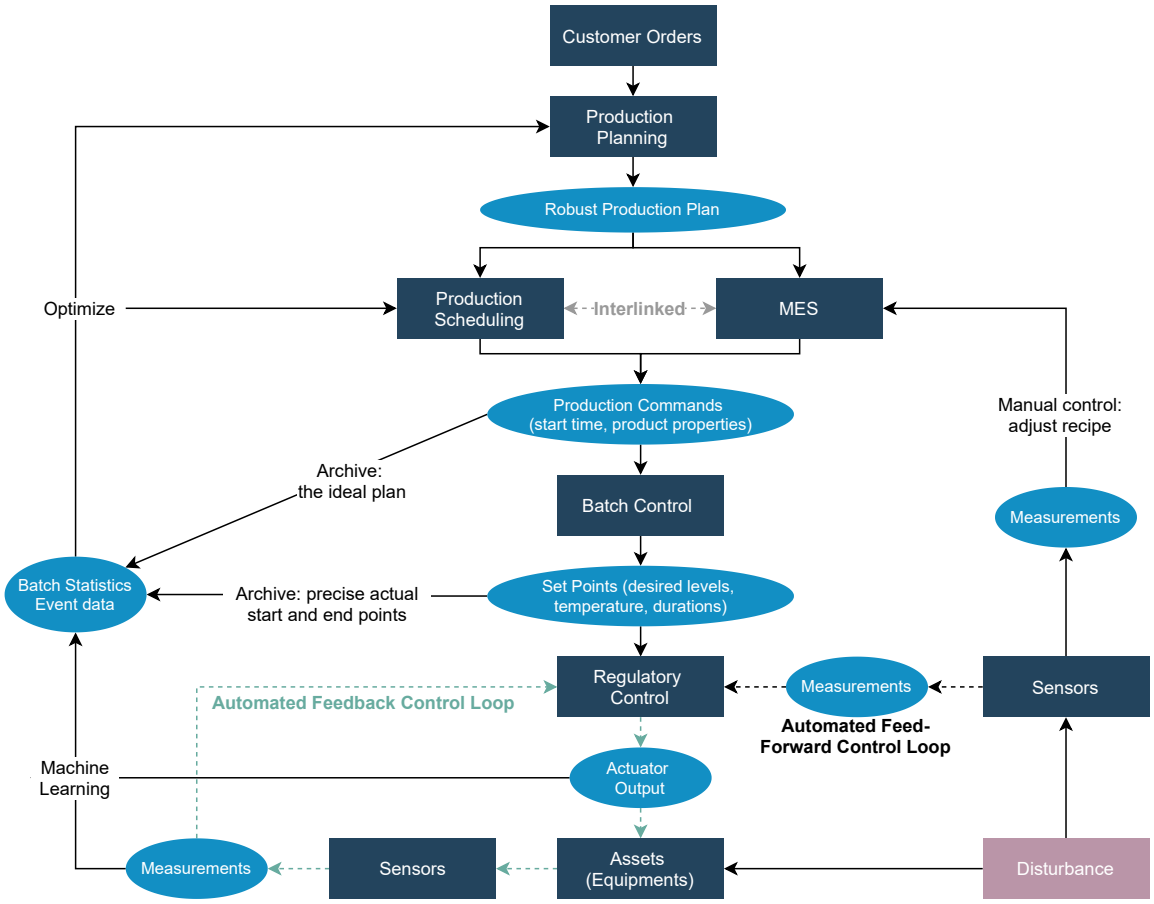


Figure 2.1: Hierarchical Control of a batch process. Adapted from [1, 2].

- *Production Planning:* includes tasks of Business Planning and logistics with a broad of activities, such as establishing the basic plant production schedule and modifying the basic plant production schedule for orders received, based on resource availability changes, energy sources available, power demand levels, and maintenance requirements.

- *Production Scheduling & Manufacturing Execution System:* areas of Manufacturing Operations and control which includes: reporting on area production including variable manufacturing costs, performing data collection and offline analysis as required by engineering function, and modifying production schedules to compensate for plant production interruptions that may occur.

- *Batch Control - Top Control Layer:* implements the set points that define product quality and

amount based on the recipe (can be supplied by the MES depending on how automated the process is). For the most part, is usually a *feed-forward* control.

- *Regulatory Control - Bottom Control Layer:* reacts to disturbances. The most established method is *automatic feedback control* - faster than feed-forward control. Feedback control measures the output of a process (sensor), compares it with the set point (controller), and then adjusts one or more input variables automatically to get the desired output value (actuator). A functioning control system increases the reliability and reproducibility of processes.

As decisions in planning and scheduling are mainly based on *batch data*, the information about the batches duration and their variability is key to the production planner [13, 18]. Batch data and time-series information are stored as binary, integer, float, or string values associated with a timestamp.

2.1.1 Sensors

In the chemical and biochemical industries, temperature, flow rate, pressure, level and weight sensors need to be installed to monitor the processes and enable safe operation. Table 2.1 summarizes the type of sensors commonly used, and also expresses the typical causes of noise to be found [7].

At this point, it is convenient to distinguish between noise and disturbances. While **disturbances** are meant to be abnormal behaviors that deviate the operation from the expected course, requiring the correction by process control; **process noise** is naturally occurring fluctuations in instrumentation signals and in the process (e.g., agitation or solids being added through a hatch).

Table 2.1: Most commonly used measurement options for process control. Causes associated with process and sensor noise for each of the five process measurements [6–8].

Measurement	Type of Sensors	Causes of Noise
Temperature	- Resistance temperature detector - Thermocouple	- Time delays - Multiple thermal capacities - Compression
Flow	- Orifice Plate - Venturi - Thermal mass - Turbine	- Pump vibration causing stream turbulence - Valve malfunction - Slugging - Foam formation
Pressure	- Strain gauges - Diaphragm - Optical fiber	- Flow changes across a junction - Cavitating pumps - Slugging - Air leakage and air blockage

Table 2.1 continued from previous page

Measurement	Type of Sensors	Causes of Noise
Level	- Float-activated - Head devices - Electrical	- Splashing and turbulence of liquid entering the tank (or agitation) - Changes on or above the surface of the liquid (p. e. foam formation)
Composition	- Gas-liquid chromatography - Mass spectrometry - Magnetic resonance analysis - Infrared spectroscopy	Time delays

2.1.2 Disturbances

In a realistic setting, industrial processes are subject to disturbances and uncertainties affecting the normal practice of an industrial site [1]. These disturbances can result from a variety of sources, including external environmental variables, as discussed in Table 2.2; and, the occurrence can be random or have an underlying pattern (systematic).

Table 2.2: Type of disturbances and possible causes associated with them [9–11].

Type of Disturbances	Possible causes
Deviations in production, changeover and cleaning times	- Unknown specifications of input material - Changes in operating conditions - Unavailability of resources (such as equipment) - Equipment malfunction - New raw material on site
Deviations in product specification	- Varying consumer demand - Unknown specifications of input material - Unknown side reactions - Tuning between shifts (operator error)
Equipment failure or malfunctioning	- Aging/Clogging of equipment - Wrong operator interventions - Fouling
Erroneous sensor readings	- Faulty calibration - Presence of solid material, ice, or bubbles in a line

Because the focus of the thesis is to model the phenomena which make process monitoring/fault detection needed, a brief introduction to each topic is introduced.

2.1.3 Fault Diagnosis

Fault diagnosis is most useful, not only for the detection of faults as a batch progresses, but also for revealing whether or not a specific batch belongs to the desired or normal behavior [9]. In essence, provides the knowledge to identify disturbed or normal batches. Fault diagnosis (FDI) is divided into three steps: fault detection, fault isolation, and fault identification. *Fault detection* takes place when the process deviates significantly from a desired standard trajectory, and identifies the presence of faults in a system and their times of occurrence. Followed by *fault isolation*, the second step determines the type and location of the faults. At last, *fault identification* aims to determine the size and time-varying behaviour of the faults [9, 25].

Normally, three types of faults can be distinguished, according to the part of the system they affect [25]: *sensor fault*: abnormal variation in measurements—once a faulty sensor is detected and identified, it can be either reconstructed or replaced; *actuator fault:* malfunction on a device that eventually perturbs the system dynamics; and, *process fault:* changes in internal parameters of the system that modify its behavior.

Frequently, the behavior of a fault is **highly unpredictable** and changes may be abrupt (discontinuities of the process), incipient (gradual), or intermittent [25]. To protect batch manufacturing from uncertainty and disturbances is highly recommended to use **simple monitoring**. However, another important use of monitoring is verifying the process performance relative to the behavior represented in the plant model used for control design [9].

Many of the issues with sensor faults are related to the process data used for the software sensor building. The challenges with process industry data are mainly from missing values, data outliers, drifting data, data co-linearity, sampling rates and measurement delays [12]. Table 2.3 presents some causes and strategies associated with three of the five problems with process data from the software sensor development and maintenance point of view.

Table 2.3: Causes and strategies associated with several problems with process industry data [12].

Problem	Causes	Strategies
Missing values	- Failure of hardware sensor - Problems in maintenance and sensor removal - Transmission of data between sensors - Errors in the database - Problems in accessing the database	- Replace the missing values with the mean values of the affected variable - Skip the missing data - Reconstruction of the missing values for them to be dependent on other variables

Table 2.3 continued from previous page

Problem	Causes	Strategies
Drifting data	For process drifts: - Changes of the process or some external conditions - Changes in environmental conditions and raw materials purity For sensor drifts: changes in measuring devices	- Apply the moving window technique: the model is updated on a periodical basis using only a defined number of the most recent samples - Adaptation of the sensor without performing any corrective actions to the process - Recalibration of the measurement devices
Data colinearity	Partial redundancy in sensor arrangement	- Transforming the input variables into a new reduced space with less colinearity - Select a subset of the input variables which are less colinear (feature selection)

Modelling faults allows improving the reliability and robustness of the plant by making new decisions based on such variability [8]. This is the main purpose of using fault detection which is similar to the Tennessee Eastman process plant [23, 26]—a widely used simulator for the development and validation of different control and monitoring strategies.

2.2 Modelling & Simulation in Batch Processes

Mathematical modelling is at the core of process control and optimization, since models of unit operations and tasks are needed to predict the process performance for decision-making support. Simulation models can be classified by their nature (empirical vs first-principles) or if they are in the form of continuous-time, discrete-event, or hybrid kind of models. Table 2.4 describes some characteristics, advantages, and disadvantages of each model.

Table 2.4: Characteristics of different simulation models: continuous, discrete, and hybrid [13].

	Continuous-Time	**Discrete-Time**	**Hybrid**
Description	Described by systems of differential equations	Used in planning and scheduling of projects where variables are defined at discrete points in time only	Has both discrete and continuous dynamics. Higher-level behavior of complex automated systems
Software Environment	Simulink	Stateflow	Stateflow and Simulink

Table 2.4 continued from previous page

	Continuous-Time	Discrete-Time	Hybrid
Advantages	Allows to visualize how a chemical reaction evolves based on predefined variables	The system behavior is analyzed at fewer discrete points in time. Model execution is fast and has a robust convergence. It also provides a reference grid of time for all operations competing for shared resources, such as equipment items	Allows to capture the interaction of event-driven and time--driven dynamics
Disadvantages	Extensive data needed to calibrate, validate and run a simulation	Large combinatorial problems of intractable size, especially for real--world problems, and hence limits its applications	Higher computational capacity

Taking into account the characteristics observed for each model type, hybrid simulation stands out. A fast and robust execution allows studying a wide range of scenarios in a short time which have proven to be effective and resourceful when dealing with batch scheduling [13, 27].

2.2.1 Challenges in model-building

Due to several characteristics of batch processes, model-based process optimization requires a greater effort when comparing it with continuous processes. Those include: the dynamic character of batch processes and their complexity—phase transitions as well as batch scheduling, planning and tracking; incomplete monitoring; model-uncertainties, and complex manual operations which can create reasonable doubts about the quality of data acquired after implementation; and, the non-linearity of most chemical processes [14, 22].

2.3 Modelling of Batch Processes with Machine Learning

Since the development of the algorithm is not the thesis focus, here only a brief overview of ML is given. ML are comprehend basically algorithms and tools that allow a computer to learn from data. With ML, a computer can act without being explicitly programmed to do so [3]. In this thesis, ML algorithms were chosen and applied to perform classification, such as **correctly identification of labels and recognition of the recipe**, especially when subjected to disturbances and noise.

For a certain problem, an algorithm is developed that automatically (considering the data gathered and prepared) can achieve a solution. All of the examples for a system to evaluate are called *test set*. The ML process can be divided into seven steps, as shown in Figure 2.2, wherein: **Step 1 - Data**

Collection: Define scopes and goals concerning the data gathered; **Step 2 - Data Preprocessing:** Adjusting and manipulation: normalization, duplication, and error correction if needed; **Step 3 - Choosing a model:** Select the right ML algorithm; **Step 4 - Training:** Using data to incrementally improve the model's ability; **Step 5 - Validation:** Allows testing model against data that has never been used for training - **data set**, as previously stated. Usually, the sample data is divided up to 80% testing set and 20% data set; **Step 6 - Hyperparameter or Parameter Tuning:** Further improvement of the training set. Depends on the specifics of the data set, model, and training process; **Step 7 - Prediction:** The final step of the process where the model will hopefully answer some questions stated at the beginning of the problem [3].

Figure 2.2: Steps of a ML process. Adapted from *Google Cloud Tech.*

After evaluating the solution, if the results obtained were not satisfactory, the next step would be to analyze the errors. The ML approach is an iterative process and it only ends when there is a positive evaluation. Nevertheless, if the machine is not capable of learning, the process ends when the positive evaluation is not achieved. This is one of the biggest challenges: using an algorithm that is bound to fail and only be aware on Step 5.

Nevertheless, this process can be **automated** to achieve a better solution where the algorithm learns by itself. There can always be room for improvement, so the data is updated and from those results, the algorithm can suffer some changes.

ML systems can be broadly divided into: **How they are trained:** classified according to the amount and type of supervision they get during training. In this case, if exists or not human supervision; **How they learn:** classified according to the possibility (or not) of the system to learn incrementally from a stream of incoming data; **How they work:** classified according to the form the systems generalize: interpolation or extrapolation [3]. As seen in Figure 2.3 there are several subcategories in respect to the system classification.

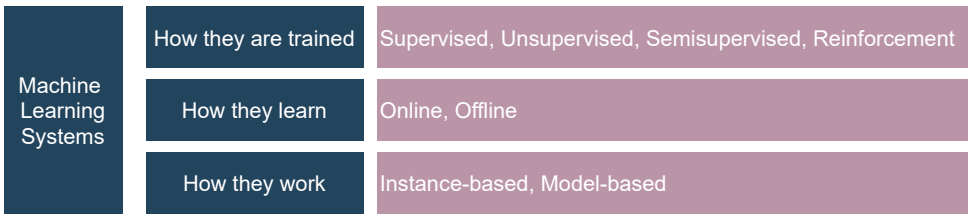


Figure 2.3: Classification of ML Systems. Adapted from [3].

For the thesis, the focus remains on supervised and offline learning. The algorithm learns from the simulated data where the labels are already provided.

2.3.1 Challenges of Machine Learning

Generating batch statistics based on the time-series data of the process is a challenging task for the ML algorithm. The purpose of Section 2.3.1 is to grant enough knowledge of the main issues when implementing an algorithm.

Seeing that the main task is to select a learning algorithm and train it with some data, there are two main challenges: *bad data* and *bad algorithm*. Other problems that might occur are a defect in *data frequency* or even *inconsistency* in the data obtained. Here are some examples of **bad data** [3].

1 - Insufficient Quantity of Training Data fewer data results in lower accuracy. The main solution is to acquire more data.

2 - Nonrepresentative Training Data by using a nonrepresentative training set, the model is unlikely to make accurate predictions. This is true in both instance-based and model-based learning. It is often hard to train a set that represents the cases to be generalized. If the sample is too small, the result is *sampling noise*; if very large they can be nonrepresentative if the sampling method is flawed - *sampling bias*. For that reason, noisy samples should be removed.

3 - Poor quality data training data is full of errors, outliers, and noise. Sensors can give fictional values that need to be identified and cleared. In the case of labels, misspelled labels may occur which artificially generates false new labels, thus biasing classification. Accordingly, the system will have some difficulties performing well at recognizing hidden patterns. One way to fix the problem is to spend time cleaning up the training data:

- *Reduce the dimension of the training data* by removing instances that are clearly outliers; or fix the errors manually;
- If there are missing features on the training data, it is possible to *ignore those instances* or fill in the missing data with *average values*. Another alternative is to *train two models*: with and without the feature.

4 - Irrelevant Features as already seen, having missing features on the training data leads to poor performance of the system. The system can only learn if there are enough relevant features and not too many irrelevant ones. One of the biggest challenges on ML is creating a good set of features to work on. Finally, it is shown some examples of **bad algorithms** [3].

1 - Overfitting the training data *difficulty on generalizing well*. This happens when the model is too

complex relative to the amount and noisiness of the training data. Overfitting is detected by identification and statistical significance tests. There are two main solutions for this problem:

- *Simplify the model:* by having fewer parameters, by reducing the number of attributes in the training data; or by constraining the model (also called *regularization*);
- *Collect more training data*;
- *Reduce the noise* in the training data – as seen on *Poor quality data* – by removing outliers, noise, and errors.

The amount of regularization can be controlled by a hyperparameter. As already stated, it allows for further improvement of the training set. If the hyperparameter regularization is set to a very large value, as result, the model is flat (a slope close to zero) which results in sensitivity loss. The learning algorithm will almost certainly not overfit the data, but it will not find a good solution either [3].

2 - Underfitting the training data Exactly the *opposite of overfitting* - happens when the model is too simple to learn the underlying structure of the training data. Some solutions:

- Select a *more complex model* by adding more parameters;
- Feed better features to the learning algorithm – *feature engineering*;
- *Reduce the constraints* of the model by reducing the regularization of the hyperparameter.

In conclusion, the system will not perform well if the training set is too small, the training data is nonrepresentative, full of errors and noisy; or if it is bursting with irrelevant features. Lastly, the model should not be too complex (risk of overfitting) or too simple (risk of underfitting) [3]. A few challenges of ML might also be due to sensors data collection. As such, Table 2.3 should also be consulted.

2.4 Methodology

Due to the inherent repetitiveness behavior of batch processes, there are reproducible behaviors among the data. In practice, data is often masked by process noise and disturbances [9]; for that reason, it is expected that ML algorithms can assist humans in generating batch statistics based on this time-series data.

The most common and efficiently proven algorithms to work with time-series data in supervised learning are **Random Forests** (RF) and the **Hidden Markov Model** (HMM) [28, 29]. The following section briefly analyses both algorithms, including their differences in implementation, accuracy, and limitations. From Figure 2.4, changes between the two algorithms at the level of implementation and dealing with the training data are noticeable: while the HMM deals with categorical features, RF uses continuous features [29]. The learning remains supervised since labels are provided for the training.

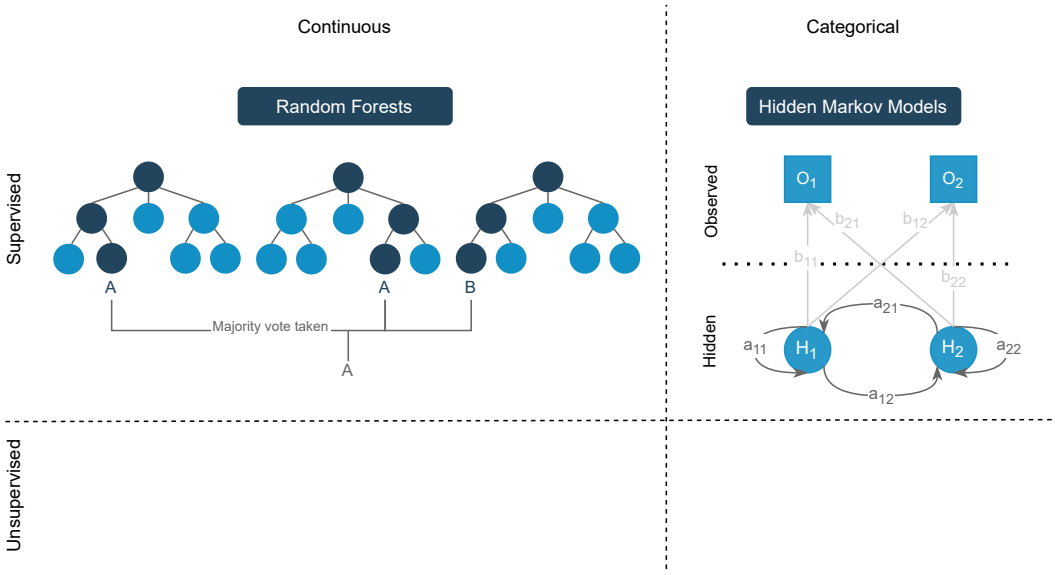


Figure 2.4: Classification of HMM and RF by type of learning and supported data.

2.4.1 Hidden Markov Model

The HMM is a *probabilistic model of time-series data*. Based on the likelihood of a specific event to occur, this model derives from the principle that two consecutive states are dependent on one another [29]. The HMM has been linked to several applications in speech recognition and the process industry [30]. More precisely, an HMM consists of three parts [30]:

1. A finite number of states *(N states)* where the signal possesses some measurable and distinct properties;
2. At each time step, *t*, the state changes according to a transition probability function, depending on the previous state. Note that the transition may be such that the process remains in the same state;
3. After each state transition, an output is generated according to the probability function. This distribution is fixed and does not depend on how the state was reached. As a result, there are *N observable probability functions*, each of which represents a stochastic process.

The following introduces the model notation, where H, O, A, B represent parameters set in Figure 2.4 [30]: T, number of observations; N, number of states; M, number of possible observations at the output; $H = h_1, h_2, \ldots, h_N$ states; $O = o_1, o_2, \ldots, o_N$ set of possible outputs; $A = \{a_{ij}\}, a_{ij} = P(h_j(t+1)) \mid h_i(t)$, probability distribution of the state transitions; $B = \{b_{ij}\}, b_{ij} = P(o_j(t+1)) \mid o_i(t)$, probability distribution of the outputs in state j; $\pi = \{\pi_i\}, \pi_i = P(h_i(t=1)$, initial state distribution.

Compared to other time-series classification methods, the HMM makes classification decisions de-

pending on previous and subsequent time steps. It can thus place the current time step in the context of the entire time-series and globally decide which series of states is most likely for the given sequence [30]. Three parameters are essential for this model: probability distributions at the exit for each state, transition matrix, and, the initial distribution of the states.

1—Probability Distributions at the exit for each state The phases and operations of the benchmark process are represented by the hidden states (H_1 and H_2). This means that to each phase, a probability function that describes how the measured values behave is assigned. These probability functions are time-independent, which means that part of the information from the measured values is lost, namely the time-dependent behavior of the process within a phase. Information about the sequence of observations is displayed in the HMM via the various hidden states [5].

For the probability distributions of the states, it is assumed that the output dimensions are independent of one another and that each dimension has the same weight/importance when recognizing the states. The multivariate probability distribution, P, of a state corresponds to the combination of the univariate probability distributions, P_i, of the individual dimensions:

$$P(h) = P_1(h_1) \times P_2(h_2) \times \ldots \times P_d(h_d)$$

$$x = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_d \end{bmatrix}$$

In essence, $N \cdot d$ histograms will be provided for these distributions (where d represents dimension). In practice, the training of the model begins with the training data being divided up according to the states; afterward, for each state, the data is split again according to dimensions and, counted how often each possible value occurs. Lastly, the results are summarized, for each state, in a probability distribution. Note that the HMM can not handle both discrete and continuous variables being necessary to **discretize continuous values**.

2—Transition Matrix It describes the probability of transitions between states (can be either remain in the same state or move to another). For example, a transition probability can provide such information as the following: "if the model is in state A, there is a 60% chance that it will remain in this state for the next time step, and a 40% chance that it will move forward to state B. If there is a 0% probability for a transition between A and C, such path proves to be impossible according to the model". In summary, the transition matrix is the part of the model where the recipe of the underlying process is learned.

3—Initial Distribution of the states The start distribution of the states must be specified for the HMM. In the given application, it is usually not known in which batch phase the measurement begins. For

this reason, an even distribution of each state is assumed so that a sequence begins with the same probability with each state.

2.4.2 Random Forests

A RF is an ensemble classifier, i.e., it structures an array of base classifiers from training data. This kind of ensemble technique has proven to be effective when dealing with labels easily disrupted with small disturbances [29]. The RF adopts the bagging technique to generate the model—it builds several decision trees, each one from the same initial data set, with a subset of decision trees. To grow each tree, random samples are injected into the model-building, creating a random number of prediction samples (last decisions: A, A, B from Figure 2.4) [31]. **The generation of a wide variety of decision trees allows to diminishing the partiality of the ensuing tree**. Each tree provides a classification, and each classification is logged as a vote. Afterward, the votes attained are counted, and the majority one is acknowledged as the fresh sample final classification/prediction [31]—case of prediction A from fig. 2.4.

RF have proven to be robust classifiers and regressors that do not depend on the scaling of the input data and can also map non-linear relationships and be used to select features [32].

In contrast to the HMM, the output of the RF only depends on the current input values. To compensate for this disadvantage, the number of available measured values for the RF is increased. This is implemented by introducing a **moving window** of size k. It allows the model to use the current and the last k measured values as a basis for the class decision. The input time-series is thereby shortened by k values for which no classification can be carried out. It is assumed, however, that this can be neglected for $k << n$. As a hyperparameter, the number of decisions trees and the size of the moving window are analyzed. The training takes place according to the procedure detailed in references [5, 31].

2.4.3 Evaluation

Several studies compare these algorithms in different categories: overall accuracy on labeling, classification speed, memory consumption, feature computation, and model complexity. A comparison between both algorithms is provided in Table 2.5 [29].

Table 2.5: Summary performance of HMM and RF based on a study regarding internet traffic (+ is worse than ++).

Category	HMM	RF
Overall accuracy	50-96%	>85%
Classification Speed	+	++
Memory Consumption	+	++
Feature Computation	Low	High
Complexity	+	++

The accuracy offers a direct statement about the prediction quality of a classification method. In this particular study, the user could be inclined to choose the RF algorithm since the overall accuracy is higher. Nonetheless, not even one of the literature studies was regarding time-series data and labels sequence recognition—which becomes an important aspect for the feasibility study of the benchmark model. In other studies, when using sequence labeling with the HMM, accuracy levels of 96% are achieved [33], and neural networks, 97% [34]. Since the dissertation task is identical to the studies above mentioned, one could also expect similar results.

Even though the RF is a great classifier, the simulation of the batch process will run with several labels and specific transitions/relations between them. This dependency between states can not be recognized with a RF.

According to [5], three criteria are used to evaluate the learning of the algorithms: Accuracy, Confusion Matrix, and Mean Absolute Error (MAE). First, **accuracy** which is given by the number of correct labels divided by the total number of labels. Secondly, a more in-depth analysis of the results obtained is possible if a **truth matrix** (confusion matrix) is established. It allows depicting the states that frequently are confused and wrongly labeled. However, this evaluation is more time-consuming than with the accuracy. At last, the **MAE** is introduced. Whether a time step represents a change point or not leads to a binary class decision. The evaluation based on parameters such as accuracy and precision can be troublesome since even an error of one time step leads to a worse result. It is, therefore, easier to evaluate errors in the identification of change points using the distance between true and estimated change points. Within the scope of the thesis, the MAE should be considered, where:

$$MAE = \frac{\sum_{i=1}^{\#CP} \mid Predicted\ CP - Real\ CP \mid}{\#CP} \tag{2.1}$$

A discussion of future work and milestones achieved using both these algorithms is introduced in Chapter 5.

3

Process Model Simulation

Contents

In Chapter 3, the batch process model used to test the ML algorithms is presented, as well as the disturbances considered to generate the various simulated scenarios related with industrial experience. Section 3.1 describes the main characteristics of the process, Section 3.2 shows the computational framework developed to conduct the simulations, while Section 3.3 illustrates how the implementation was done in both continuous and discrete modelling environments. Posterior to the implementation of the benchmark model without disturbances, a new section is introduced, providing the knowledge necessary to enforce the benchmark model with disruptions and interferences—from Section 3.4 onward.

3.1 Benchmark Process Model

The benchmark case study consists of a batch process with *filling, processing, draining, and cleaning* operations wherein the entire process is viewed at the unit level. That is, all operations occur in a single vessel called Unit1_Vessel1, as depicted in the process flow diagram of Figure 3.1. Three liquid raw materials (Educt1, Educt2, Educt3) and one solid inputs are used in the reaction, whose loads are valve-controlled; and, solids are added through a hatch placed on top of the vessel. The process model here presented is purely conceptual wherein the goal is to have a working benchmark that captures the complexity of batch operations, featured by time-dependent operations, phase transitions, and disturbances of different nature.

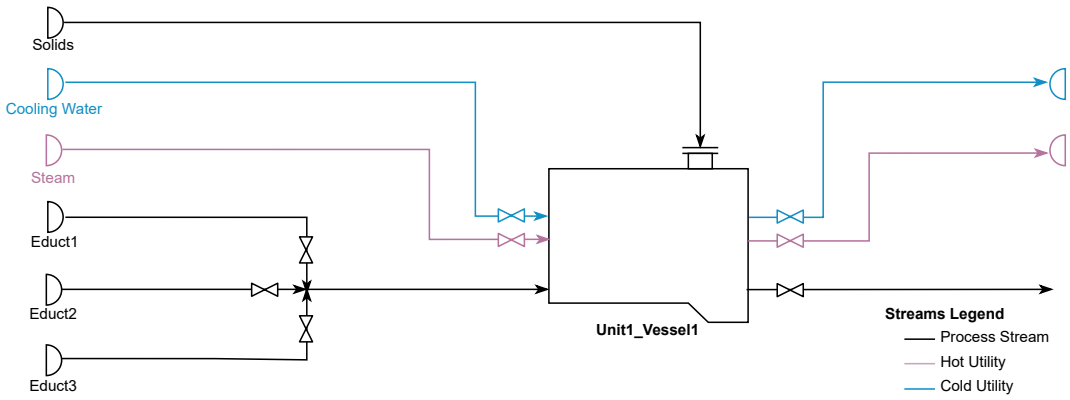


Figure 3.1: Process Flow Diagram of the Reference Model.

Besides raw materials, utilities are necessary to obey the process design parameters and batch recipe. It is considered, on the benchmark process model, that the reaction requires a hot utility to achieve the desired gradient of temperature. Although no numerical values of gradients are given, medium pressure steam is applied (since being one of the most common hot utilities in a chemical site). Following the reaction, the vessel is assumed to be cooled using water as the utility. Because most operations follow a low-temperature gradient, it is of the utmost necessity to decrease the temperature of the vessel to assure the proper operating conditions. There are abstract streams where utilities are

used for the simulation to be as close to reality—with utilities, temperature gradients, raw materials, among others.

A batch process is characterized by following a specific recipe. As such, Figure 3.2 illustrates the path, operations (dark blue), and phases (light blue) for the chosen reference process following the nomenclature according to the ISA-88 norm. The 10 operations considered for a batch, include all the filling, processing, and draining operations. The cleaning procedure, after each batch, is given by the last operation. The production of a single batch takes roughly 19.5 hours, and the CIP, 1.4 hours.

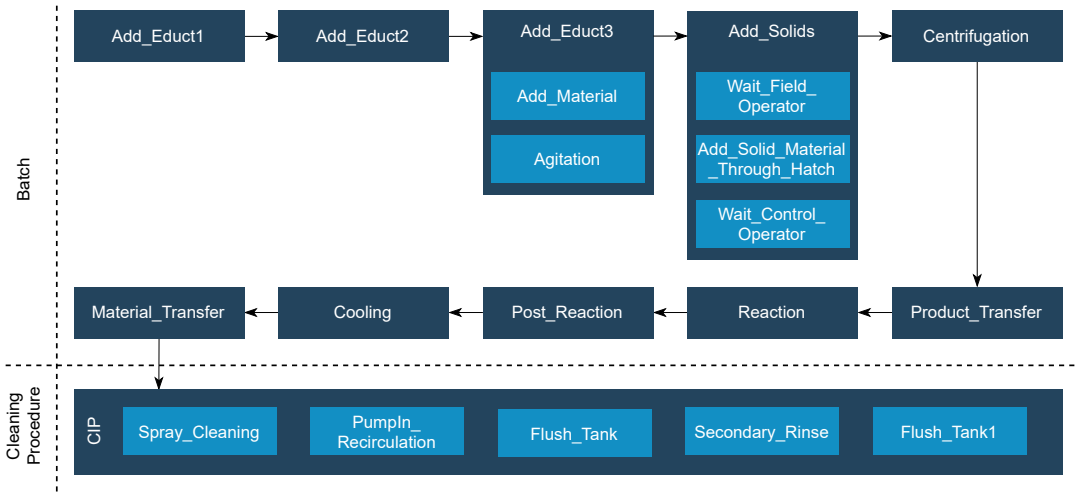


Figure 3.2: Benchmark Model Recipe including all operations and phases of a single batch.

Since the focus is batch phase identification, it is not the task of the ML algorithm to model heat and mass balances. For that reason, **chemical reactions, and kinetics were omitted**. Simulations are, for that reason, limited to mass balances and vessel height (modeled variables are therefore vessel contents and their sensors, as well as valves, flows, and the hatch position). In this way, the simulation executes much faster and can be used for massive simulation studies.

While the process just described is apparently simple, complexity rises when mathematical expressions are added to emulate different behaviors at the vessel level. Instead of having only linear profiles, the vessel level can follow an exponential, step, and stair function. Thus, ML algorithms can be used to predict variables profiles, identify phase transitions and label recognition. As discussed, ML algorithms have some difficulty in recognize non-linear behaviors, hence adding these types of level profiles.

After establishing the recipe, it is necessary to specify duration and volume conditions for the model implementation. For this purpose, the flow rate is adjusted as a degree of freedom to connect volume with time. Characteristics of the process such as level profile and conditions can be found in Tables 3.1 and 3.2.

Table 3.1: Operations and phases in the benchmark process model including transition trigger conditions, nominal durations, and level profiles.

Event Label		Event ID	Type of Operation	Transition Trigger Condition	Nominal Duration	Level Profile
Add_Educt1		1	Filling	L1_PV >= 30% †	3 min	Linear
Add_Educt2		2	Filling	L1_PV >= 60%	3 min	Linear
Add_Educt3	Add_Material	3.1	Filling	L1_PV >= 65% or after(5,min)	5 min	Linear
	Agitation	3.2	Filling	L1_PV >= 70%	5 min	Noise is added to the signal ‡
Add_Solids	Wait_Field_Operator	4.1	Filling	after(15,min)	15 min	Linear
	Add_Material_Through_Hatch	4.2	Filling	L1_PV >= 85%	5 min	Noise is added to the signal
	Wait_Control_Operator	4.3	Filling	after(5,min)	5 min	Linear
Centrifugation		5	Filling	L1_PV >= 90% Design: 5 steps (each step with a fixed flow rate)	step increase: 3 min inactivity: 7 min overall: 42 min	Step
Product_Transfer		6	Draining	L1_PV <= 60%	3 h	Linear
Reaction		7	Processing	L1_PV >= 85%	12 h	Exponential
Post_Reaction		8	Processing	after(2,h)	2 h	Noise is added to the signal
Cooling		9	Processing	L1_PV <= 70%	1 h	Exponential
Material_Transfer		10	Draining	L1_PV <= 0%	5 min	Linear
CIP	Spray_Cleaning	11.1	Cleaning	L1_PV >= 10%	30 min	Noise is added to the signal
	PumpIn_Recirculation	11.2	Cleaning	after(14,min)	14 min	Noise is added to the signal

Table 3.1 continued from previous page

Event Label		Event ID	Type of Operation	Transition Trigger Condition	Nominal Duration	Level Profile
	Flush_Tank	11.3	Cleaning	L1_PV <= 0%	5 min	Noise is added to the signal
	Secondary_Rinse	11.4	Cleaning	L1_PV >= 5%	30 min	Noise is added to the signal
	Flush_Tank1	11.5	Cleaning	after(2.5,min)	2.5 min	Noise is added to the signal

† L1_PV corresponds to the level sensor, as a percentage of how much the vessel is filled.

‡ It is to be understood that the behavior of these phases can not be simulated only on the discrete environment. Noise is not meant to be corrected, it is part of the nominal profile of the process.

Table 3.2: Characteristics of the simulation.

Property	Value	Unit
Vessel Volume	12.5	m^3
Batch Duration	19.5	h
Cleaning procedure duration	1.4	h
Nominal number of batches	370	-
Simulation Time †	333 (7992)	days (h)
Inactivity Period ‡	5 min—2 hrs	-

† Operation year has 333 days, already leaving a month for cleaning and maintenance.

‡ The inactivity period corresponds to the time between the end of a cleaning procedure and the beginning of a new batch. It is set as a random duration between 5 min and 2 hrs differing on each new batch production.

3.2 Computational Framework for Simulation

Simulink® and Stateflow® are both visual programming languages. While Stateflow® is based on finite state machines and suited for discrete-event simulation; Simulink's primary applications are dynamic systems modelling and simulation. Combining discrete and continuous environments has proven to be effective and resourceful when dealing with batch scheduling [27]. As such, the process sequence as well logical behavior of operations/phases are implemented in Stateflow® and the continuous subsystem in Simulink®. Note that a hybrid system arises at the top part of Figure 3.3 when time-driven and event-driven dynamics are both present. The hybrid system may be deterministic or stochastic and it may be modeled in either discrete or continuous time [4].

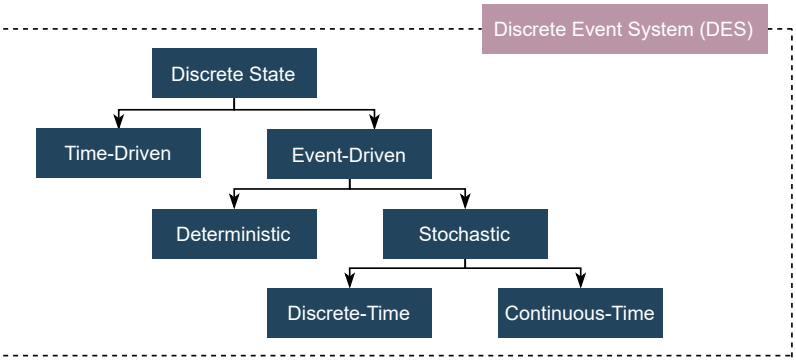


Figure 3.3: Minor system classification for discrete event simulations [4].

The simulation framework consists of continuous and discrete systems linked to a scenario control provided by a MATLAB script, as Figure 3.4 shows. The simulation is initialized in Simulink®/Stateflow® with the input parameters loaded in the MATLAB workspace. The simulation is then executed, and the output data is written in both continuous and discrete environments. If chosen, the output can be written back to MATLAB unless the user interrupts the simulation in-between. Therefore, using the script allows automating the execution of the simulation; to provide relevant instructions for the simulation, and manage variables input and output.

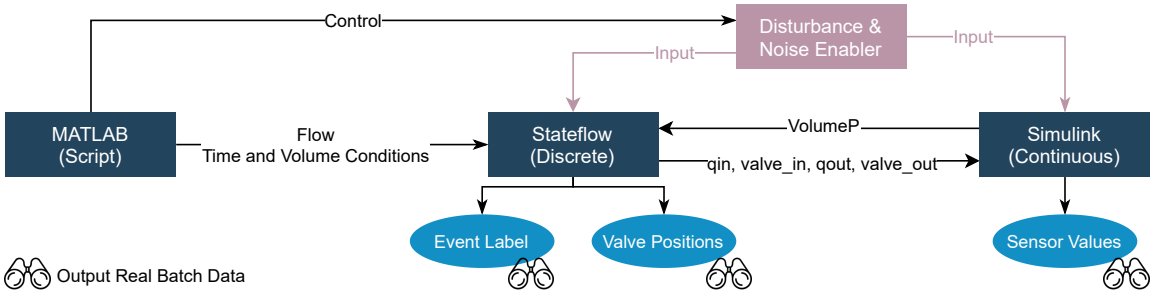


Figure 3.4: Computing Framework of the simulation.

Figure 3.5 presents the benchmark model as implemented in Simulink® and Stateflow®. The subsystems, highlighted with different color blocks, are described as follows:

- *Purple Block:* dynamics of the tank implemented on the continuous environment: mass balance, utilities and process material flows, and height; generates all sensor variables (noise and disturbance-free) after integration;
- *Orange Block:* identifies the noise that is added to the signal to simulate the behaviors of two phases: $Agitation$ and $Add\_Solid\_Material\_Through\_Hatch$; two operations: CIP and $Post\_Reaction$; and, the inactivity period between batch production;
- *Yellow Block:* complete discrete environment with logical transitions—implementation in Stateflow® in which phases and operations transitions are present;
- *Gray Blocks:* collects the output variables from both continuous and discrete environments.

Simulink® exports the sensor values, which, as in reality, are of continuous nature, while Stateflow® exports time start/end points of operations, valve positions, and event labels. Table 3.3 presents all the output variables of the simulation.

Table 3.3: Output variables in the discrete and continuous environments of the simulation.

Environment	Variable Category	Variable Label	Description	Type of Output Values
Discrete	Time Start/End Points	BatchID	Number of batches produced	Integer
		CleaningID	Number of cleaning operations	Integer
		EventFrameID_ Reference	ID of each phase and operation	Integer
		EventFrameLabel_ Current	Label of each phase and operation. With specific disturbances active, the label provides the information of which disturbance is enabled	String
	Valve Positions	Y1_Digital	Valve is only opened on Add_Educt1	Binary
		Y2_Digital	Valve is only opened on Add_Educt2	Binary
		Y3_Digital	Valve is only opened on Add_Educt3	Binary
		Y4_Digital	Valve is only opened on Centrifugation	Binary
		Hatch_Digital	Hatch opens on Add_Solid_Material_ Through_Hatch	Binary
Continuous	Sensor Values	L1_PV	Level Sensor. Percentage of how much the vessel is filled	Float (0 to 100)
		H1_PV	Height Sensor (given in meters)	Float

Table 3.3 continued from previous page

Environment	Variable Category	Variable Label	Description	Type of Output Values
		N1_PV	Motor Sensor (given in rpm) turned on from Agitation to Cooling	Float
		F1_PV	Cooling Water Flow Sensor (given in m^3/h) active on Cooling	Float
		F2_PV	Steam Flow Sensor (given in m^3/h) active on Reaction	Float

3.3 Benchmark Process Model Implementation

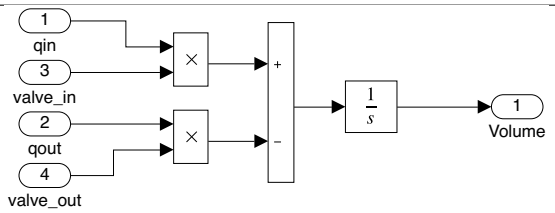
3.3.1 Dynamic Model of the Holding Vessel

For this particular case, the volume of the tank is calculated from a mass balance where incompressible flow and constant density at the vessel entry and exit are assumed:

$$\frac{\mathrm{d}Volume}{\mathrm{d}t} = valve_{out} \cdot q_{out} - valve_{in} \cdot q_{in} \tag{3.1}$$

Variables and parameters of Simulink® given in Table 3.4 are connected to Stateflow® both as input and output. By changing the binary value of $valve_{\mathrm{in}}$ and $valve_{\mathrm{out}}$, it is possible to control the liquid flow direction. For simplicity's sake, the valve positions were implemented in Stateflow® instead of Simulink®. Valves, pumps and pipes dynamics were not considered in this work, but these features can be later implemented for additional complexity.

Table 3.4: Representation of disturbances on the discrete environment.

Variables	q_{in} is the initial flow rate q_{out} is the exiting flow rate
Parameters	$valve_{\mathrm{in}}$ is a binary variable related to the flow rate into the tank $valve_{\mathrm{out}}$ is a binary variable related to the flow rate out of the tank
Input	q_{in}, q_{out}, $valve_{\mathrm{in}}$, $valve_{\mathrm{out}}$
Output	$Volume$
Representation	

Because the trigger transitions variables are mainly volumetric, the volume is transformed to a signal representing the vessel filling level (already stated in Table 3.1 as $L1\_PV$). This variable becomes the

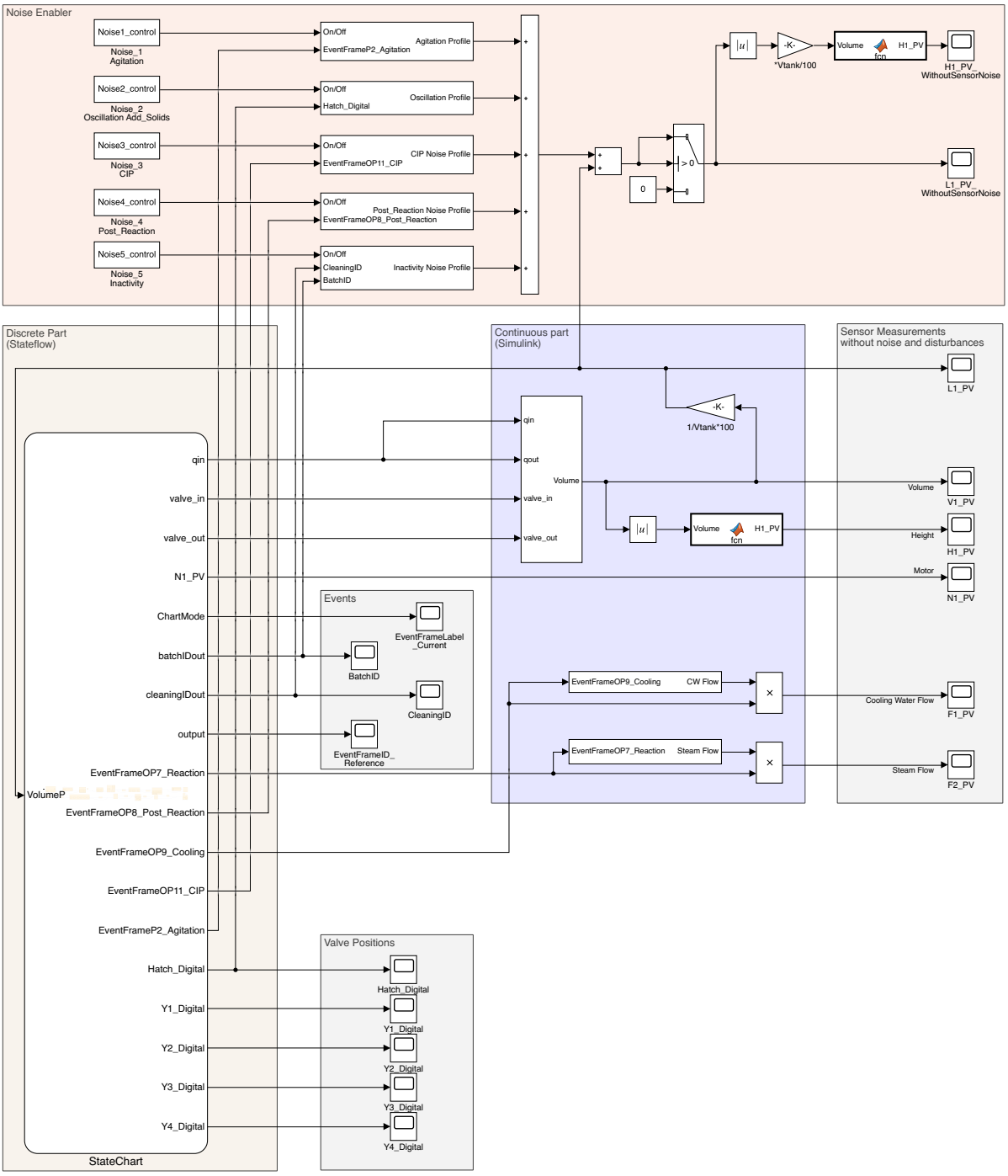


Figure 3.5: Representation of the benchmark model as implemented in Stateflow® and Simulink®.

sole input of the discrete environment and considers a vessel maximum volume of 12.5 m^3—consult Eq. (3.2).

$$Volume_P = \frac{Volume}{Volume_{vessel}} \cdot 100 \tag{3.2}$$

The solver used in Simulink® to integrate Eq. (3.1) was **ode1 (Euler)** with a fixed-step size of 10^{-3} which provides the adequate speed to run the simulation given its simplicity [35]. Even providing fast calculations, ode1 is less accurate and stiff when comparing to other solvers [27]. However, the solver can be changed by the user and studied if whether it makes a difference for the ML algorithm. To remark, this solver leads to a numerical error within an acceptable tolerance given the model's level of abstraction.

3.3.2 Heating and Cooling

Several operations have a discrete variable triggered when the state becomes active. To add more complexity to the model, instead of having a discrete variable such as a valve position on both $Cooling$ and $Reaction$, a dynamic variable is computed. Since there is some difficulty in ML to recognize non-linear profiles, the flow rates of heating/cooling agents are assumed to increase/decrease exponentially.

The implementation in Simulink® lies on the combination of two blocks: *Transport Delay* and *Transfer Fcn*. On the latter, a second-order function is implemented for the exponential behavior that lasts until the end of the operation. Since the operation duration differ, the *Transfer Fcn* blocks compute different functions—Tables 3.5 and 3.6. Nevertheless, having binary variables exported from Stateflow® (subsystems input: $EventFrameOP7\_Reaction$ and $EventFrameOP9\_Cooling$) that changes whenever a phase or operation occurs, narrows the behavior to those specific states. The parameters of the second-order functions were driven on a basis of trial-and-error.

Table 3.5: Variables, parameters, inputs and outputs to compute the steam flow rate.

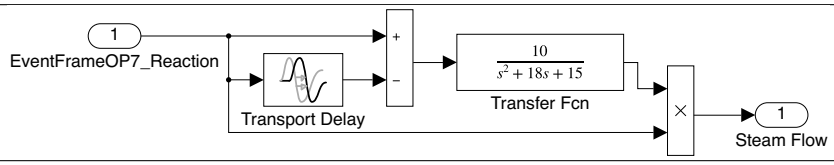
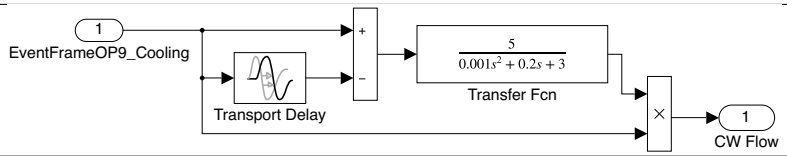
Variables	$Steam\ Flow$ is the steam flow of the process
Parameters	$EventFrameOP7\_Reaction$ is a binary EventFrame variable linked to Reaction $Transport\ Delay = 0.05\ h$
Input	$EventFrameOP7\_Reaction$
Output	$Steam\ Flow$
Representation	

Table 3.6: Variables, parameters, inputs and outputs to compute the cooling water flow rate.

Variables	$CWFlow$ is the cooling water flow of the process
Parameters	$EventFrameOP9\_Cooling$ is a binary EventFrame variable linked to Cooling $Transport\ Delay = 0.05\ h$
Input	$EventFrameOP9\_Cooling$
Output	$CWFlow$
Representation	

3.3.3 Height Sensor

For complexity purposes, the vessel is chosen to be curved—the height will have a logarithmic profile until a $Height\_Threshold$ value, followed by a linear behavior (Figure 3.6). In Simulink®, the height sensor is given as a function of the vessel volume with a $Volume\_Threshold$ of 6.25 m^3.

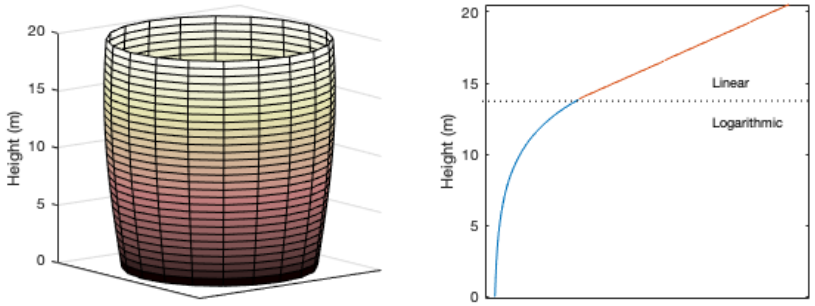


Figure 3.6: Draft of the representation of the curved vessel (left); and, behavior of the height profile (right).

3.3.4 Noise Enabler

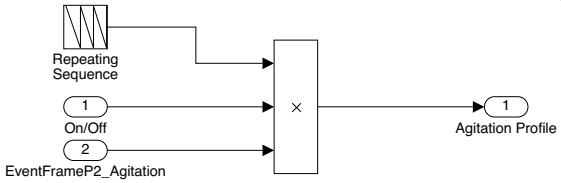
As mentioned in Table 3.1, during specific states, noise is added to the level signal to accomplished a specific level profile. The behavior of these states can not be correctly implemented in Stateflow® due to the limitations of the simulation environment to mimic complex dynamic behaviors. The solution lies on using the Simulink® library browser to accomplish the desired behaviors.

The implementation of the Noise Enabler is the last step of the overall implementation. In practice, the real implementation is initiated with Eq. (3.1), and followed by the discrete environment implementation in Stateflow®. Nevertheless, to simplify the reading, the section is now presented since it appears in a continuous environment. The following subsections introduce the implementation of each profile for the states being affected.

Agitation Profile

Turning the motor on for mixing to start leads to recurrent oscillations in the level sensor. For the implementation, it is assumed the volume fluctuates between -5% and +5%. The *Repeating Sequence Block*, as shown in Table 3.7, provides the model with the oscillation behavior between the above-mentioned values of volume.

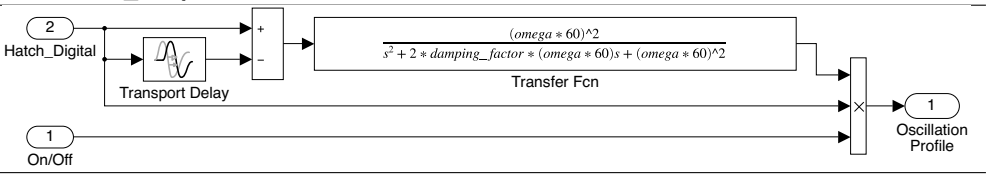
Table 3.7: Variables, parameters, inputs and outputs to implement the Agitation profile.

Variables	$Repeating\ Sequence$ mimics the desired behavior $Agitation\ Profile$ is the profile obtained after the implementation of the repeating behavior
Parameters	$EventFrameP2\_Agitation$ is a binary EventFrame variable linked to the Agitation phase On/Off enables the behavior for this noise profile. Called from the MATLAB workspace
Input	$Repeating\ Sequence, On/Off, EventFrameP2\_Agitation$
Output	$Agitation\ Profile$
Representation	

Oscillation Profile for Adding Solids

Similar to the implementation of the heating and cooling agents, the implementation in Simulink® for the $Add\_Solid\_Material\_Through\_Hatch$ phase, lies on the combination of two blocks: *Transport Delay* and *Transfer Fcn*. With the current parameters on *Transfer Fcn* ($omega$ and $dampingfactor$), the behavior due to splashing and such leads to an uneven level with diminishing spikes.

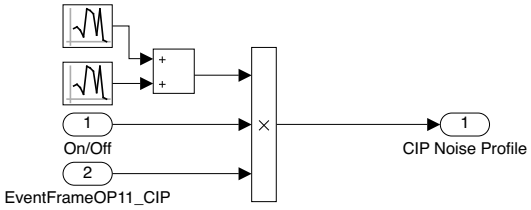
Table 3.8: Variables, parameters, inputs and outputs to implement the Add Solids profile.

Variables	$Oscillation\_Profile$ is the profile obtained after the implementation of the oscillation
Parameters	$Hatch\_Digital$ is a binary variable related with the opening/closing of the Hatch On/Off enables the behavior for this noise profile. Called from the MATLAB workspace $omega = 4\ min$ (natural oscillation period) $damping\_factor = 0.4$ (dimensionless variable evaluating system stability. Stable for >0) $Transport\ Delay = 0.05\ h$
Input	$On/Off, Hatch\_Digital$
Output	$Oscillation\_Profile$
Representation	

Noise Profile for the CIP

Cleaning procedures are often masked with heavy sensor noise due to the filling and draining of water which induces agitation and oscillation behaviors. For a heavy signal output behavior, two *Random Number* blocks are used with different variations, mean values, and sample times. By definition, a sample time refers to the rate at which a discrete system samples its input. For example, increasing a sample time leads to a lower frequency of a signal during the simulation [36]. Table 3.9 provides a simplified description of the parameters, variables, inputs, and outputs necessary to implement to achieve the desired noisy profile.

Table 3.9: Variables, parameters, inputs and outputs to implement the CIP Profile with noise.

Variables	2 $Random\ Number$ blocks to mimic the heavy sensor noise $CIP\ Noise\ Profile$ is the profile obtained after the implementation of the noisy behavior
Parameters	$EventFrameOP11\_CIP$ is a binary EventFrame variable including Event ID 11.1 to 11.5 On/Off enables the behavior for this noise profile. Called from the MATLAB workspace First Block: $variance = 2$, $mean = 2$, $sample\ time = 0.01\ h$ Second Block: $variance = 3$, $mean = 3$, $sample\ time = 0.03\ h$
Input	$Random\ Number$, On/Off, $EventFrameOP11\_CIP$
Output	$CIP\ Noise\ Profile$
Representation	

Noise Profile for Post Reaction

Often these operations have small variations in volume either due to volume expansion/compression or to changes in the vessel temperature (using utilities, for example, that increase the temperature of the vessel) [11]. The noise-base signal covers this physical behavior by having small spikes of 1% on top of the volume measurement. The noise implementation results in equal variances and mean values from batch to batch, but with spikes in different points in time—Table 3.10. As a result, not all batches will look the same: small changes are to be detected.

Noise Profile for the Inactivity Period

Before a new batch production, small spikes on the level sensor of +10% of volume might appear. Since inactivity is defined by not having a batch production or a cleaning procedure, the *Compare to Constant* block chooses the specific times where both *CleaningID* and *BatchID* inactive. By adding the *Uniform Random Number* block to the signal, the desired behavior is obtained—consult Table 3.11.

Table 3.10: Variables, parameters, inputs and outputs to implement the Post Reaction profile with noise.

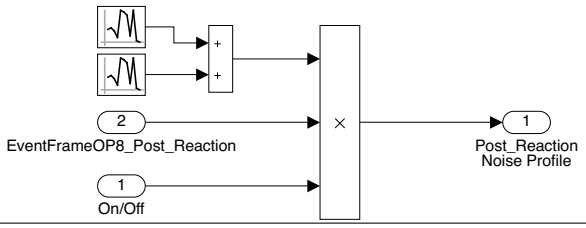
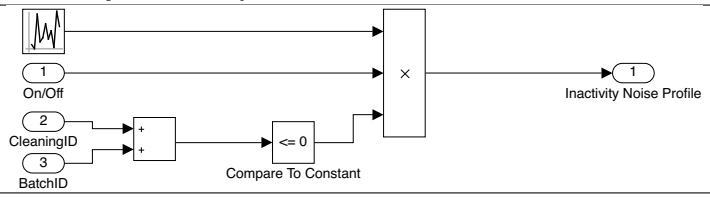
Variables	2 $Random\ Number$ blocks to mimic the heavy sensor noise $Post\_Reaction\ Noise\ Profile$ is the profile obtained after the implementation of the noisy behavior
Parameters	$EventFrameOP8\_Post\_Reaction$ is a binary EventFrame variable linked to Post_Reaction On/Off enables the behavior for this noise profile. Called from the MATLAB workspace First Block: $variance = 0.1$, $sample\ time = 0.02\ h$ Second Block: $variance = 0.05$, $sample\ time = 0.01\ h$
Input	$Random\ Number$, On/Off, $EventFrameOP8\_Post\_Reaction$
Output	$Post\_Reaction\ Noise\ Profile$
Representation	

Table 3.11: Variables, parameters, inputs and outputs to implement the Inactivity period with noise.

Variables	$Uniform\ Random\ Number$ mimics the desired behavior $Inactivity\ Noise\ Profile$ is the profile obtained after the implementation of the noisy behavior
Parameters	$BatchID$/ $CleaningID$ are integer variables for the number of batches/cleaning procedures On/Off enables the behavior for this noise profile. Called from the MATLAB workspace For the Uniform Random Number Block: $maximum = 0.1$, $seed = 20$
Input	$Uniform\ Random\ Number$, On/Off, $BatchID$, $CleaningID$
Output	$Inactivity\ Noise\ Profile$
Representation	

3.3.5 Logical Operations

The operations and phases of the reference chemical process are specified in this environment, as well as the transitions between states (either volumetric or time-constrained). Figure 3.7 represents three operations within the StateChart, where the two phases of Add_Educt3 are represented. This operation is given by a superstate (outer chart) and two sub-states for both phases (inner blocks). From Figure 3.7, one state has to be properly explained: *Initialize*. The state has to be implemented on the simulation to ensure the simulation runs without problems between each new batch and cleaning procedure—verifies the tank is completely empty before producing a new batch. Even meaningless for the process itself, it is necessary to ensure the simulation runs without problems between each new batch and cleaning procedure.

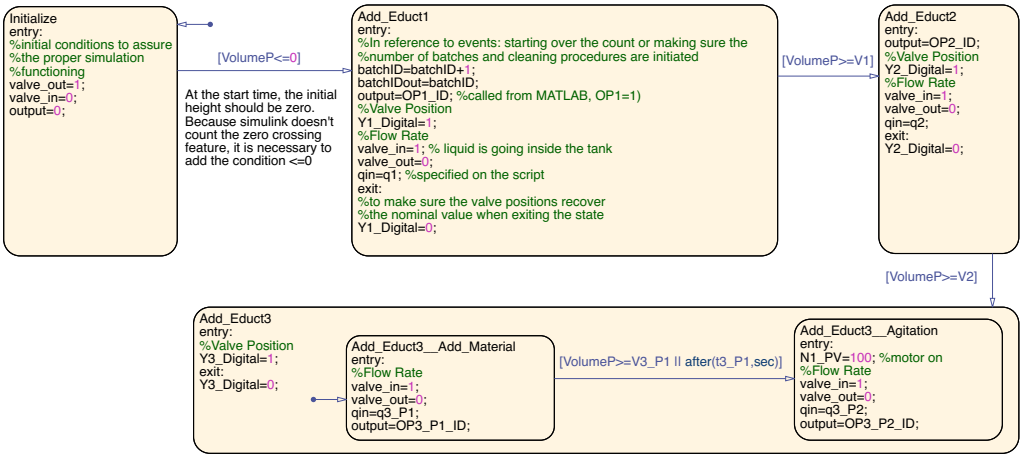


Figure 3.7: StateChart representation of the first three operations of a batch for the Benchmark Model 1.0.

3.4 Disturbances Mapping

Disturbances can be either classified by their occurrence or environment implementation (Figure 3.8). For systematic disturbances are to be considered: **step:** mirror time effects such as a recipe change; **seasonality:** oscillation emulating time effects such as season of the year; **stair:** for example, consecutive steps (changing supplier multiple times); and, a **drift:** such as a linear term overlay.

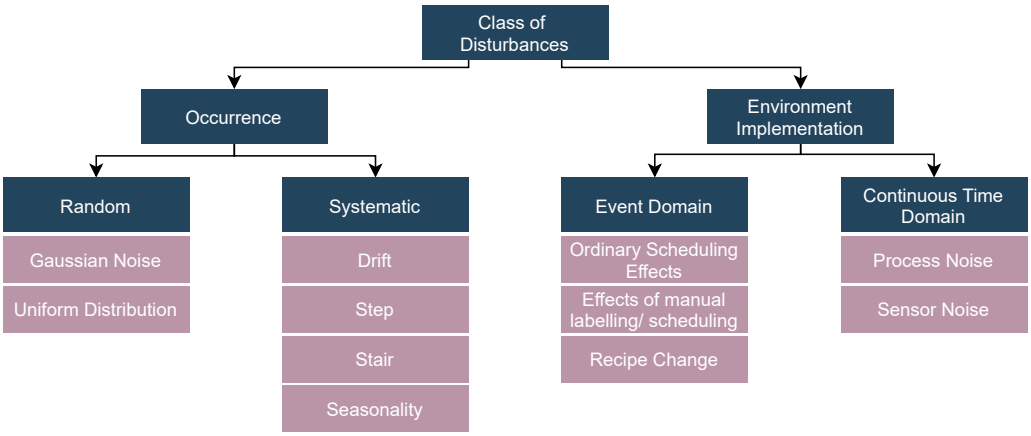


Figure 3.8: Class of disturbances divided by occurrence and environment implementation.

Several disturbances scenarios can be considered when discussing the topic of disruptions and interferences in a real site. Table 3.12 lists key disturbances scenarios studied in the thesis based on the information already provided in Section 2.1.2. Disturbances are classified by type, cause, and origin of the fault.

Table 3.12: Disturbances mapping classified by class and cause.

Class	ID	Cause of Noise/Fault	Behavior Description	Real Masked Effect/Cause
Ordinary Scheduling Effects	1	Delay of single-phase end	Phase end is delayed, no change in actuation occurs during the phase	In the beginning: can mask the effect of an operator taking too long to push a button. In the end: the reaction took too long to mixing, p.e
	2	Delay of multiple phases ends/starts		
	3	A single phase occurs in irregular intervals	For instance, if the yield was found to be not sufficient after quality sample, an additional reaction step is provided †	Unknown specifications of input material
	4	A series of phases occur irregularly	For instance, the cleaning procedure does not always occur at the end of a batch	Valid procedure to happen in a site
	5	Tank is filled to different final fill levels	Common in a batch campaign: after a specific number of batches, the volume decreases between 30–70% of the batch size. Also, small variability of 2% to 5% can be added to final filling stages	Unavailability of resources Consumer demand changing
Effects of manual scheduling or manual labelling	6	Start and/or end points of label(s) are shifted	The end of phase 1 happens, but the logged time end of phase 1 is at the beginning of phase 2. The label of which starts is delayed	Operator taking note of a wrong number or correcting a mishandle from the last shift
	7	Phase wrongly has the name of another phase	It appears has phase 1 lasts for the time of the 2 phases. It appears a wrong label but the sensor measurements remain the same	Wrong operator intervention
	8	Two consecutive phases have the name of the first phase		
	9	A valve is opened and closed several times, and nothing happens before material transfer starts	Instead of being opened and closed once, it happens more times. Valve starts open	Pump was not working, or an upstream valve was still closed

Table 3.12 continued from previous page

Class	ID	Cause of Noise/Fault	Behavior Description	Real Masked Effect/Cause
	10	A valve is opened and closed several times, and nothing happens after material transfer ends	Instead of being opened and closed once, it happens more times. Valve starts closed	Operator is opening and closing valves several times probably making sure it is certainly closed
Recipe change	11	Phase name remains the same, but actuation changes	A different valve is opened at several instances without that behavior having meaning to the reference process	New raw material on site Varying consumer demand
Process Noise	12	A pump slowly supplies less throughput	The flow rate slowly decreases in time. To reach the same level of liquid in the tank, a certain task takes more time to complete	Aging/Clogging pump
	13	The motor provides less agitation	Rotation number decreases at random times, and reset after each batch	Operator intervention due to: Liquid characteristics (high viscosity) Prevent foam formation
	14	Utilities Flows are masked with noise	Incremental changes in the flow and, as a result, increasing spikes are observed	Low Utilities Supply Flow exiting another unit
Sensor Noise	15	Loss of signal: a series of data points are not written	The sensor output has no value until a random time	Sensor Failure
	16	Value outside of sensor range	If the level measurement deviates from normal values (0 to 100), an error message appears	Loss of effectiveness
	17	A sensor suffers from a gradual drift for a period of time ‡	Drift lasting until the end of the simulation, resulting in volume shift, as much as, the slope chosen	Faulty Calibration Equipment Malfunction
	18	A sensor suffers from a gradual drift and is suddenly recalibrated	Similar to ID15 but the sensor has a sudden recalibration after a specific number of batches	

Table 3.12 continued from previous page

Class	ID	Cause of Noise/Fault	Behavior Description	Real Masked Effect/Cause
	19	A sensor suddenly has an offset (recalibration or fault) † ‡	At random points in time, with small durations, the sensor has offsets. The actuation remains unaltered	
	20	A wide variety of sensor noise	Several amplitudes, frequencies, and, statistical distributions: gaussian (ID20), uniform (ID20A) and with oscillations (ID20B)	
	21	White-Band Noise on Sensors	Added to the signal introducing small spikes in measurements	

† Not possible with the current implementation: chemical reactions and dynamics being omitted. It is, nevertheless, a challenge to tackle in future work.

‡ There are a lot of possible ways to implement this in a disturbance generator. It can be implemented by choosing a slope and start time of the drift.

† ‡ Possible to implement two different cases: if this sensor is merely an output; or if it is used for feedback to the control system (then a phase duration will most likely change.

3.5 Modelling and Implementation of Disturbances

Disturbances are modeled stochastic, with the user controlling likelihood and severity to test ML algorithms in different scenarios. The parameterization of disturbance scenarios from the main script enables massive simulation studies. For that reason, the control center is set in the MATLAB environment, being enabled systematically with benchmark cycles of disturbances (detailed discussion in Section 4.4).

Tables 3.13 and 3.14 allow for a demonstration of the implementation of a disturbance scenario for each Stateflow® and Simulink®. For presentation sake, each disturbance has an ID which is the same as in Table 3.12.

Disturbances in Stateflow® are set with a probability, $probabilityThreshold$, which provides a percentage of batches that will be affected for that disturbance in particular. Only one example will be explained since all disturbances in this environment follow the same implementation.

For Disturbance ID1, if the control enabler is on for Operation 6, $Product\_Transfer$, ($Disturbance\_EnablerOP6\_ID1 = 1$), a random probability will be set within the StateChart ($chance\_disturbanceOP6\_ID1$). Whenever this value is less than the $probabilityThreshold\_DisturbanceOP6\_ID1$, the disturbance is enabled and the transition will be from $Product\_Transfer$ to $Product\_Transfer\_Disturbance\ Waiting$. On the Waiting block, the behavior is supposed to remain the same. Even though $Product\_Transfer$ does not have a discrete variable triggered by the operation, if that is the case (e.g., on Add_Educt1), the valve position will continue the behavior set before. The delay in time will also be completely random.

Table 3.13: Representation of disturbances on the discrete environment.

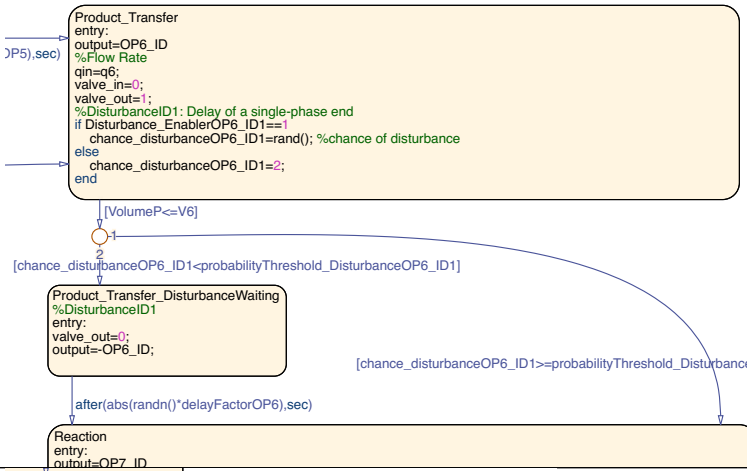
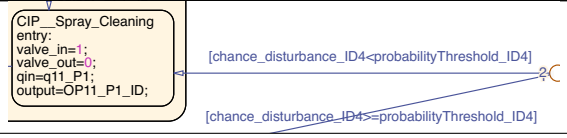
ID	Disturbance representation in Stateflow	Events affected
1 & 2		After each nominal operation/phase
3 & 4		Only for the cleaning procedure

Table 3.13 continued from previous page

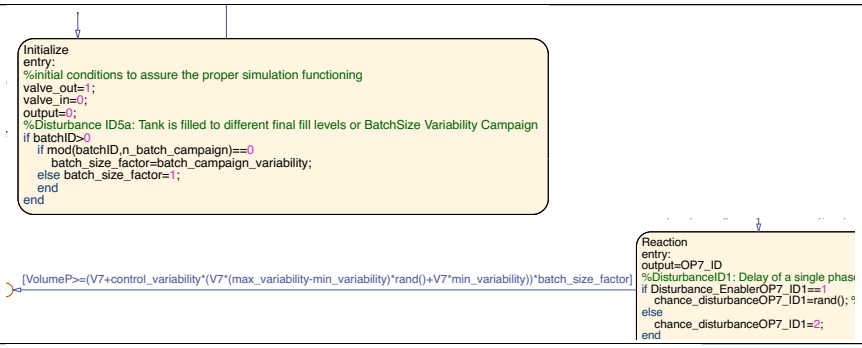
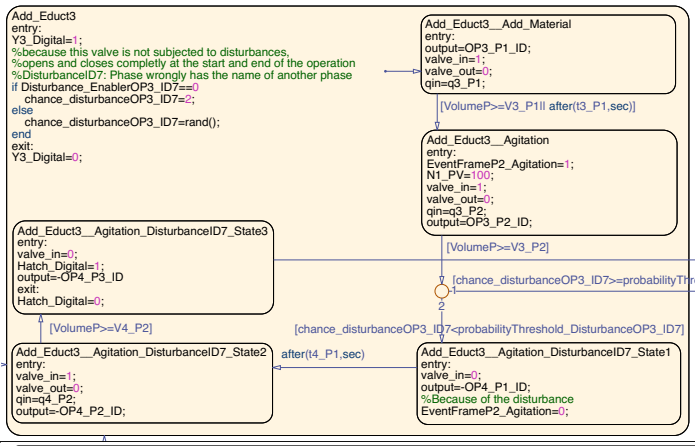
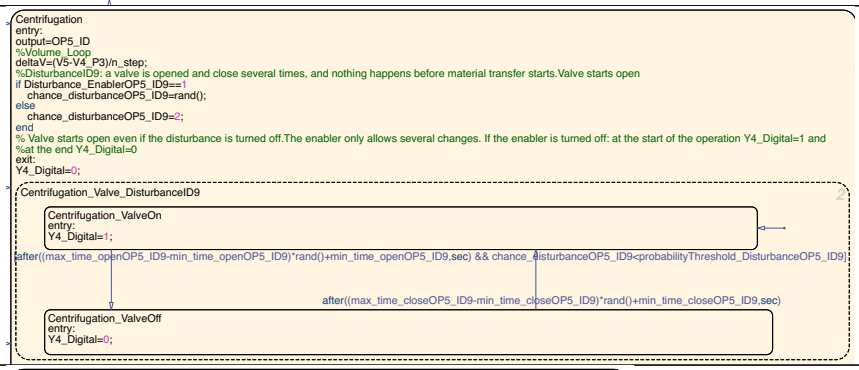
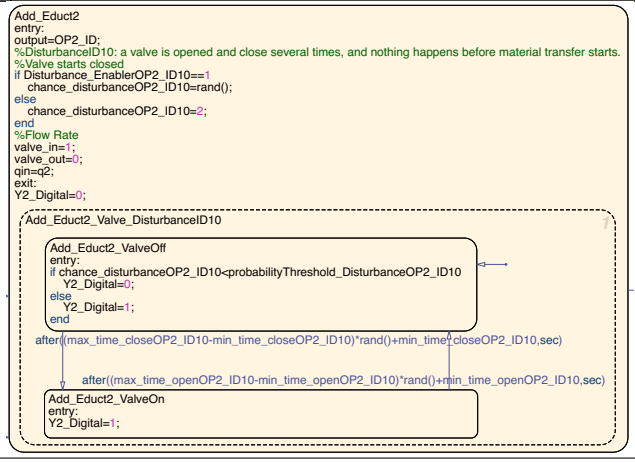
ID	Disturbance representation in Stateflow®	Events affected
5		For all nominal operation/phase with volumetric constrains
7 & 8		Only OP4 is masked as OP3
9		Only enabled for OP1 and OP5. As such, Y1 and Y4 are affected
10		Only enabled for OP2. As such, Y2 is affected

Table 3.13 continued from previous page

ID	Disturbance representation in Stateflow®	Events affected

13	Cooling
	entry:
	output=OP9_ID
	if Disturbance_EnablerOP9_ID13==1
	chance_disturbanceOP9_ID13=rand(); %chance of disturbance
	else
	chance_disturbanceOP9_ID13=2;
	end
	exit:
	N1_PV=0;
	Cooling_ID13
	Cooling_Off
	entry:
	N1_PV=N1_PV
	Cooling_On
	entry:
	N1_PV=N1_PV*(1-rand());
	[chance_disturbanceOP9_ID13<probabilityThreshold_DisturbanceOP9_ID13]
		Only enabled
		for OP5 and OP9

As presented, ID1 to ID10 and, ID13 are implemented in the discrete environment, which leaves ID11, ID12, and ID14 to ID21 with the implementation in Simulink®. Depending on the disturbance class, the subsystems in Simulink® differ, being possible to paired them in groups when similar.

Disturbances with **sensor noise** (ID15 to ID21) are implemented similarly: enabled with the number of batches and set with a minimum and maximum duration of occurrence in batch production - Figure 3.9. Indirectly, the implementation of these continuous environment disturbances ends up working as a probability by choosing minimum and maximum points of occurrence.

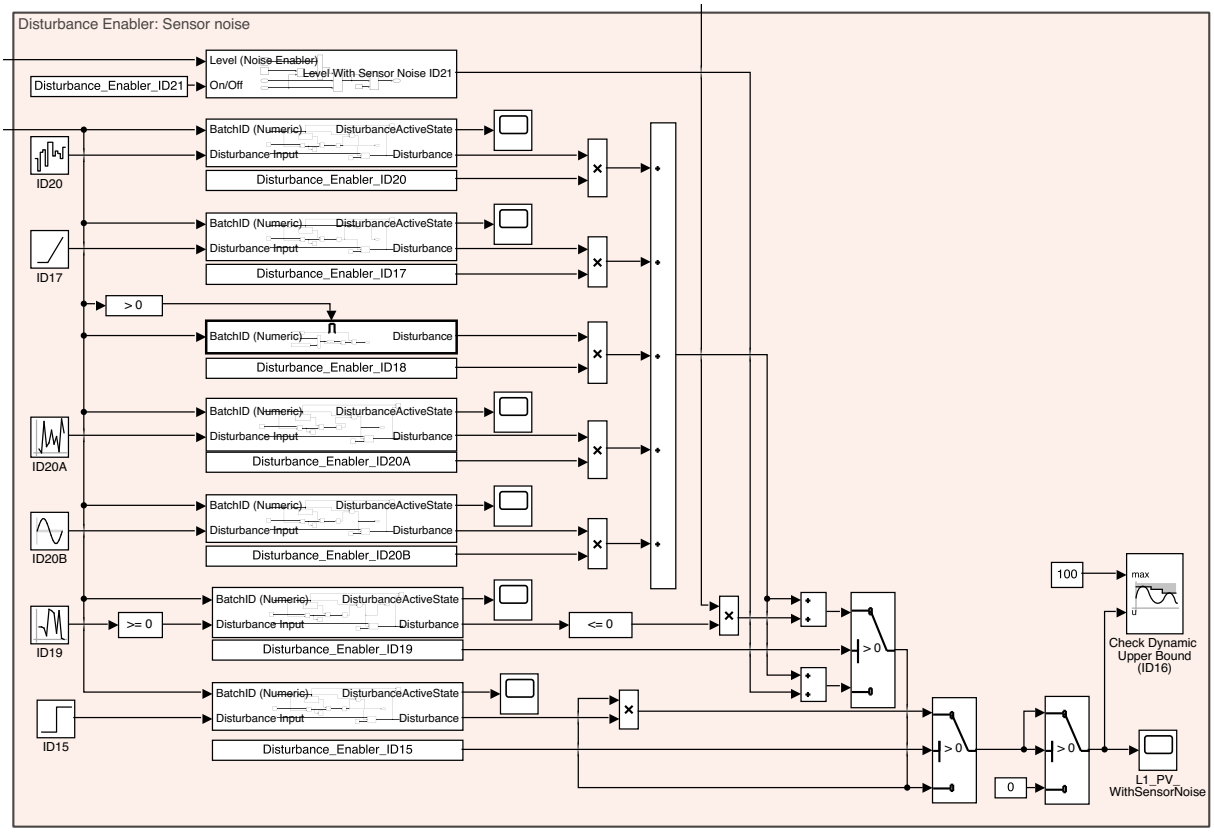


Figure 3.9: Graphical Framework of Simulink® with the disturbance enabler for sensor noise scenarios (ID15 to ID21).

Choosing different *Source Blocks* from the *Library Browser* allows for mimicking different disturbances scenarios—graphical representation on the far left of Figure 3.9, as $Disturbance\ Input$. Table 3.14 provides detailed information about the source block and the dependency of each disturbance scenario with a batch production.

Table 3.14: Representation of sensor noise disturbances scenarios.

ID		**15**	**17**	**18**	**19 & 21**	**20**	**20A**	**20B**
Dependency of batch production	Dependent			X	X	X	X	X
	Independent	X	X					
Disturbance Input block		*Step*	*Ramp*	-	*Random Number*	*Band-Limited White Noise*	*Uniform Random Number*	*Sine Wave*

Each disturbance has a correspondent enabled subsystem which is presented on Figures 3.10 and 3.11. The only difference between the two implementations is if the disturbing profile will occur only in batches (Figure 3.10) or, also, in cleaning procedures (Figure 3.11) - the signal connected to the *Switch Block* (input port 2, for >0) is different.

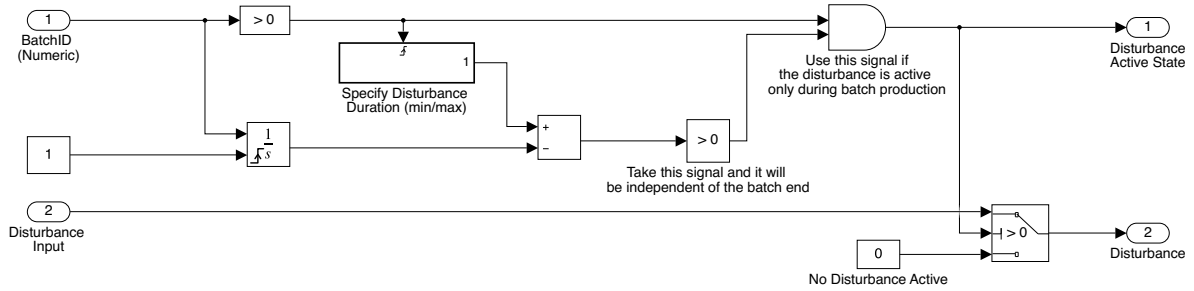


Figure 3.10: Enabled Subsystem for disturbances implemented on the continuous environment with dependency on the occurrence of batches.

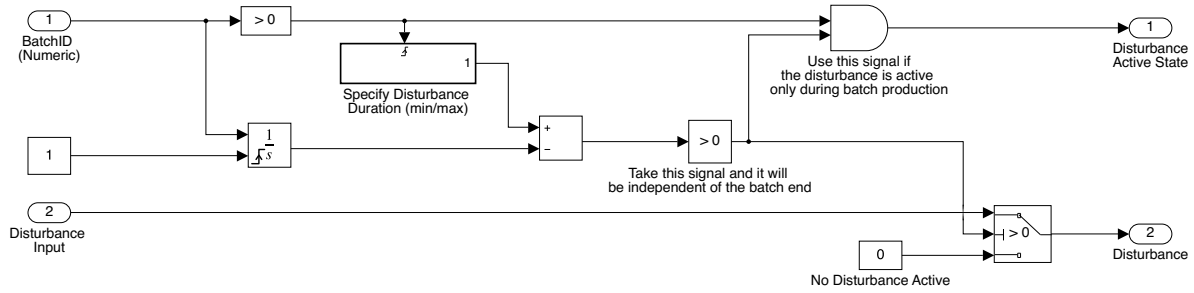


Figure 3.11: Enabled Subsystem for disturbances implemented on the continuous environment without dependency on the occurrence of batches.

ID16 has a particular implementation: instead of provoking changes on the actuation, it prevents the tank from flooding as a result of other disturbances being enabled. From a simulation point of view, it is not unreasonable to surpass the mark of 100% of the level. Nevertheless, from a chemical point of view, such is unfeasible for an industrial site. The block to use is *Check Dynamic Upper Bound* (representation

on the far right of Figure 3.9).

For ID18, the implementation is also enabled with a production of a batch, and its set for recalibration after a specific number of batches: $n\_batches\_ForReset\_ID18$. After said number, the drift starts over (subsystem representation provided in Figure 3.12).

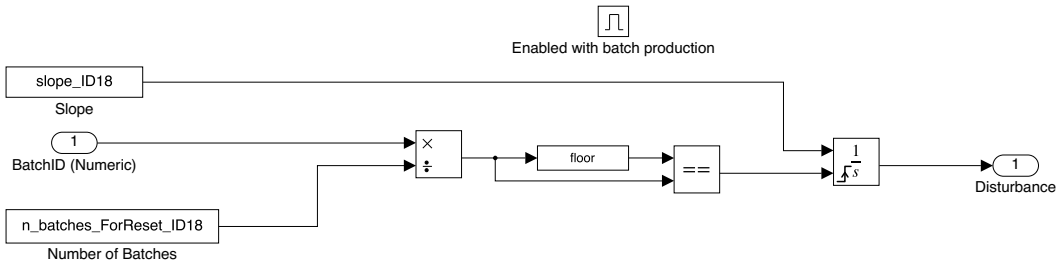


Figure 3.12: Enabled Subsystem for disturbance ID18 with dependency on the occurrence of batches.

The last sensor noise disturbance (ID21) adds a white noise profile on top of the level signal by using two *Random Number* blocks with different sample times and variances. The slow signal with increasing variance (top left of Figure 3.13) is added to a second signal. The latter, extremely fast with a low variance signal, is multiplied by the level sensor signal.

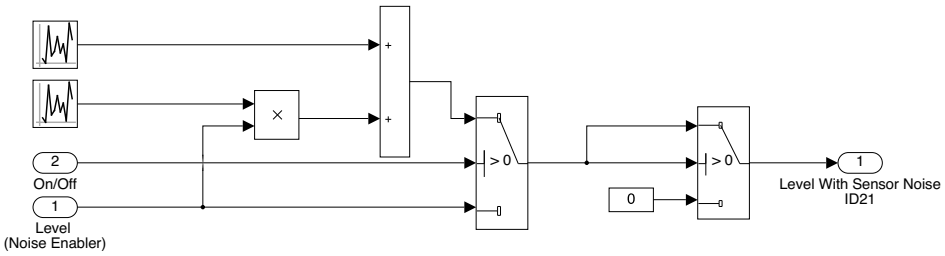


Figure 3.13: Enabled Subsystem for disturbance ID21 with dependency on the occurrence of batches.

Once discussed disturbances masking sensor noise, ID11, ID12 and, ID14 are left to present. The last two disturbances scenarios belong to the **Process Noise** class and, ID11 to **Recipe Change** scenarios.

Discussing first ID11, to mimic different valves opening and closing without being triggered by specific operations/phases, the implementation must be done in Simulink®. The combination of a *Random Number* and *Compare to Constant* blocks provide the desired behavior (Figure 3.14):

- *Compare to Constant Block:* output of True (1) or False (0) in each point in time as the input signal is less or equal to 0. The binary output mimics a valve being open and close.

- *Random Number Block:* Choosing different sample times, allows to randomly change the start and end-points of the valve's behavior. The higher the sample time, the less occurrence.

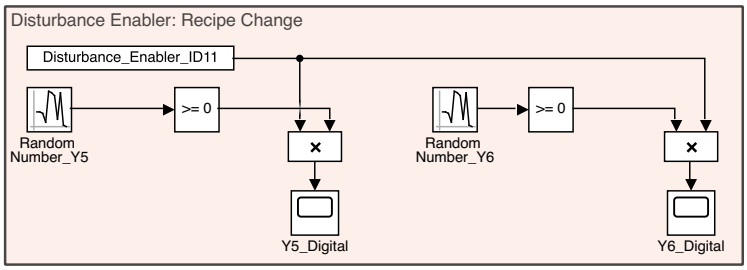


Figure 3.14: Graphical Framework of Simulink® with the disturbance enabler for recipe change disturbances scenarios (ID11).

Disturbance scenario ID12, by mimic an aging pump, provides flow variability for the simulation: flow decreases during the simulation period. Instead of having all operations/phases being affected for said variability, the sample time can be increased resulting in less occurrences.

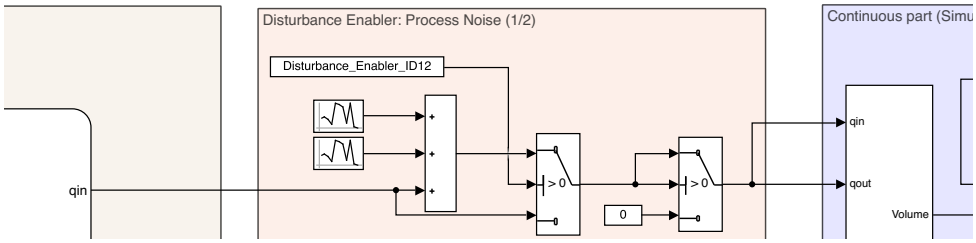


Figure 3.15: Graphical Framework of Simulink® with the disturbance enabler for process noise disturbances scenarios (ID12).

At last, ID14 is presented: to the utilities flows, sample noise is added to the signal. The main purpose of this scenario is to increase variability to the variables triggered when Reaction and Cooling occur. Even if changes are made affecting the vessel level, if the flows remain unchanged, these states are easily identified by the algorithm.

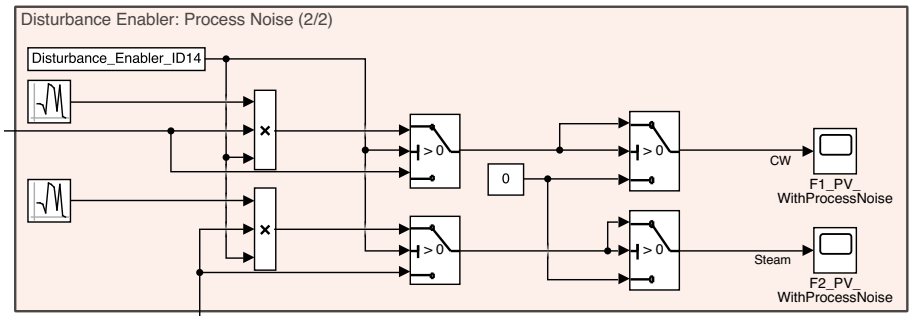


Figure 3.16: Graphical Framework of Simulink® with the disturbance enabler for process noise disturbances scenarios (ID14).

4

Results of Benchmark Model Simulation Disturbance-Free and With Disturbances

Contents

With the reference model and disturbances defined in the previous chapter, its performance can now be assessed. The following sections will cover: visualization of the reference model per batch with and without disturbances, and the impact of particular disturbances in the training of a ML algorithm.

4.1 Nominal Reference Model

As mentioned in the last chapter, the production of a batch and the following cleaning procedure last for, approximately, 21 hours. Most disturbances are implemented within batches, being relevant to share the level measurement of a disturbance-free batch—Figure 4.1. Nonetheless, the profile of a cleaning procedure is also introduced on the batch phase identification plot.

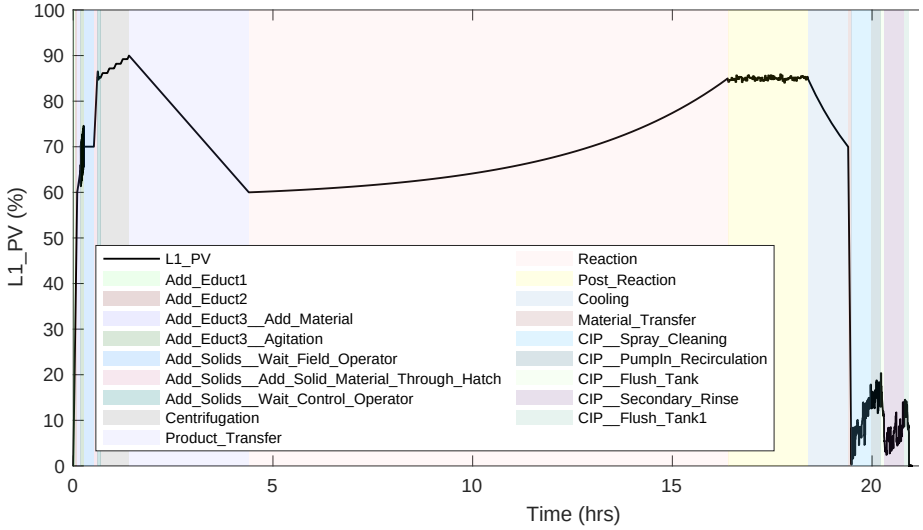


Figure 4.1: Nominal behavior of a batch and a cleaning procedure for the reference model with batch phase identification.

The reference output of the label to-be-learn by the ML algorithm provides the batch phase identification—$EventFrameID\_Reference$—, with each state lasting for the nominal duration set on process description.

Discrete variables, such as valve positions, are triggered when a state (operation or phase) is active, e.g., $Y1\_Digital$ only opens for a specific operation (Add_Educt1). When plotting sensors measurements and valve positions, there is a clear dependency between the two. For the particular example described, when the level reaches 30%, $Y1\_Digital$ will close and until 60% of fill, $Y2\_Digital$ is the one to be open (Add_Educt2 is now active). Because of that, Figure 4.3 gives an understanding of how the simulation works and, for the relevant output variables, which phases and operations can be identified.

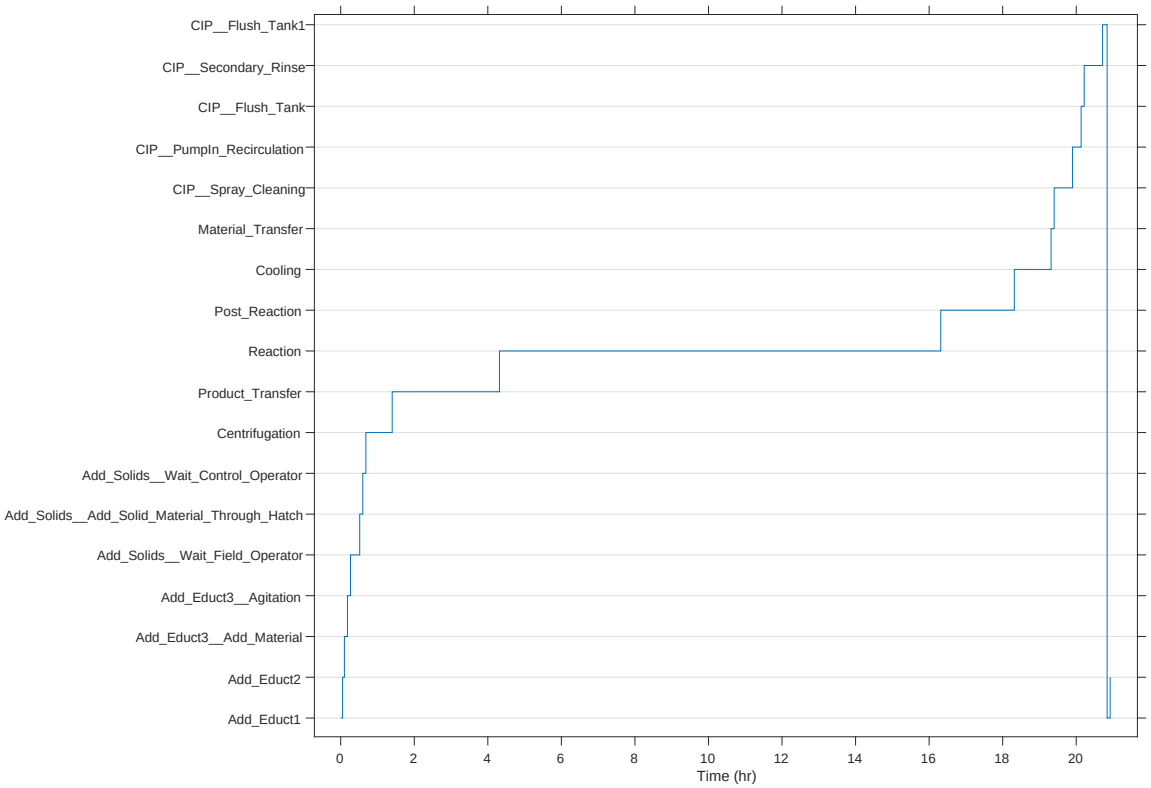


Figure 4.2: $EventFrameID\_Reference$ with the nominal operations and phases.

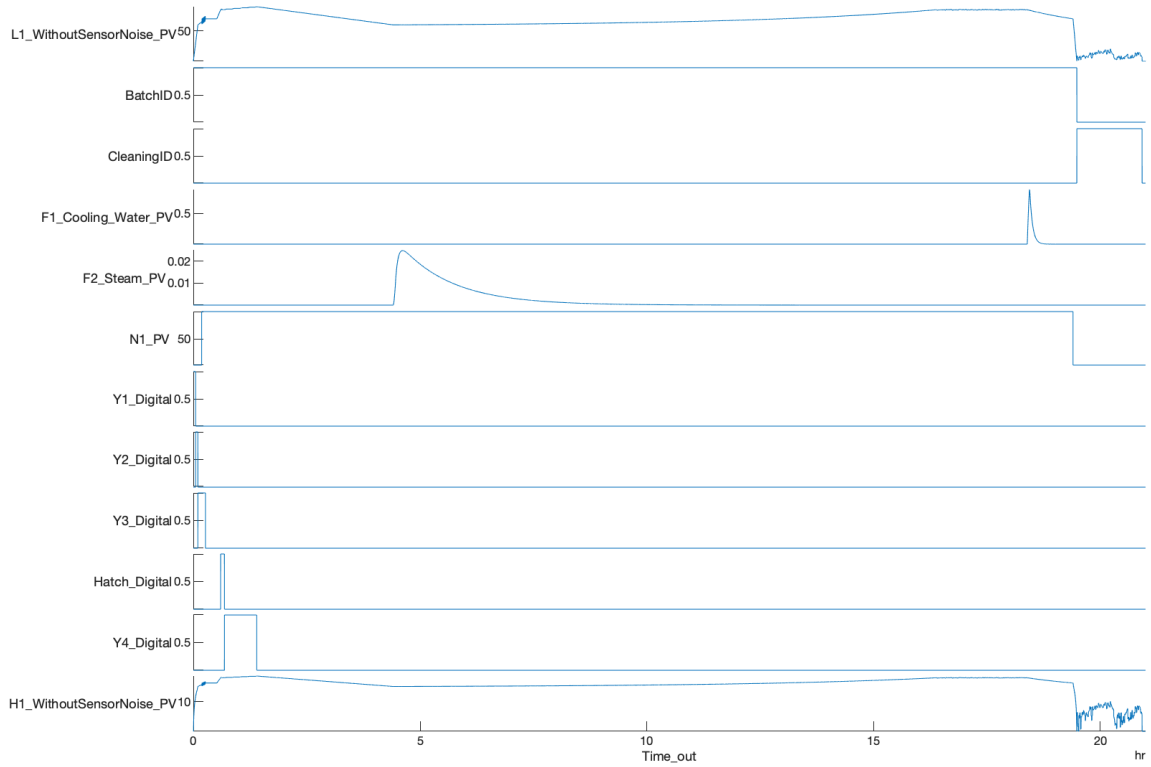


Figure 4.3: Nominal behavior for the relevant output variables of the reference model.

4.2 Non-Nominal Behavior

After establishing the nominal behavior for the reference model, by enabling the control center of each disturbance, it comes the possibility of studying non-nominal behaviors. For the feasibility study of the ML algorithm, the simulation should be running for an entire batch-operational year. Whereas, for a proper demonstration of each disturbance, running for 4 batches is enough and provides a sufficient understanding of the impact of a specific disturbance. Nonetheless, at the end of this section, the simulation will run for at least 40 batches to show the model is performing well enough to simulate numerous batches.

4.2.1 Disturbances ID1 & ID2: Delay of multiple phases end/starts

These disturbances can be grouped since the implementation is identical: only the number of operations affected by a delay is different. Enabling ID1/ID2 results in *no level change, remaining constant for the extent of the delay* - such behavior is implemented at the end of the operation.

Although it is possible to enable these disturbances for all operations, only disturbances in Add_Educt1 (OP1) and Add_Educt2 (OP2) are discussed here for presentation's sake. The $probabilityTreshold$ of the disturbance is set at 0.6, which means 60% of the batches will be affected by a delay. If 4 batches are being produced, it means that at least 2 of them should have a delay for both operations.

Since the disturbances are implemented inside the StateChart, to mimic a delay behavior, transitions between states differ when the disturbance is enabled. Exemplifying for the operations suffering said disturbance: if a delay occurs after OP1 (and, of course, before OP2), between the states of these operations, a delay state block will appear (also recognize as $Add\_Educt1\_DisturbanceWaiting$ of Figure 4.4). Such block is to appear after every single operation and passed-by every time, as a probability, if the user wishes to.

Not every batch will have the expected nominal behavior - when overlapping the level sensor measurement of all batches, such conclusion is logical. This can be seen on Figure 4.5 (as well as in Figure 4.4) which, accordingly, shows a delay occurring after OP1 and OP2 for batches 2 and 4. It is interesting to notice the time of the delay differs—because the implementation sets the delay as a random number, after each batch, the extent of the delay is different.

These disturbances are in practice extending the duration of the two operations. Because $Y1\_Digital$ and $Y2\_Digital$ are set to open/close when those two operations begin/end, the valve positions behavior will last until the delay is over. From Figure 4.6, both $Y1\_Digital$ and $Y2\_Digital$ last for longer in batches 2 and 4 when comparing with the remaining batches.

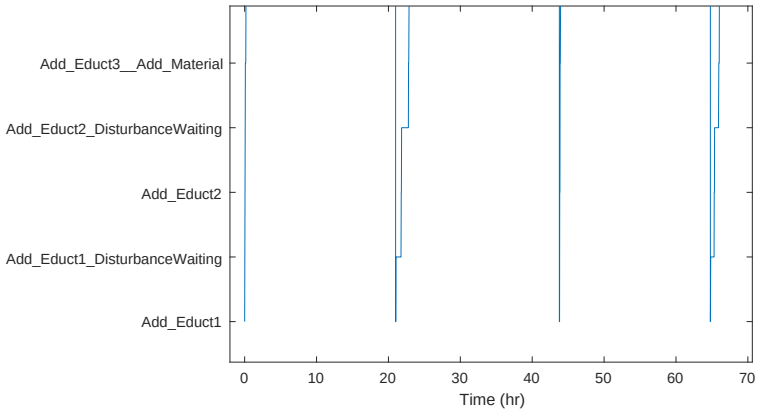


Figure 4.4: $EventFrameLabel\_Current$ output variable for the disturbance ID1/ID2. States $Add\_Educt1\_DisturbanceWaiting$ and $Add\_Educt2\_DisturbanceWaiting$ are activated with the disturbance enabler and transitions occur on batches 2 and 3.

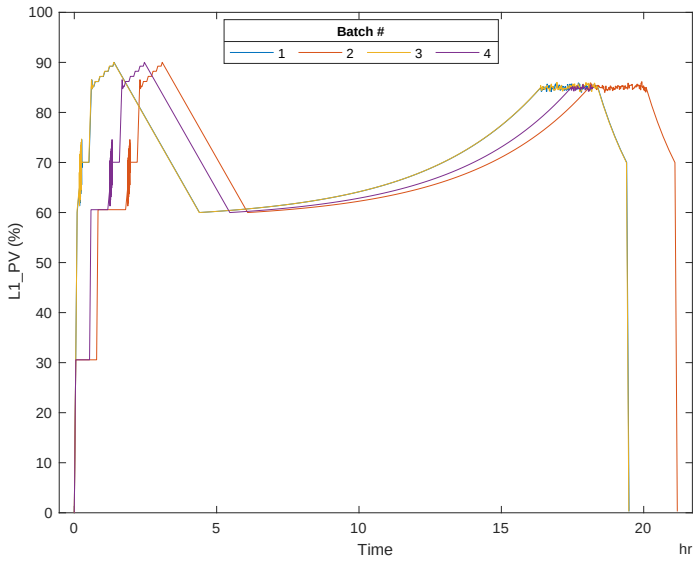


Figure 4.5: Overlapping of all batches which shows a non-nominal behavior for batches 2 and 4 due to enabling ID1/ID2.

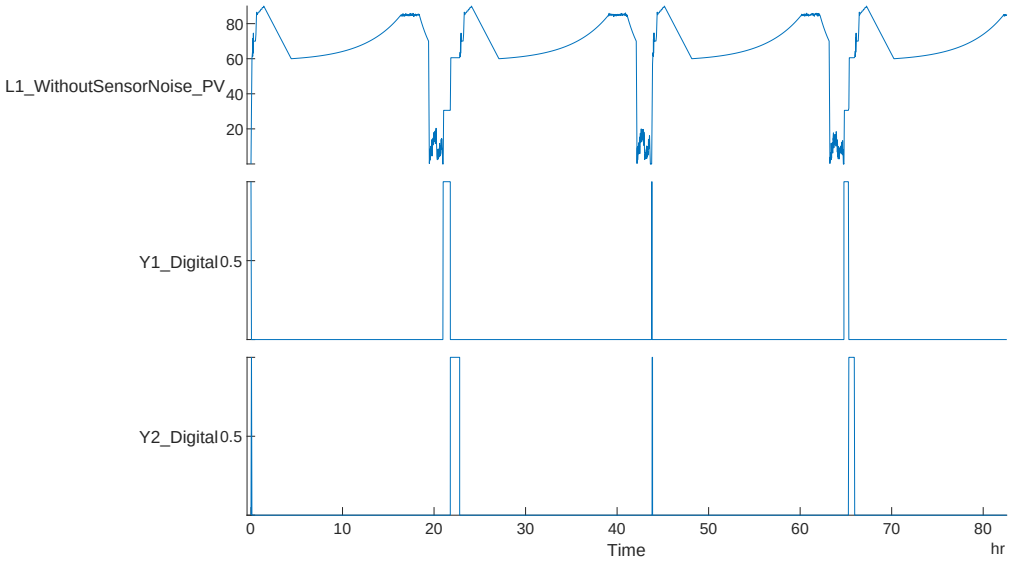


Figure 4.6: Stackedplot comparison of the relevant output variables to interpret the impact of Disturbance ID1/ID2.

4.2.2 Disturbance ID3 & ID4: A series of phases occur irregularly

ID3/ID4 introduces a more realistic setting in process engineering: instead of a cleaning procedure to occur after the end of a batch, it only follows that path 20% of the time. For the remaining period, the transition goes from $Material\_Transfer$ to the $Initialize$ state, following the nominal operations for a new batch production. Seeing the level profile for a production of 4 batches, only on one batch does the CIP profile appears—right after batch 2 is produced (Figure 4.7).

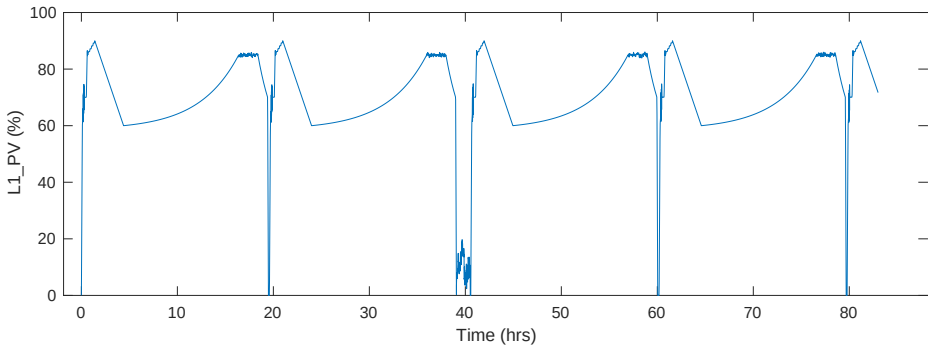


Figure 4.7: Vessel level representation to interpret the impact of Disturbance ID4.

4.2.3 Disturbance ID5: Tank is filled to different final fill levels

Scenario 1. Batch Campaign Variability

In spite of being described as a disturbance, batch size variability in a campaign is common in an industrial site, particularly to respond to a change in the customer order [14]. Let a batch campaign

consist of 10 batches: the disturbance is introduced after the 10^{th} batch, wherein the volume decreases 70%. This is why, for this disturbance scenario, the simulation runs for, at least, the production of 11 batches. Because there is a sudden decrease in volume, the duration of the disturbed batch also decreases.

Once again, overlapping the level measurements of the batches provides an overview of the influence of this disturbance on the simulation - Figure 4.8. In this case, only batch 11 presents a non-nominal behavior.

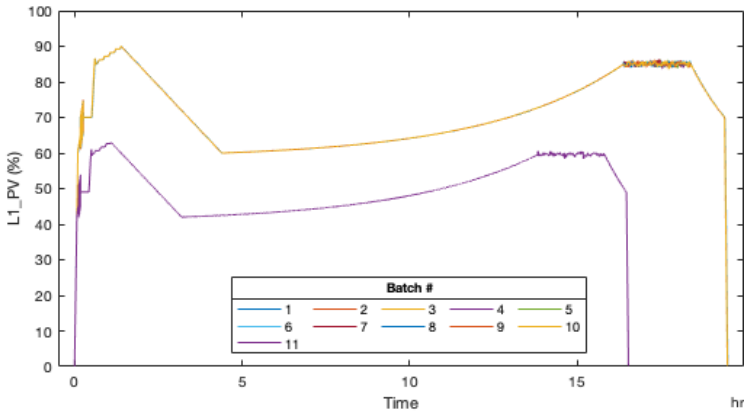


Figure 4.8: Overlapping of all batches which shows a non-nominal behavior for batch 11 due to enabling ID5.

Since start and end points of operations/phases also change on the 11^{th} batch, the valve positions will also be distinct. Accordingly, Figure 4.9 presents the open/close actions of these valves for said batch—orange line in contrast to the other blue ones.

Scenario 2. Volume End-Phase Variability

Instead of having a decrease in volume for a specific batch, variability at end phases/operations is added by making the volume vary between +2% and +5% of the nominal value. The user can manipulate the level of variance on the script. Nonetheless, by plotting this behavior in Figure 4.10, operations as $Centrifugation$ and $Post\_Reaction$ already have clear changes from batch to batch. For example, for batch production 3 and 4 different level end points can be recognized—in practice, this type of variability gives parallel level representations.

4.2.4 Disturbances ID6 & ID7 & ID8: Phase wrongly has the name of another phase, even if consecutive phases; no change in actuation

Because the implementation is equivalent, these disturbances can be paired up. Normally due to wrong operator interventions, phases/operations can be wrongfully labeled—actuation has no alteration as

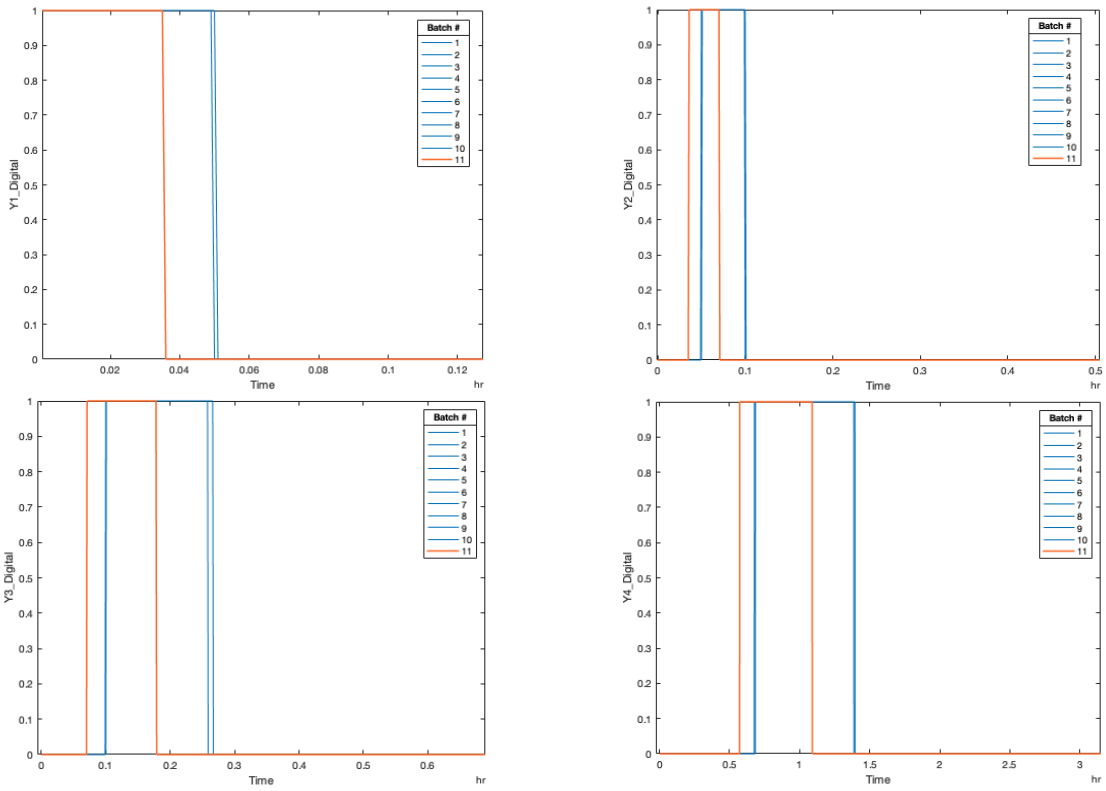


Figure 4.9: Valve Positions differ when disturbance ID5 is enabled. Batch disturbed represented in orange.

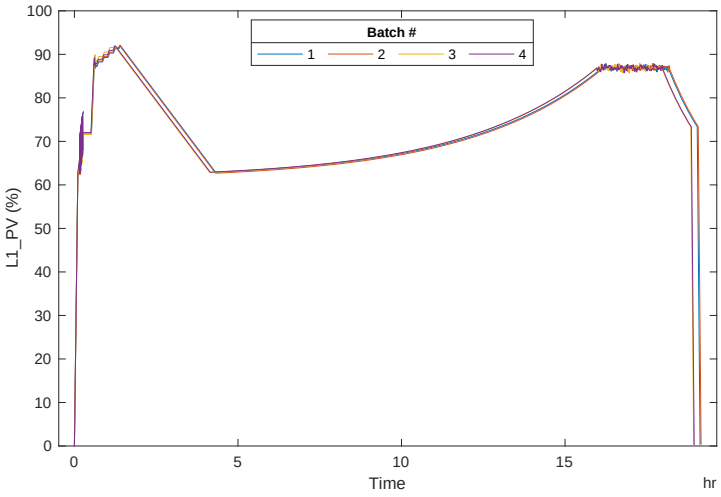


Figure 4.10: Overlapping of batches which shows a the nominal behavior even when enabling ID5 for volume phase end variability.

well as sensor values but labels are mistaken. The previous statement can be showed in Figure 4.11 by overlapping all batches and compare with the nominal behavior of Figure 4.1 (page 49).

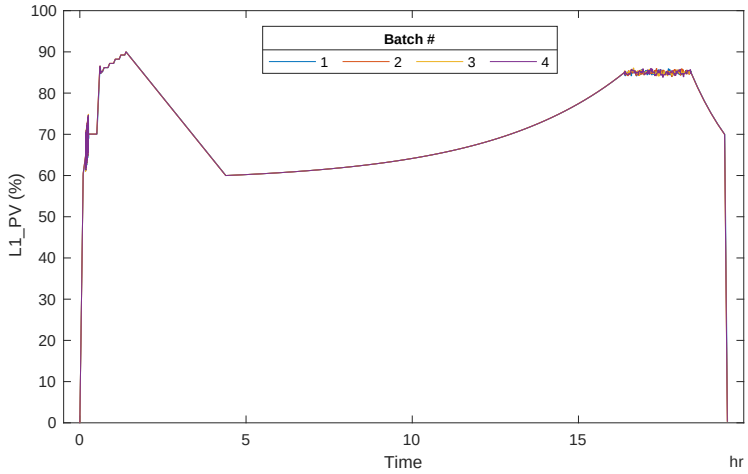


Figure 4.11: Overlapping of batches which shows a nominal behavior even when enabling ID7.

The implementation on StateChart is such that the entire operation Add_Solids will be masked as the previous state, $Add\_Educt3\_Agitation$. For a better understanding of the label disturbance in Stateflow®, the reader should consult A.1.1.That being the case, the output label will not go through the states of OP4 when the disturbance occurs—as verified by Figure 4.12 from 21.3 hrs to 21.6 hrs.

In post-processing, the label for these disturbances will be similar to the last nominal phase/operation; so, all states of $Add\_Educt3\_\_Agitation\_DisturbanceID7$ will be presented solely as $Add\_Educt3\_Agitation$. Subsequently, there will be no record for the ML algorithm of new states being introduced: it will be up to the learning algorithm to recognize label disturbances.

4.2.5 Disturbance ID9: A valve is opened and close several times, and nothing happens before material transfer starts

This disturbance models pump malfunctions resulting in valve positions disrupted by being opened and close several times (binary values change frequently). As such, no change on the continuous variables is detected, only on the valve positions (discrete variables).

Since the implementation in Stateflow® is made for the operations $Add_Educt1$and $Centrifugation$, only changes will occur on $Y1\_Digital$ and $Y4\_Digital$. Thereby, the representation of Figure 4.13 provides a sufficient preview of the above-mention disturbance scenario.

Fixing a probability of 60% for ID9, at least half of the batches being produced are affected. On the grounds that operation 4 takes longer to complete, the behavior of $Y4\_Digital$ is easily envisioned; $Y1\_Digital$ not so much. That being the case, Figure 4.13 is enhanced for a better preview of the

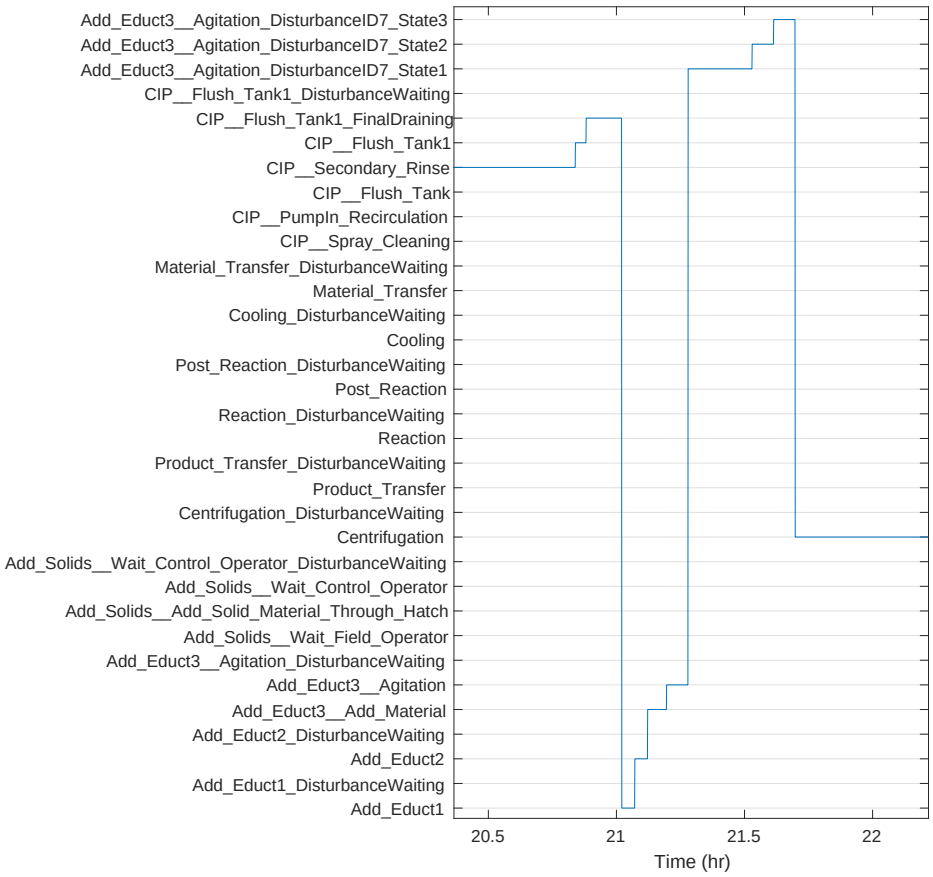


Figure 4.12: $EventFrameLabel\_Current$ output variable for the disturbance ID6/ID7/ID8. States of operation Add_Solids have no value as a result of the disturbance being enabled.

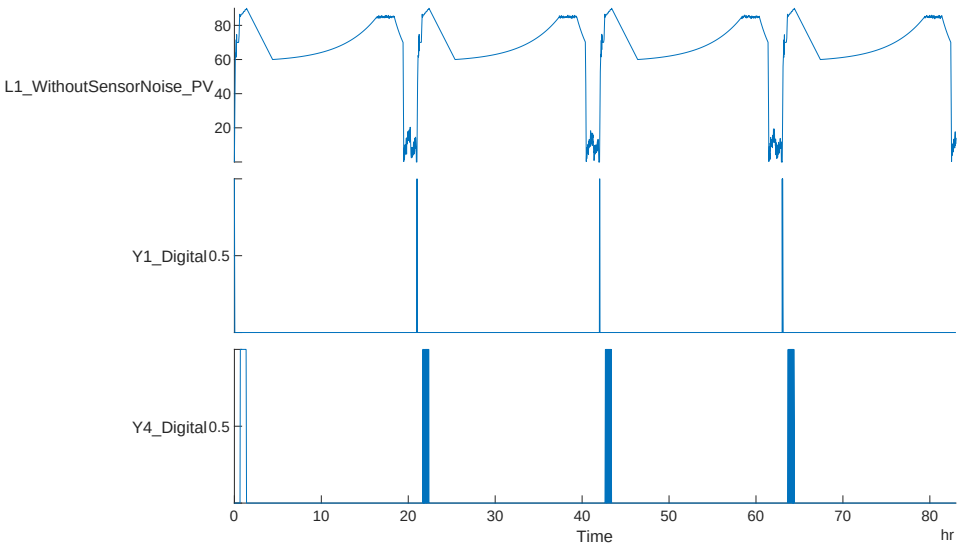


Figure 4.13: Stackedplot comparison of the relevant output variables to interpret the impact of Disturbance ID9.

behavior described—Figure 4.14.

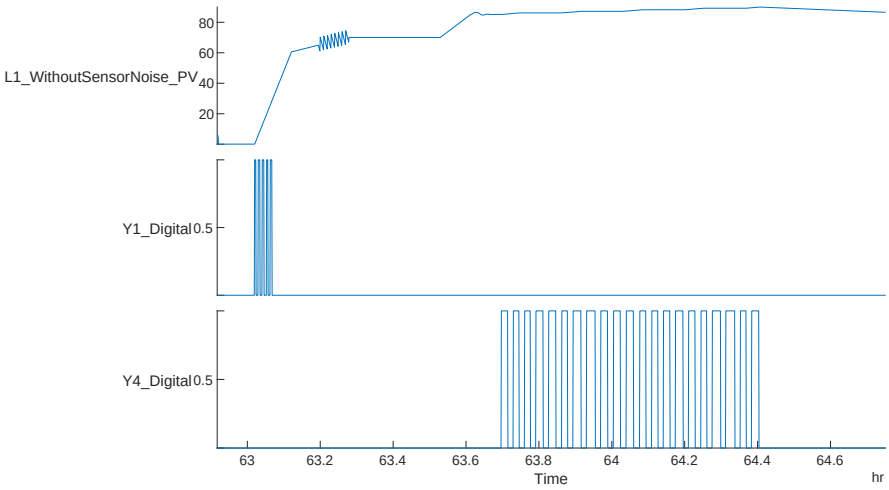


Figure 4.14: Stackedplot comparison of the relevant output variables to interpret the impact of Disturbance ID9. Close-up to see the behavior of both valves.

The amount of time a valve is open/close is random and given between minimum and maximum values following a uniform distribution. For this particular scenario, open/close behaviors were short but said behavior can change by increasing minimum and maximum durations.

4.2.6 Disturbance ID10: A valve is opened and close several times, and nothing happens after material transfer starts

Similar to the disturbance ID9 with one difference: instead of the valve starting open when the operation is triggered, the valve starts closed.

As on the previous disturbance, the amount of times a valve is open and close for the duration of the disturbed operation is given randomly. Setting for a quick shift of values does not allow to properly demonstrate this behavior fixing the x-axis as the duration of the simulation on the grounds of the operation lasting for 3 minutes. Therefore, a close-up is in order which allows visualizing the aforementioned behavior.

Since $Y2\_Digital$ when the operation is active, for a nominal behavior this valve will open/close at the start/end points of Add_Educt2. On account of the valve starting closed, only after a random time (with the operation already began), the valve opens and behaves as expected for this disturbance scenario.

4.2.7 Disturbance ID11: Phase name remains the same, but actuation changes

The addition of a variable that has absolutely no meaning for the unit in the study should add complexity to the model—the binary value for the position of the valves is triggered at random times of the simula-

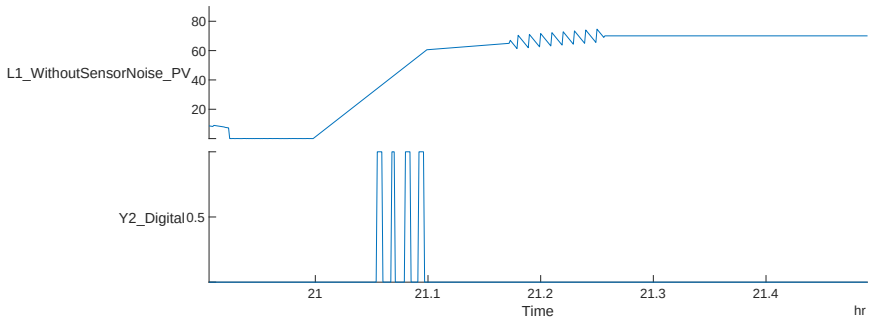


Figure 4.15: Stackedplot comparison of the relevant output variables to interpret the impact of Disturbance ID10.

tion. The implementation, discussed in the previous chapter, allows for the valve to be opened/closed as much as the user allows it—the higher the sample time, the lesser times the valve is opened.

The disturbance is enabled independently of the batches which mean, these meaningless valves ($Y5\_Digital$ and $Y6\_Digital$) can be open during cleaning procedures. By plotting the level sensor in parallel with these valves, it becomes clear $Y5$ and $Y6$ do not follow a uniform distribution being, for said reason, unpredictable—consult Figure 4.16.

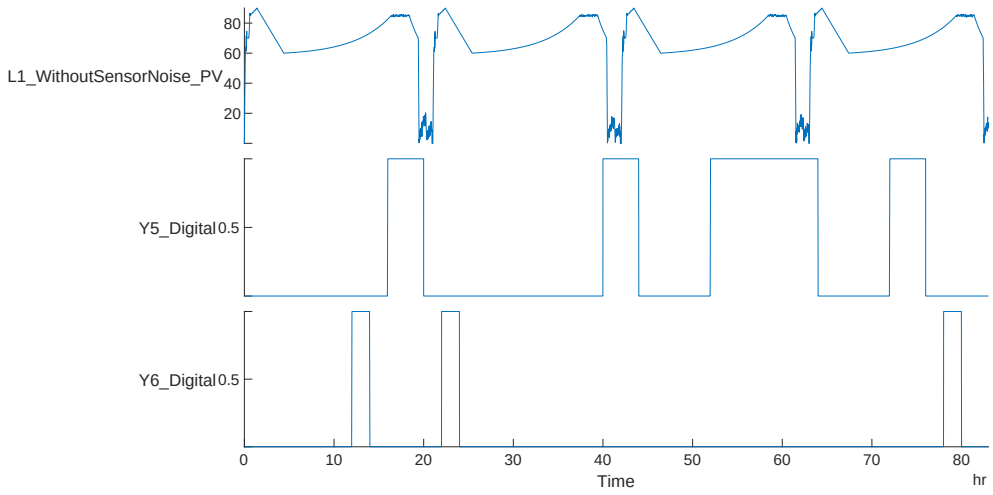


Figure 4.16: Stackedplot comparison of the relevant output variables to interpret the impact of Disturbance ID11.

As a reminder: several valves were implemented on the benchmark process being open as a trigger of particular operations. Executing the simulation with 2 valves opened/closed at the same time is a challenge—especially if one of them, does not have the same duration and is triggered as the other. It is up to the ML algorithm to depict the relevant variables and to understand their meaning [29].

4.2.8 Disturbance ID12: A pump slowly supplies less throughput

As stated by the name of the disturbance, by aging or clogging, a pump supplies less throughput during the process. By implementing such disturbance before the *Integrator Block*, changes are mostly abrupt - comparing this case, Figure 4.17, for example, with ID21 in Figure 4.28 where the disruption is made after the flow integration.

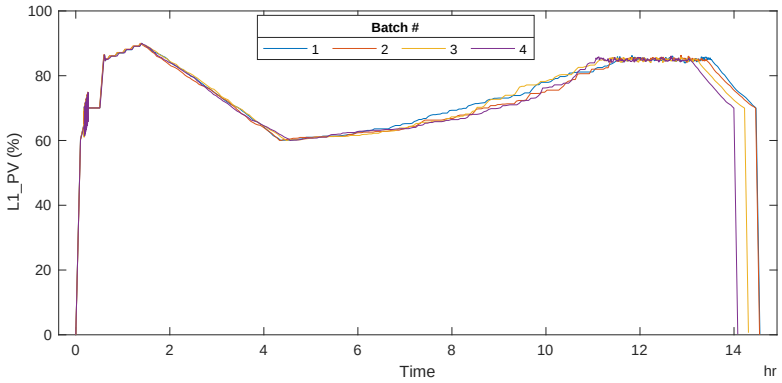


Figure 4.17: Overlapping of batches which shows non-nominal behavior due to enabling ID12.

The higher the deviation of the flow rate from the nominal value, the higher is the disturbance in the sensor measurements (level and height). Visible from Figure 4.17, the exponential profile for $Reaction$ and $Cooling$ were the most affected operations—non-parallel profiles were obtained with white noise.

To avoid having every Batch/CIP disturbed, the sample time is set to be different from 0. In this case, 0.01 h and 0.05 h were chosen, each from a different *Random Number Block* . The slow signal presents a higher variance of 10% while the fast signal, only presents deviations of 5%.

Increasing variability will provide more severe distributions of the sensor measurements and, at some point, paired up with other disturbances such as ID1/ID2 will create very real and likely scenarios as **variability and delays are the most common disruptions in a chemical site** [13].

4.2.9 Disturbance ID13: The motor provides less agitation

The operator during the production of a batch, commonly, adjusts the rotation of the agitation motor to prevent foam formation; or, because the liquid viscosity is too high affecting liquid properties. Decreasing the rotation, diminishes viscosity ensuring the proper working conditions. The motor is switched on at the beginning of $Add\_Educt3\_\_Agitation$ and set at 100 rpm until it is switched off on $Cooling$. The disturbance in the number of rotations is implemented in Stateflow® to occur in two operations: $Centrifugation$and $Cooling$. Because it does not occur at specific times or even with specific reductions, the percentage of decrease is also chosen to be random.

For the plot of Figure 4.18, $probabilityThreshold\_DisturbanceOP5\_ID13$ and $probabilityThreshold\_$

$DisturbanceOP9\_ID13$ are set, respectively, as 0.5 and 0.6. Thus, is to be expected that exactly 2 batches the motor has a decrease in OP5; and, at least in 2 batches, the decreases in the rotation succeeds on OP9. On the same plot, it can be seen the rotation reduction is distinct: on batch 3, for OP5, the reduction is 80% but, for the same operation, on batch 4, the rotor decreases by 20%.

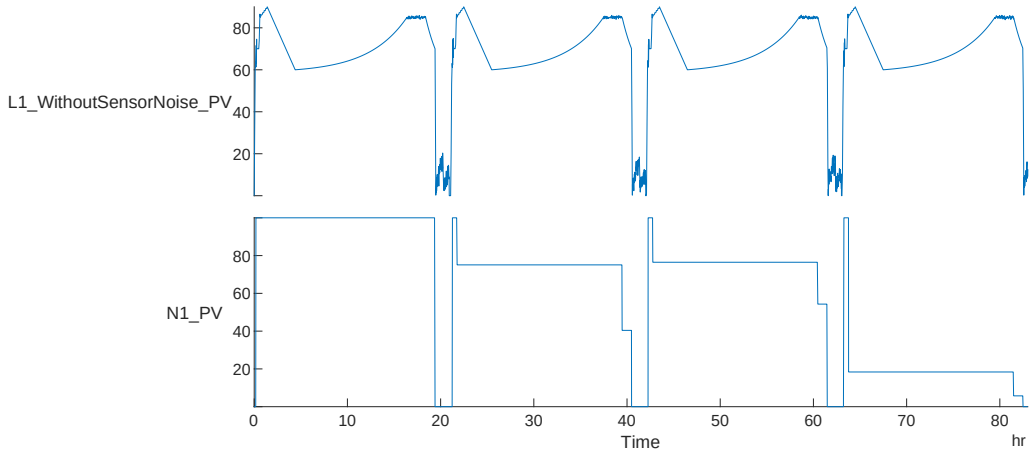


Figure 4.18: Stackedplot comparison of the relevant output variables to interpret the impact of Disturbance ID13.

4.2.10 Disturbance ID14: Utilities flows are masked with noise

Either due to low utility supply, disruptions, or delays in the process affecting utility availability in this unit, the flow sensor of cooling water and steam can present a non-nominal behavior. Recollecting the nominal behavior of an exponential profile for both utilities, the flow sensor is disturbed by a gradual increase of spikes. When the operation ends and the flow reaches 0, small spikes of 1% of flow can be observed.

The primary purpose of this disturbance is, when paired with other disturbances scenarios that affect both $Reaction$ and $Cooling$, to affect the predictability of recognizing said operation. In other words, to increase the difficulty for the ML algorithm since the trigger variables are also affected.

Note that the utilities flows implementation is independent of the tank level and, as a result, no changes in the sensors occur.

4.2.11 Disturbance ID15: Loss of signal

ID15 can be considered a random disturbance since the probability of occurring, in reality, is narrow—a series of data points are not written and, as a result, the sensors present no value during a specific time from the moment the batch production starts.

Representing the level sensor with and without sensor noise on parallel plots should be an adequate

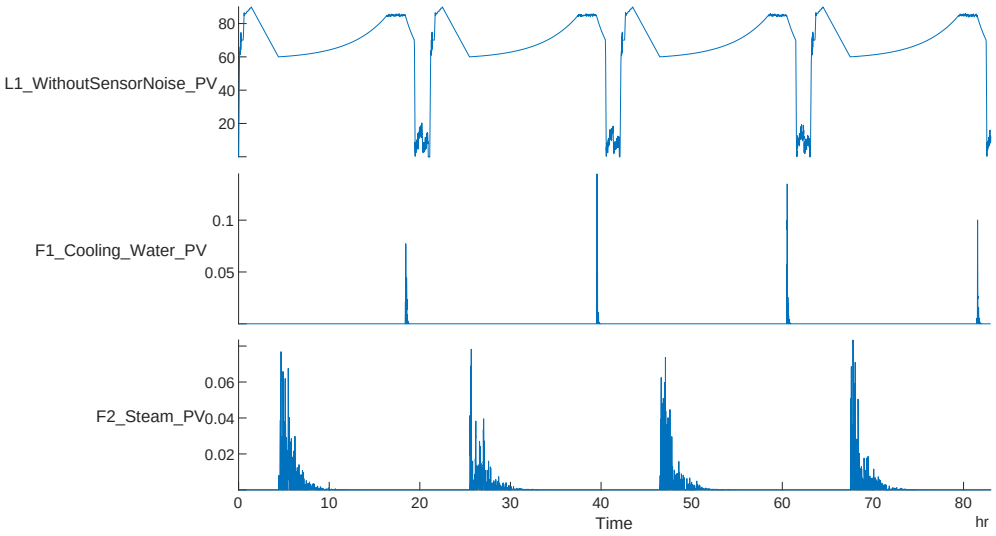


Figure 4.19: Stackedplot comparison of the relevant output variables to interpret the impact of Disturbance ID14.

presentation of these results (Figure 4.20). In this particular case, there is loss of signal in the first batch and a nominal behavior from batch 2 onward.

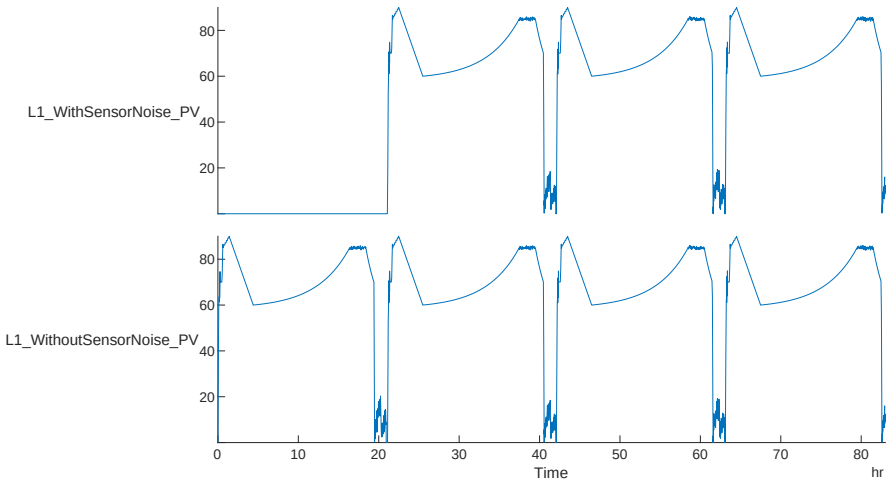


Figure 4.20: Stackedplot comparison of the relevant output variables to interpret the impact of Disturbance ID15.

It should be interesting to see how the ML algorithm reacts to the disturbance scenario: even with no signal during the first batch, the actuation for the remaining variables do not change (e.g., the valves present the normal behavior to expect: opening and closing for the trigger operations).

4.2.12 Disturbance ID17: A sensor suffers from a gradual drift for a period of time

Bearing in mind this particular disturbance, the drift has to be carefully chosen to prevent the vessel overflowing—if the slope is massive, the level sensor can surpass more than 100%. As such, the goal of ID17 is for the level to suffer a drift for the entire time of the simulation without over-flooding of the tank.

Although the level difference never surpasses more than 5% of the nominal value, Figure 4.21 shows small changes with an increase of level after batch 2. When simulating longer, a slope of 0.05 might be too high of a value. Because ID16 gives an error message, the user will be supplied with the information that such parameter should be rethought of.

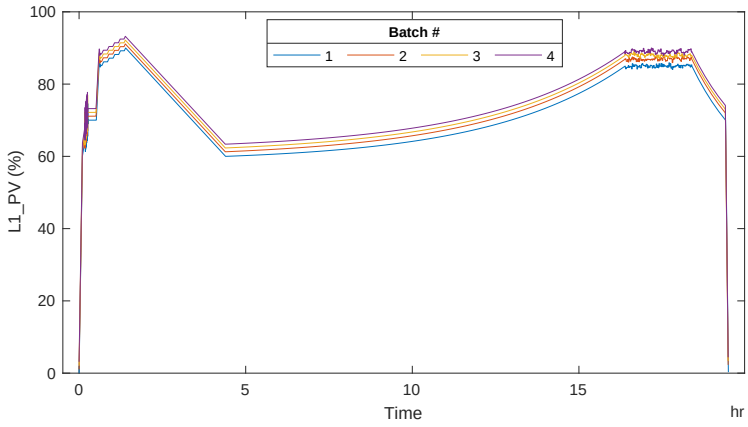


Figure 4.21: Overlapping of batches which shows a non-nominal behavior due to enabling ID17.

The graphic environment of Simulink® provides an understanding of the disturbance profile when batches are being produced. In this case, for ID17, the start/end points of the duration of this disturbance are independent of the number of batches by choice (check Figure 4.22). For the next disturbance, such independence will not be expected.

4.2.13 Disturbance ID18: A sensor suffers from a gradual drift and is suddenly recalibrated

Choosing the extent of the drift as well as the time in which the level behavior gets back to normal, allows mimicking a different disturbance scenario. Since ID18 is extremely similar to ID17, all particularities of the previous implementation should be take in to account with only one difference to depict: at each 5^{th} batch, the sensor is recalibrated.

By running more time, batches will have the same behavior when paired in groups: batches 1 and 5; 2 and 6; 3 and 7; and henceforth—Figure 4.23. That fact is due to the recalibration being set at a number of equally spaced batches with each batch having an increase of level in 20%.

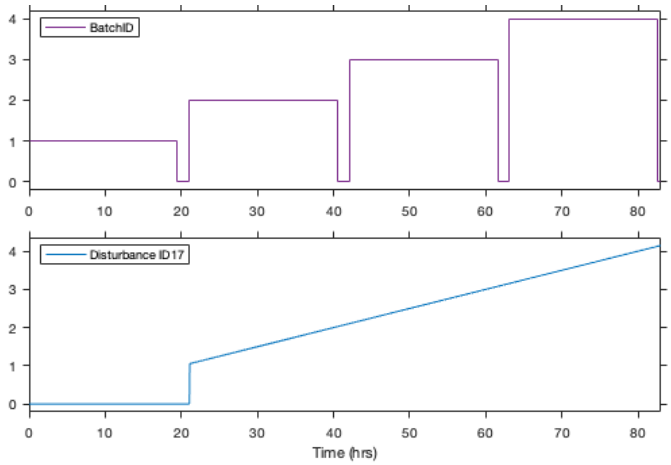


Figure 4.22: Batch and profile of the disturbance represented (ID17).

Having a recalibration at the same number of batches reminds of ID5 (the level decreasing at the end of a batch campaign) which can become predictable—an important point to discuss in the next section of the ML algorithm feasibility study.

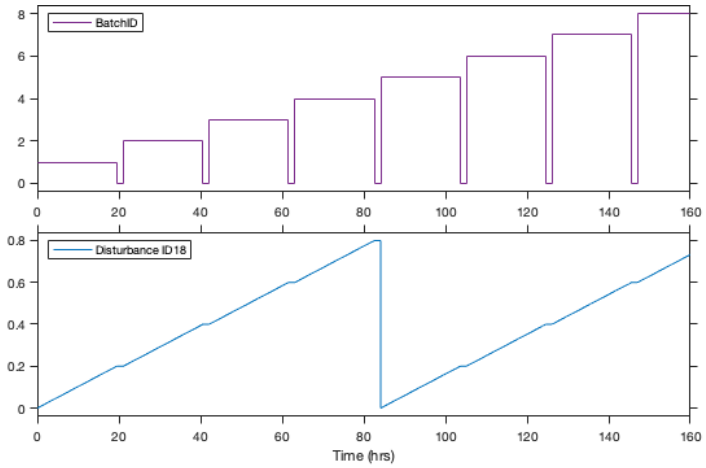


Figure 4.23: Batch and profile of the disturbance represented (ID18).

4.2.14 Disturbance ID19: A sensor suddenly has an offset

Either due to recalibration or fault, the sensor measurements will lack some points in time. Even an expected disturbance in a site, **offsets are not so common and when occurring is only by a few instances**—to be kept in mind when analyzing the results. For that reason, the minimum and maximum duration of the disturbance should be aligned with that fact: set a duration according to the number of batches being produced. Because the simulation runs for the production of 4 batches, the minimum and maximum duration of ID16 are set at 3 and 6 hours. Since the implementation is according to the

occurrence of batches, Figure 4.24 illustrates the random behavior in which ID19 comes about; also, the total amount of time does not exceed 6 hours.

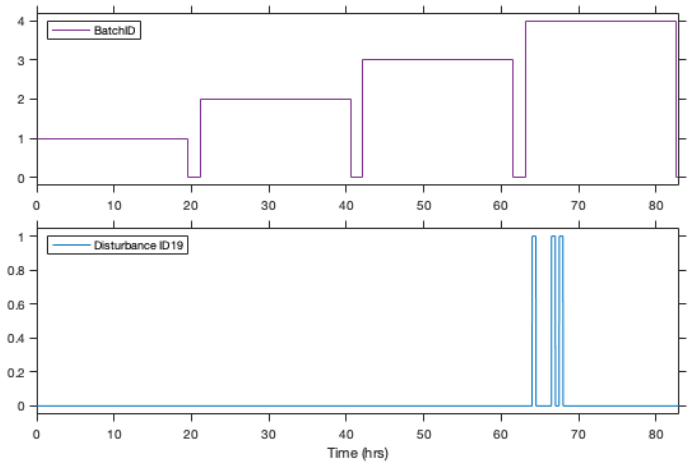


Figure 4.24: Batch and profile of the disturbance represented (ID19). Clear that the disturbance occurs at random batches, per consequence, at random times.

Offsets are singular and easily detectable. For that reason, representing the level sensor with and without sensor noise on parallel plots is an adequate presentation of these results - Figure 4.25.

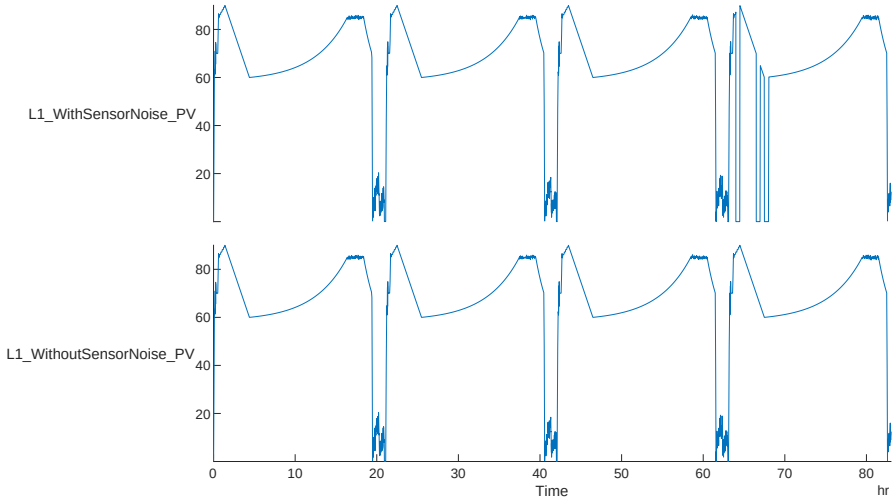


Figure 4.25: Stackedplot comparison of the relevant output variables to interpret the impact of Disturbance ID19.

4.2.15 Disturbance ID20: A wide variety of sensor noise

Equipment failure is one of the most common faults in an industrial site [11] which results in sensors masked with noise. As such, the probability of the disturbance to occur should take that into account.

That being the case, Figure 4.26 presents a parallel representation of the level sensor with ID20 (gaussian), ID20A (uniform), and ID20B (oscillations) being enabled. The profile of each singular disturbance scenario is displayed in Figure 4.27.

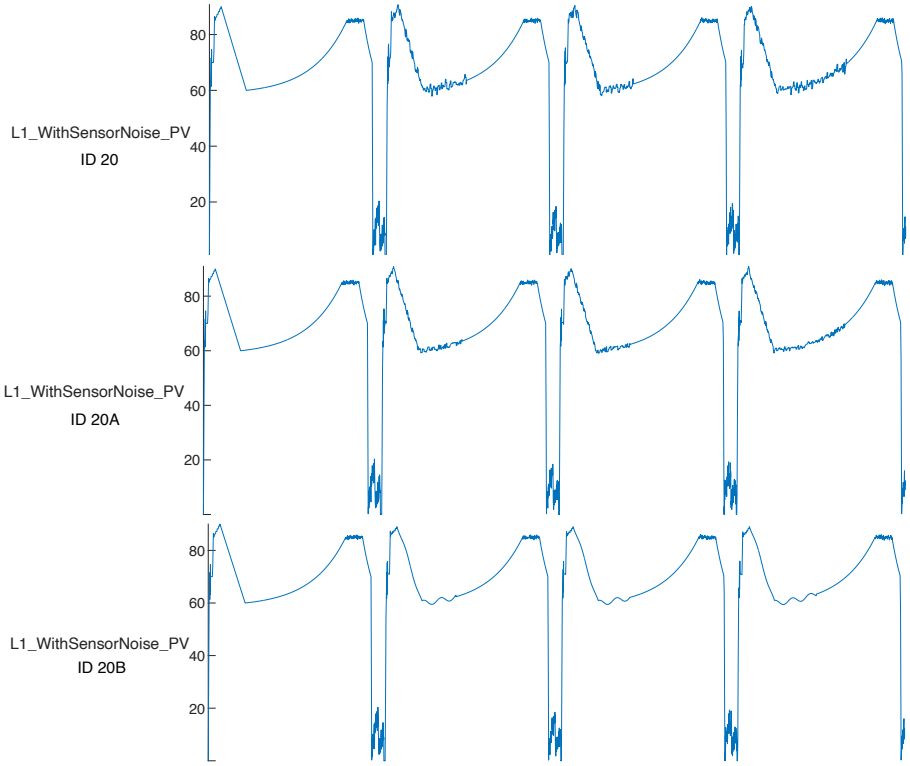


Figure 4.26: Stackedplot comparison of the relevant output variables to interpret the impact of Disturbances ID20, ID20A, and ID20B.

Considerable changes are detected in the level sensors as a result of particular sensor noise. For a human, and probably for the ML algorithm, such behavior is unusual and requires a proper explanation—the difference is: the human expert can recognize such behavior as a disturbance; up to the algorithm to find an alternative to process the data without injuring the optimal results.

4.2.16 Disturbance ID21: White-Band Noise on the level sensor

White noise is a random signal produced with equal intensities at different frequencies. Thereby, using two *Uniform Random Blocks* provides an equal intensity for the overall simulation: level varies between -5% and +5%. With low intensities on the white-noise, it is very difficult to visualize the disturbance in an entire batch. Thus, a close-up window is provided of the level measurement from Add_Solids to the beginning of $Reaction$. The right side image of Figure 4.28 provides the representation of the level sensor without ID21 being enabled.

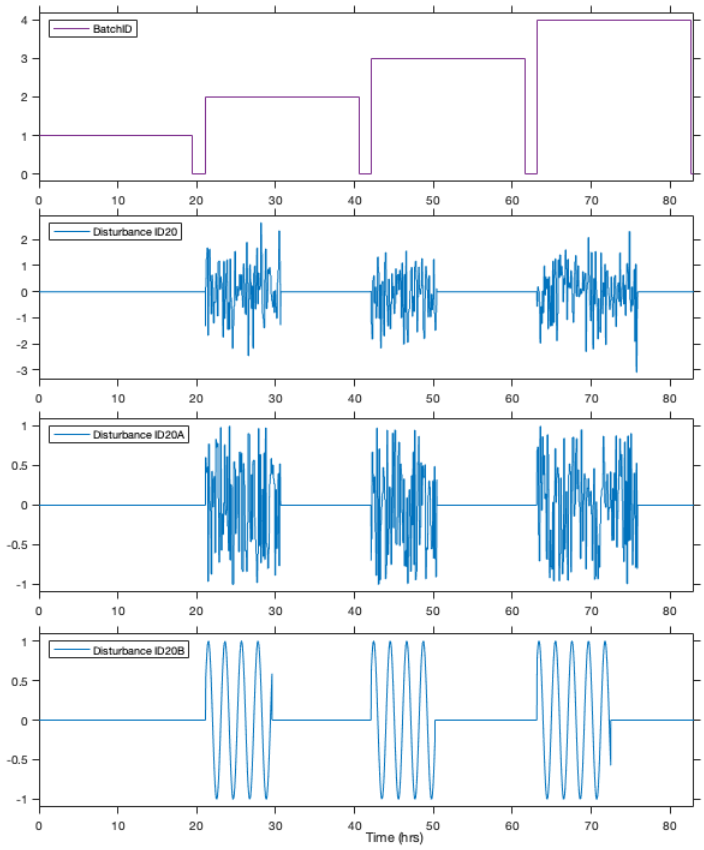


Figure 4.27: Batch and profile of the disturbance represented (ID20). Clear that the disturbance occurs at random batches, per consequence, at random times.

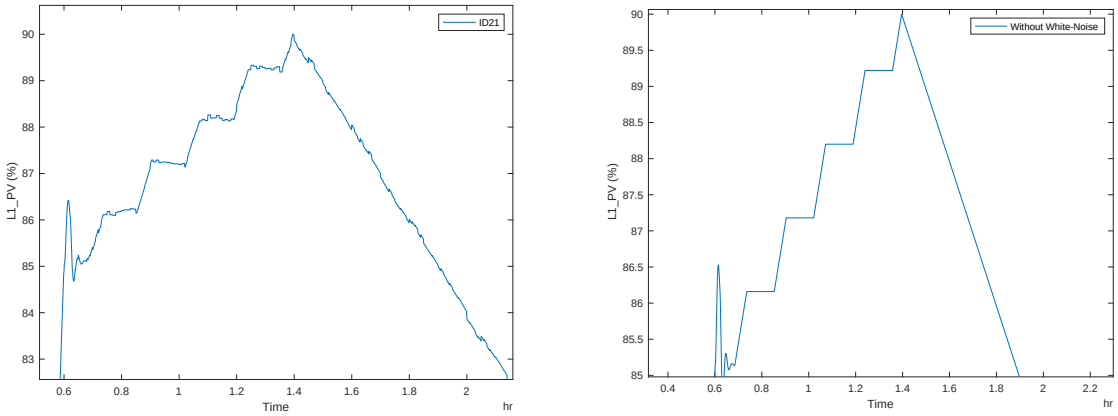


Figure 4.28: Stackedplot comparison of the relevant output variables to interpret the impact of Disturbance ID21.

4.3 Benchmark Process Feasibility Study

Simulating longer should provide the conclusion that the model implementation works independently of the number of batches being produced allowing for massive simulation studies. The stackedplot representation with the meaningful variables for a plus 40 batches production is introduced in Figure 4.29. Nonetheless, a lot of data is presented making it difficult for a detailed analysis.

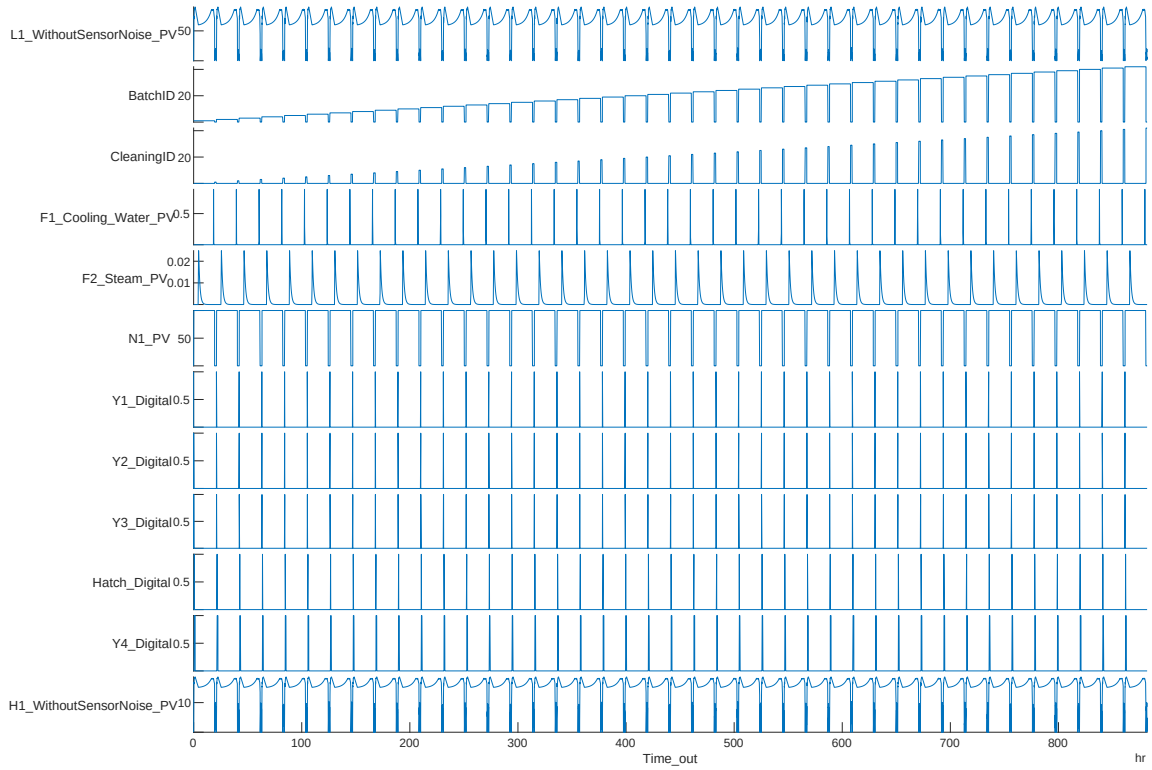


Figure 4.29: Nominal behavior for the relevant output variables of the simulation for the production of 42 batches.

The following section highlights a few of the most challenging profiles, disturbances, and noise for the ML algorithm with a brief explanation of how the algorithm should overcome them.

4.4 Machine Learning Feasibility Study

4.4.1 Single Disturbances Scenarios

By following a sequence with an increasing complexity level of disturbances scenarios, starting with “single” disturbances (each one implemented separately), each point in the sequence where the algorithm fails, is a good metric for its performance. Such arrangement is made regarding the disadvantages and limitations of the algorithms in this study, particularly of the HMM [37]:

- **Correlation with non-linear behaviors:** There is some difficulty for ML algorithms to recognize non-linear behaviors. For that reason, different types of level profiles were added: step, stair, exponential, and with noise (as discussed in section 3.1. Even though it is not a disturbance, non-linear profiles add more complexity to the model and should be recognized in a disturbance-free environment.

- **Discretization of continuous variables:** for the HMM, in particular, it is not possible to handle a combination of discrete and continuous variables. For the algorithm's training, one type of variable needs to be chosen—on this case, the user chose the discretization of continuous variables. Since discretization is accomplished by dividing time into smaller step-sizes, dynamic behaviors might be lost from step to step.

- **Dependence between two consecutive states:** adding label variability which affects the probability of an event to be recognized can be a challenge. To clarify, one example is provided: if the training data provided has missing labels (because one operation is masked as the previous one), the most difficult is the sequence to be correctly identified. This is probably the most challenging disturbance—an operation/phase label simply does not appear for the disturbed batch. As well as ID7, ID4 follows a similar line of thinking: specific phases occur irregularly affecting how the algorithm reads the recipe.

- **Not recommended to represent multiple overlapping features:** since it is a likelihood-based method, having multiple features linked to a singular state might affect the density estimation. One example might be ID11 where a new valve is switched on at random points in time.

  Disturbances with level variability might also affect the identification of states. If in a scenario, the volume of a batch increases by 20% at a random number of batches, the probability density of each operation will have a different minimum and maximum level measurement. This can lead to a misconception of labels leading to wrong identification.

- **Lack of understanding physical constrains:** not particular of the HMM but a general characteristic of ML algorithms. Despite the reference process being built for simulation purposes (as it does not represent an actual process), there are still physical behaviors to respect. Of a few, two can easily be understood: the level can not surpass more than 100% (flooding) and also can not have a negative value (dry). If training data is provided to the algorithm with the unfeasible physical behaviors discussed, no conclusion will be drawn, and the analysis will continue.

Attending the former, disturbances are presented in Figure 4.30, from top to bottom with an increasing level of complexity. In case the algorithm fails, the possible reason is presented on the dark blue block with a brief explanation on the dot-line block.

Figure 4.30: Benchmark Cycle for singular disturbances scenarios.

In conclusion, the most problematic disturbances scenarios will be the ones where **likelihood and dependency between consecutive states** are affected, meaning ID4, ID5, ID7, and ID11.

4.4.2 Benchmark Cycles with Multiple Disturbances Scenarios

After verifying the algorithm works properly with singular scenarios, these disturbances are paired up in groups with numerous degrees of difficulty: low to high frequencies, durations, and amplitudes. Until this point, levels 0 and 1 should have been correctly verified (from Table 4.1), *Level 2* reaching *Level 3* are paid more attention to.

Table 4.1: Level of increasing complexity for the implementation of disturbances scenarios.

Level	Description
0	No disturbances: prediction of only the reference model
1	Singular disturbances: implementation of "single" scenarios
2	Some disturbances: in most batches but at different times. If the training data is too similar to the evaluation data, the algorithm easily picks the operations/phases labels
3	More disturbances: different kinds in the same batch but always changing frequencies through batches, for example

By providing 21 singular disturbances scenarios, a large number of multiple scenarios can be made and it is up to the user to decide which. However, 6 cycles are presented as a challenge for the ML algorithm with increasing levels of complexity from cycles A to F—Figure 4.31. Note that Cycle F has the same parameters as Cycle E but, with an additional disturbance: ID7 (incorrect label).

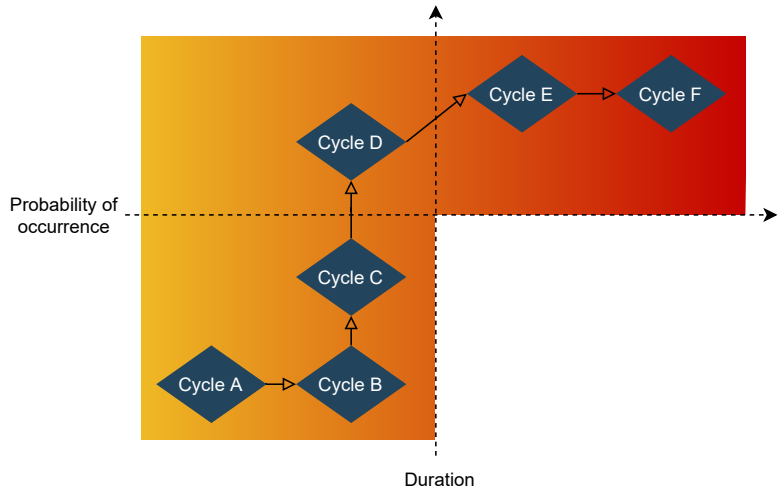


Figure 4.31: Benchmark Cycle for multiple disturbances scenarios.

Since variability and delays are the most common types of interferences in a real process, **ID1/ID2, ID5, ID12 and ID21 are enabled for all cycles**—only variability, duration, and number of affected operations change. vThe description of each cycle is presented in Table B.1 on Appendix B.

4.4.3 Machine Learning Algorithms Evaluation

The training and evaluation of the algorithms (HMM and RF) are not the purposes of the thesis, having this complementary study made in [5]. The multiple disturbances cycles properties discussed above are followed, being this section only to discuss those results.

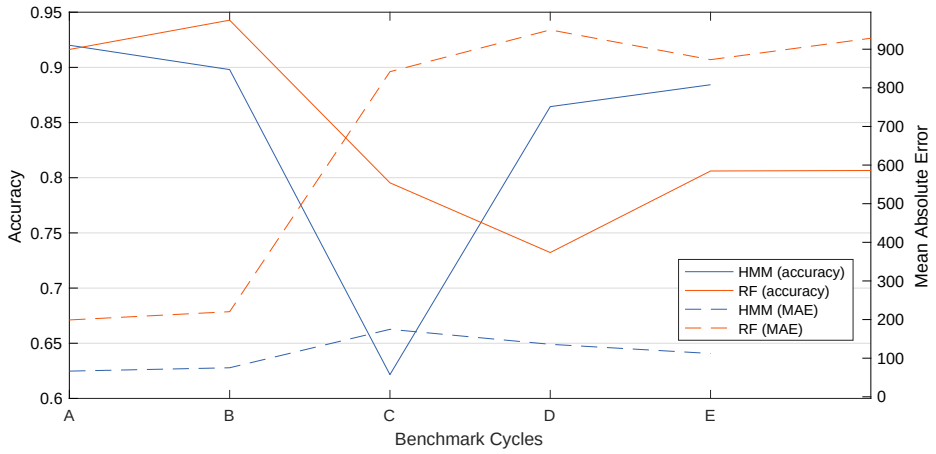


Figure 4.32: Evaluation of accuracy and MAE for each benchmark cycle. Both RF and HMM results are introduced [5].

An overview of the achieved accuracy and MAE is given in Figure 4.32. The following presents the conclusions that can be draw:

- Cycles A and B, as expected, present the best results for accuracy with the MAE decreasing slightly for the HMM on Cycle B. The overall results indicate the RF, in contrast to the HMM, is less able to generalize and tends to overfit. The increasing level of difficulty from Cycle A to F was almost followed for both algorithms.

- For Cycle A, the HMM correctly estimates the number of change points, and the recipe sequence is recognized and adhered to. Nonetheless, there is confusion where the change points occur being incorrectly estimated. Tables 6.1 and 6.2 of [5] allow a demonstration of the Confusion Matrix—a summary of the prediction results on a classification problem. The most problematic transitions are from $Centrifugation$ to $Product\_Transfer$ and between all phases of the cleaning procedure. Between the HMM and RF, the latter confuses these state transitions more often than the HMM. This confusion between states might be due to heavy noise being added the signal for the CIP—the overlap of volumes for consecutive states is a limitation; or, a step profile in the case of $Centrifugation$.

- Cycle C was the most challenging cycle, especially for the HMM. Because the implementation of ID15 is set to be independent on the number of batches, fewer occurrences of the CIP ended

up affecting the level profile for a longer time. While the RF is able to order input characteristics according to their influence on the classification decision [31], the HMM, always outputs the same phase for these batches.

- The HMM could not make any predictions for Cycle F because individual observations could not be assigned to any state. By enabling a label disturbance, this algorithm failed completely in recognizing states.
- Overall, the RF confuses more states and accordingly produces more change points (higher MAE); whereas, the HMM is still able to identify the correct sequence. This is particularly relevant when ID4 is active for Cycles C to F—where changes in the recipe are made.
- In summary, the HMM delivers accuracies for four of the six benchmark cycles. Even though the accuracy does not reach the 96% required in Section 2.4.3, results are similarly reaching the most of 92% for Cycle A.

5

Conclusion

Batch processes are a relevant production paradigm for pharmaceutical and fine-products industries. As shown, with the benchmark model and the disturbances mapping discussed in previous sections, a series of challenges in the process industry can be addressed. Some of the major issues arising during operation and achieving of process data are due to deviation in production, changeover, and cleaning times; deviation in product specification; and, equipment failure and malfunction.

The focus of this work was to develop a benchmark model to support the testing and comparison of ML methods in their description of batch processes. The entire process is viewed at the unit level with filling, processing, draining, and cleaning operations. The model was then used to generate data where active challenges, disturbances, and data noise are fully controlled. Different scenarios were generated where up to 21 disturbances of several types, causes, or fault origins are injected into the simulated data for testing. The scenarios were grouped into six benchmark cycles (A to F) with increasing levels of complexity in terms of intensity, duration, and probability.

A good balance was found by implementing the process sequence and the disturbances using both continuous and discrete environments. Simulink® and Stateflow® allow simulating faster and massive simulation scenarios with low CPU in comparison with mechanistic models. As it is not the task of the algorithms to learn heat balance and kinetics, the abstraction was made to not contemplate them. Also, a trade-off between model complexity and user-friendliness was chosen by implementing a process recipe with the current number of valves, event labels, and sensors; having such a model allows for meaningful studies which can be understood by a wide range of users.

From the algorithm's training point of view, the disturbances which caused the most damage on the algorithm training were: ID7 (purpose incorrectly labeled training data) and ID15 (loss of several signal points) combined with ID4 (phase occurring irregularly). Within the scope of the work, it is clear that HMM was able to deal with most of the implemented process disruptions and was able to make better predictions than the RF. In particular, it was shown that the HMM can learn the sequence of the recipe steps and correctly assess the number of phase transitions. Errors arise in determining the correct point in time for the phase change. For two of the six implemented test cycles, the performance of the HMM came short since it could not make any predictions (Cycle F with ID7) or only poor predictions (Cycle C with ID15). On the other hand, the RF was not affected, having accuracies of 81%/79%, respectively.

The simulation model introduced can be used for the training of ML algorithms for the identification of batch phases but also as a basis for further complexity to arise on the implementation level. Several aspects can change in the implementation of the simulation model to approach closely a real batch process, including [8]:

1. A valve can be open to a certain percentage or a certain degree. For example, even though the valve is opened 10%, changes might not be detected until it reaches 15% or 20% of the opening (air blockage or slugging behaviors are mimicked);

2. Inclusion of equilibrium and kinetically-controlled chemical reactions. If so, the complexity of the process and disturbances can be broadly increased. For instance, ID3 might have an additional reaction step if the yield is not sufficient to move forward in the operation;

3. Temperature monitoring where effects of heat transfer on the system dynamics are taken into account;

4. Study of the hierarchical control by fixing set points, and manipulated and controlled variables. Not many changes would be needed as Simulink® provides the necessary blocks for the implementation;

5. More non-linear profiles can be added to the recipe as steps, stairs, or noisy profiles. As discussed in section 2.4.3, these are the more problematic transitions.

Having discussed already the future endeavors of the first benchmark model, a few more suggestions are given for an upcoming use of this work as reference. Firstly, the second benchmark model for batch tracking purposes should be properly introduced being provided a detailed explanation in Appendix A.2.1. Essentially, a more complex problem is made acquainted: instead of transitions being dependent on the number of batches, the recipe to follow can depend only on the vessel level. One example could be, the material goes always from Unit 1 to Unit 2.1. This is only set to change if Unit 2.1 is filled, not being possible to receive a new batch. In that case, Unit 2.2 is used and, at last, Unit 2.3.

Bibliography

[1] J. Hahn and T. F. Edgar, *Process Control*. John Wiley & Sons, 2014, pp. 1–44.

[2] ISA, "ISA-95: Enterprise-Control System Integration Part 1: Models and Terminology," 2000.

[3] A. Gron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2017.

[4] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. Springer, 2008.

[5] G. Just, "Zeitreihenanalyse zur erkennung von batchphasen in der prozessindustrie," *Technische Universität Dresden*, 2021.

[6] D. Green and M. Z. Southard, *Perry's Chemical Engineers' Handbook*, 9th ed. New York: McGraw-Hill Education, 2018.

[7] E. D. Seborg, T. F. Edgar, and D. A. Mellichamp, *Process Dynamics and Control*, 4th ed. John Wiley & Sons, 2017.

[8] A. Stief, R. Tan, Y. Cao, J. R. Ottewill, N. F. Thornhill, and J. Baranowski, "A heterogeneous benchmark dataset for data analytics: Multiphase flow facility case study," *Journal of Process Control*, vol. 79, pp. 41–55, 2019.

[9] E. Korovessi and A. A. Linninger, *Batch Processes*. Taylor & Francis, 2005.

[10] J. Schumacher, "A framework for batch-operation analysis within the context of disturbance management," *Computers & Chemical Engineering*, vol. 24, pp. 1175–1180, 2000.

[11] K. Kidam and M. Hurme, "Analysis of equipment failures as contributors to chemical process accidents," *Process Safety and Environmental Protection*, vol. 91, no. 1, pp. 61–78, 2013.

[12] P. Kadlec, B. Gabrys, and S. Strandt, "Data-driven Soft Sensors in the process industry," *Comput. Chem. Eng.*, vol. 33, pp. 795–814, 2009.

[13] C. A. Floudas and X. Lin, “Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review,” *Computers & Chemical Engineering*, vol. 28, pp. 2109–2129, 2004.

[14] F. D. Bähner and J. K. Huusom, “A Debottlenecking Study of an Industrial Pharmaceutical Batch Plant,” *Ind. Eng. Chem. Res.*, vol. 58, no. 43, pp. 20 003–20 013, 2019.

[15] S. Lee, T. O’Connor, X. Yang, C. N. Cruz, S. Chatterjee, R. Madurawe, C. M. Moore, L. X. Yu, and J. Woodcock, “Modernizing Pharmaceutical Manufacturing: from Batch to Continuous Production,” *Journal of Pharmaceutical Innovation*, vol. 10, pp. 191–199, 2015.

[16] G. Georgiadis, A. Elekidis, and M. Georgiadis, “Optimization-Based Scheduling for the Process Industries: From Theory to Real-Life Industrial Applications,” *Processes*, vol. 7, no. 7, 2019.

[17] L. L. Simon, N. Osterwalder, U. Fischer, and K. Hungerbuehler, “Systematic Retrofit Method for Chemical Batch Processes Using Indicators, Heuristics, and Process Models,” *Ind. Eng. Chem. Res.*, vol. 47, no. 1, pp. 66–80, 2008.

[18] D. Petrides, D. Carmichael, C. Siletti, and A. Koulouris, “Biopharmaceutical process optimization with simulation and scheduling tools,” *Bioengineering*, vol. 1, no. 4, pp. 154–187, 2014.

[19] S. Amaran, B. Sharda, and S. J. Bury, “Targeted incremental debottlenecking of batch process plants,” in *2016 Winter Simulation Conference (WSC)*, 2016, pp. 2924–2934.

[20] D. van Beek and J. Rooda, “Languages and Applications in Hybrid Modelling and Simulation: Positioning of Chi,” *Control Engineering Practice*, vol. 8, no. 1, pp. 81–91, 2000.

[21] M. Barrera and L. Evans, “Optimal Design and Operation of Batch Processes,” *Chem. Eng. Commun.*, vol. 82, no. 1, pp. 45–66, 1989.

[22] D. Bonvin and D. Hunkeler, “Control and optimization of batch processes: Improvement of process operation in the production of specialty chemicals,” *IEEE Control Systems Magazine.*, pp. 1–6, 2006.

[23] J. Downs and E. Vogel, “A plant-wide industrial process control problem,” *Computers & Chemical Engineering*, vol. 17, no. 3, pp. 245–255, 1993.

[24] ISA, “ISA-88: Batch Control Part 1: Models and Terminology,” 1995.

[25] J. Marzat, H. Piet-Lahanier, F. Damongeot, and E. Walter, “Model-based fault diagnosis for aerospace systems: A survey,” *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 226, no. 10, pp. 1329–1360, 2012.

[26] N. Ricker, “Optimal steady-state operation of the tennessee eastman challenge process,” *Computers & Chemical Engineering*, vol. 19, no. 9, pp. 949–959, 1995.

[27] F. D. Bähner, O. A. Prado-Rubio, and J. K. Huusom, “Discrete-continuous dynamic simulation of plantwide batch process systems in matlab,” *Chem. Eng. Res. Des.*, vol. 159, pp. 66–77, 2020.

[28] R. S. Tsay and R. Chen, *Nonlinear time series analysis*. John Wiley & Sons, Inc., 2019.

[29] A. Munther, R. R. Othman, A. S. Alsaadi, and M. Anbar, “A performance study of hidden markov model and random forest in internet traffic classification,” in *Information Science and Applications (ICISA) 2016*, K. J. Kim and N. Joukov, Eds. Singapore: Springer Singapore, 2016, pp. 319–329.

[30] L. Rabiner and B. Juang, “An introduction to hidden markov models,” *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[31] R. T. Trevor Hastie and J. Friedman, “The Elements of Statistical Learning Data Mining, Inference, and Prediction,” *Springer Science & Business Media*, 2009.

[32] J. Speiser, M. Miller, J. Tooze, and E. Ip, “A Comparison of Random Forest Variable Selection Methods for Classification Prediction Modeling,” *Expert Systems with Applications*, vol. 134, pp. 93–101, 2019.

[33] Z. He, Z. Wang, W. Wei, S. Feng, X. Mao, and S. Jiang, “A Survey on Recent Advances in Sequence Labeling from Deep Learning Models,” 2017.

[34] M. K. Mustafa, T. Allen, and K. Appiah, “A comparative review of dynamic neural networks and hidden markov model methods for mobile on-device speech recognition,” *Neural Computing and Applications*, vol. 31, pp. 891–899, 2020.

[35] Solver - matlab & simulink. Accessed in July 21st of 2021. [Online]. Available: https://www.mathworks.com/help/simulink/gui/solver.html

[36] What Is Sample Time? - MATLAB & Simulink. Accessed in August 21st of 2021. [Online]. Available: https://www.mathworks.com/help/simulink/ug/what-is-sample-time.html

[37] I. Visser, “Seven things to remember about hidden markov models: A tutorial on markovian models for time series,” *Journal of Mathematical Psychology*, vol. 55, no. 6, pp. 403–415, 2011.

A

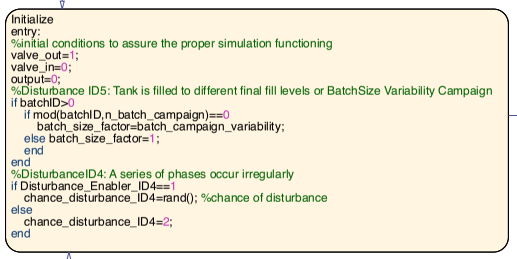
Graphical Representation of the Stateflow® Implementation and Models Further Description

A.1 Benchmark Model 1.0

A.1.1 Stateflow® Implementation

Batch Size variability campaign

Batch campain with 10 batches. After each 10th batch, the size of a batch decreases in `batch_size_factor` (called from MATLAB).



Disturbances are implemented as a probability of succeeding: `probabilityThreshold`. The value is chosen in the MATLAB environment and only that percentage of batches will be affected for the disturbance in particular.

Only one example will be explained since **all disturbances follow the same line of thinking**.

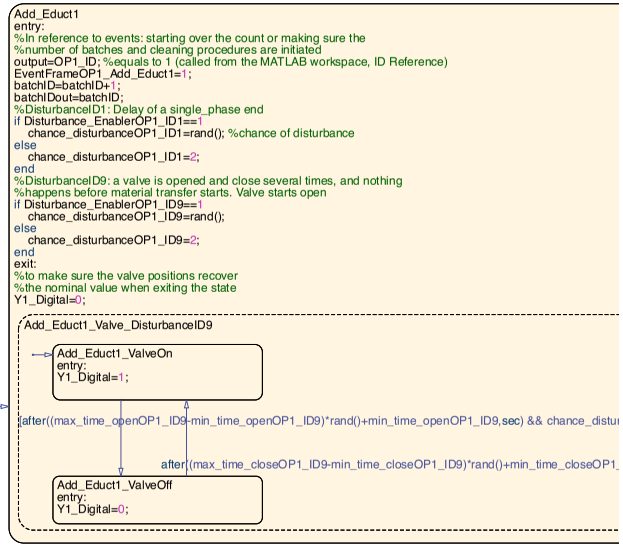
For Disturbance ID1: Delay of a single-phase end or event multiple phases/operations. If the control enabler is on (`Disturbance_EnablerOP1_ID1=1`), a random probability will be set within the StateChart (`chance_disturbanceOP1_ID1`). Whenever this value is less than the `probabilityThreshold`, the disturbance is enabled and the transition will be from `Add_Educt1` to `WaitingOP1`.

On the Waiting block, the behavior is supposed to remain the same. As such, the valve position will continue the behavior set before.

The delay in time will also be completely random.

[VolumeP<=0]

At the start time, the initial height should be zero. Because simulink doesn't count the zero crossing feature, it is necessary to add the condition <=0



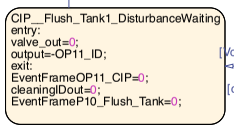
Inactivity Period:

after each cleaning procedure and before the production of a new batch, an inactivity period is set. The duration is a random number between `min_inactivity_period` and `max_inactivity_period` (when `control_inactivity=1`). If `control_inactivity=0`, `inactivity_period=min_inactivity_period`

ivity_period)*rand()*control_inactivity+min_inactivity_period,sec)

[chance_disturbanceOP11_ID1>=probabilityThreshold_DisturbanceOP11_ID1]
{valve_out=0;
EventFrameOP11_CIP=0;
cleaningIDout=0;
EventFrameP10_Flush_Tank=0;}

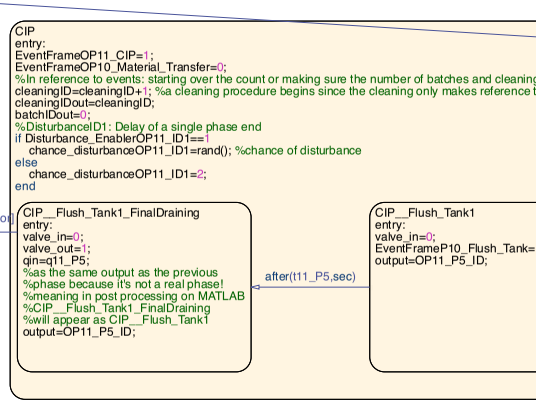
[after(abs(randn()*delayFactorOP11),sec)]

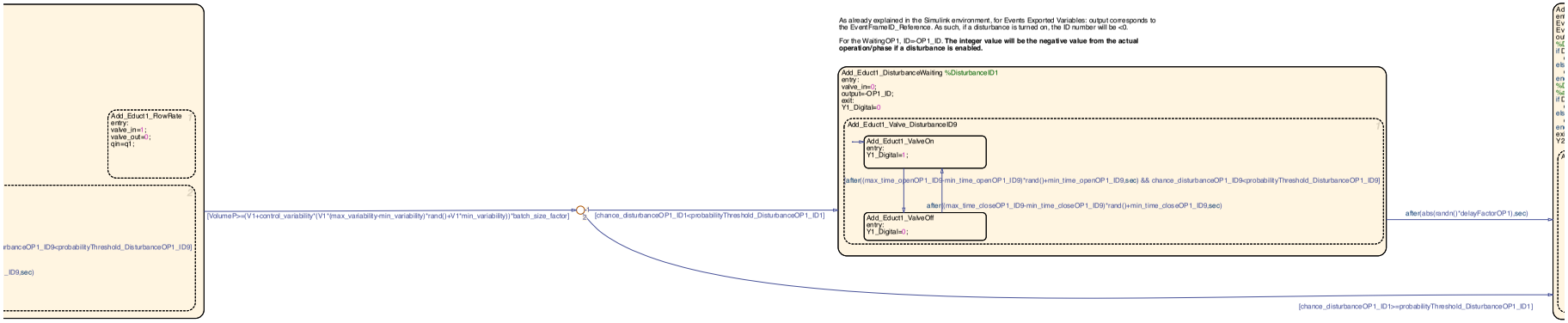


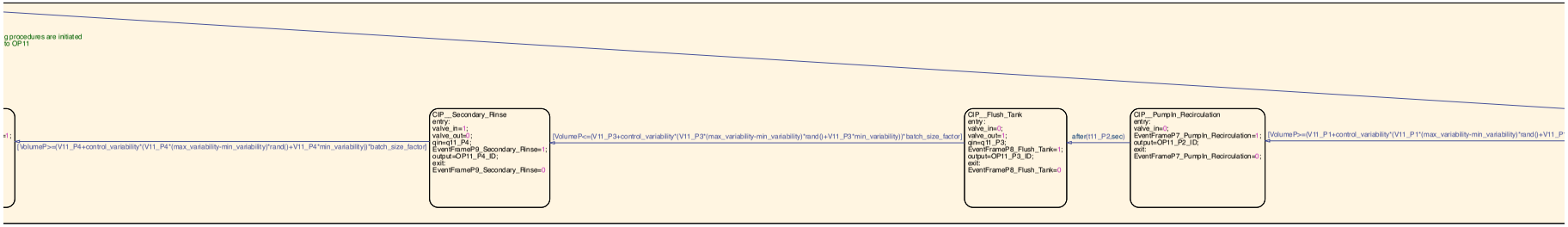
[VolumeP<=(V11_P5+control_variability*(V11_P5*(max_variability-min_variability)*rand()+V11_P5*min_variability))*batch_size_factor]

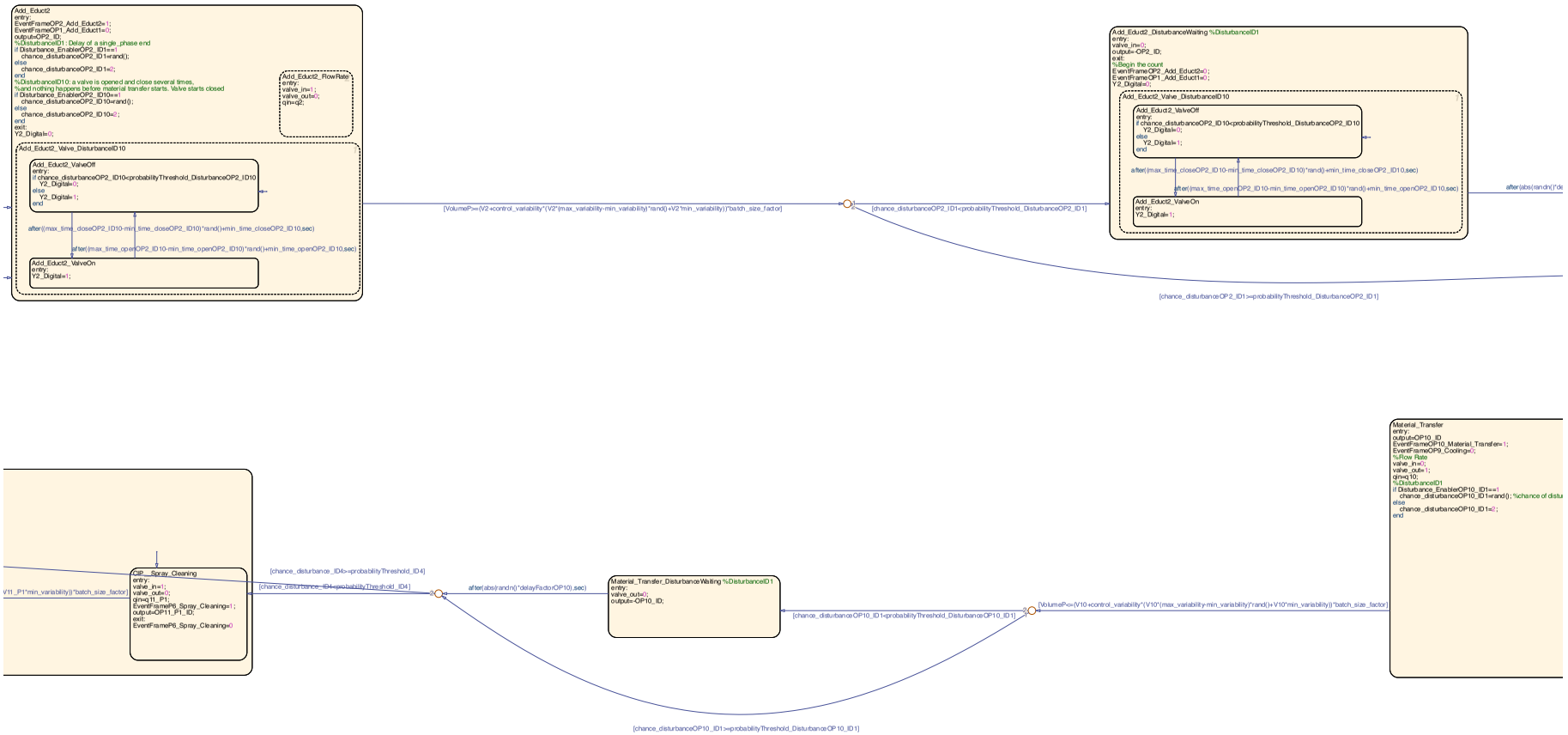
[chance_disturbanceOP11_ID1<probabilityThreshold_DisturbanceOP11_ID1]

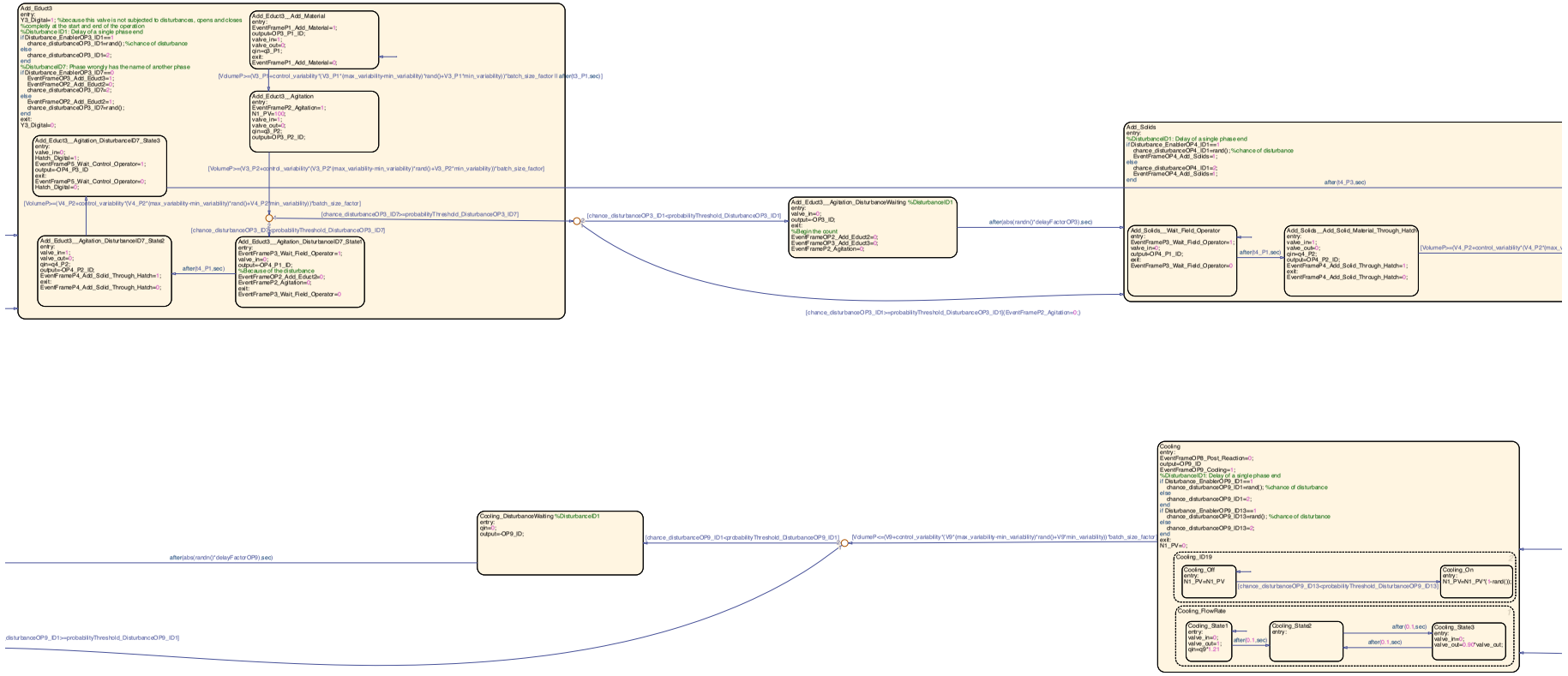
cleaningIDout=0: After a cleaning procedure ends, it is necessary to restart the count for the number of batches in respect to the end of previous events




Add_Educt1_RowRate
entry:
valve_in=1;
valve_out=0;
qin=q1;
rbanceOP1_ID9<probabilityThreshold_DisturbanceOP1_ID9]
_ID9,sec)
[VolumeP>=(V1+control_variability*(V1*(max_variability-min_variability)*rand()+V1*min_variability))*batch_size_factor]
[chance_disturbanceOP1_ID1<probabilityThreshold_DisturbanceOP1_ID1]
As already explained in the Simulink environment, for Events Exported Variables: output corresponds to the EventFrameID_Reference. As such, if a disturbance is turned on, the ID number will be <0.
For the WaitingOP1, ID=-OP1_ID. The integer value will be the negative value from the actual operation/phase if a disturbance is enabled.
Add_Educt1_DisturbanceWaiting %DisturbanceID1
entry:
valve_in=0;
output=-OP1_ID;
exit:
Y1_Digital=0
Add_Educt1_Valve_DisturbanceID9
Add_Educt1_ValveOn
entry:
Y1_Digital=1;
after((max_time_openOP1_ID9-min_time_openOP1_ID9)*rand()+min_time_openOP1_ID9,sec) && chance_disturbanceOP1_ID9<probabilityThreshold_DisturbanceOP1_ID9]
after((max_time_closeOP1_ID9-min_time_closeOP1_ID9)*rand()+min_time_closeOP1_ID9,sec)
Add_Educt1_ValveOff
entry:
Y1_Digital=0;
after(abs(randn()*delayFactorOP1),sec)
[chance_disturbanceOP1_ID1>=probabilityThreshold_DisturbanceOP1_ID1]


g procedures are initiated
to OP11
=1;
[VolumeP>=(V11_P4+control_variability*(V11_P4*(max_variability-min_variability)*rand()+V11_P4*min_variability))*batch_size_factor]
CIP__Secondary_Rinse
entry:
valve_in=1;
valve_out=0;
qin=q11_P4;
EventFrameP9_Secondary_Rinse=1;
output=OP11_P4_ID;
exit:
EventFrameP9_Secondary_Rinse=0
[VolumeP<=(V11_P3+control_variability*(V11_P3*(max_variability-min_variability)*rand()+V11_P3*min_variability))*batch_size_factor]
CIP__Flush_Tank
entry:
valve_in=0;
valve_out=1;
qin=q11_P3;
EventFrameP8_Flush_Tank=1;
output=OP11_P3_ID;
exit:
EventFrameP8_Flush_Tank=0
after(t11_P2,sec)
CIP__PumpIn_Recirculation
entry:
valve_in=0;
EventFrameP7_PumpIn_Recirculation=1;
output=OP11_P2_ID;
exit:
EventFrameP7_PumpIn_Recirculation=0;
[VolumeP>=(V11_P1+control_variability*(V11_P1*(max_variability-min_variability)*rand()+V11_P


Add_Educt2
entry:
EventFrameOP2_Add_Educt2=1;
EventFrameOP1_Add_Educt1=0;
outputs=OP2_ID;
%Disturbance ID1 : Delay of a single_phase end
if Disturbance_EnablerOP2_ID1==1
chance_disturbanceOP2_ID1=rand();
else
chance_disturbanceOP2_ID1=2;
end
%DisturbanceID10: a valve is opened and close several times,
%and nothing happens before material transfer starts. Valve starts closed
if Disturbance_EnablerOP2_ID10==1
chance_disturbanceOP2_ID10=rand();
else
chance_disturbanceOP2_ID10=2;
end
exit:
Y2_Digital=0;
Add_Educt2_RowRate
entry:
valve_in=1;
valve_out=0;
q=q2;
Add_Educt2_Valve_DisturbanceID10
Add_Educt2_ValveOff
entry:
if chance_disturbanceOP2_ID10<probabilityThreshold_DisturbanceOP2_ID10
Y2_Digital=0;
else
Y2_Digital=1;
end
after((max_time_closeOP2_ID10-min_time_closeOP2_ID10)*rand()+min_time_closeOP2_ID10,sec)
after((max_time_openOP2_ID10-min_time_openOP2_ID10)*rand()+min_time_openOP2_ID10,sec)
Add_Educt2_ValveOn
entry:
Y2_Digital=1;
[VolumeP>=(V2+control_variability*(V2*(max_variability-min_variability)*rand()+V2*min_variability))*batch_size_factor]
[chance_disturbanceOP2_ID1<probabilityThreshold_DisturbanceOP2_ID1]
Add_Educt2_DisturbanceWaiting %DisturbanceID1
entry:
valve_in=0;
outputs=OP2_ID;
exit:
%Begin the count
EventFrameOP2_Add_Educt2=0;
EventFrameOP1_Add_Educt1=0;
Y2_Digital=2;
Add_Educt2_Valve_DisturbanceID10
Add_Educt2_ValveOff
entry:
if chance_disturbanceOP2_ID10<probabilityThreshold_DisturbanceOP2_ID10
Y2_Digital=0;
else
Y2_Digital=1;
end
after((max_time_closeOP2_ID10-min_time_closeOP2_ID10)*rand()+min_time_closeOP2_ID10,sec)
after((max_time_openOP2_ID10-min_time_openOP2_ID10)*rand()+min_time_openOP2_ID10,sec)
Add_Educt2_ValveOn
entry:
Y2_Digital=1;
after(abs(randn()*de
[chance_disturbanceOP2_ID1>=probabilityThreshold_DisturbanceOP2_ID1]
V11_P1*min_variability))*batch_size_factor]
CIP__Spray_Cleaning
entry:
valve_in=1;
valve_out=0;
q=q11_P1;
EventFrameP6_Spray_Cleaning=1;
outputs=OP11_P1_ID;
exit:
EventFrameP6_Spray_Cleaning=0
[chance_disturbance_ID4>=probabilityThreshold_ID4]
[chance_disturbance_ID4<probabilityThreshold_ID4]
after(abs(randn()*delayFactorOP10),sec)
Material_Transfer_DisturbanceWaiting %DisturbanceID1
entry:
valve_out=0;
outputs=OP10_ID;
[chance_disturbanceOP10_ID1<probabilityThreshold_DisturbanceOP10_ID1]
[VolumeP<=(V10+control_variability*(V10*(max_variability-min_variability)*rand()+V10*min_variability))*batch_size_factor]
[chance_disturbanceOP10_ID1>=probabilityThreshold_DisturbanceOP10_ID1]
Material_Transfer
entry:
outputs=OP10_ID
EventFrameOP10_Material_Transfer=1;
EventFrameOP9_Cooling=0;
%Flow Rate
valve_in=0;
valve_out=1;
q=q10;
%DisturbanceID1
if Disturbance_EnablerOP10_ID1==1
chance_disturbanceOP10_ID1=rand(); %chance of distu
else
chance_disturbanceOP10_ID1=2;
end


Add_Educt3
entry:
Y3_Digital=1; %because this valve is not subjected to disturbances, opens and closes
%completely at the start and end of the operation
%Disturbance ID1: Delay of a single phase end
If Disturbance_EnablerOP3_ID1==1
chance_disturbanceOP3_ID1=rand(); %chance of disturbance
else
chance_disturbanceOP3_ID1=2;
end
%DisturbanceID7: Phase wrongly has the name of another phase
If Disturbance_EnablerOP3_ID7==2
EventFrameOP3_Add_Educt3=1;
EventFrameOP2_Add_Educt2=2;
chance_disturbanceOP3_ID7=2;
else
EventFrameOP2_Add_Educt2=1;
chance_disturbanceOP3_ID7=rand();
end
exit:
Y3_Digital=0;
Add_Educt3__Add_Material
entry:
EventFramP1_Add_Material=1;
outputsOP3_P1_ID;
valve_in=1;
valve_out=2;
qin=q3_P1;
exit:
EventFramP1_Add_Material=2;
[VolumeP>=(V3_P1*(1+control_variability*(V3_P1*(max_variability-min_variability)*rand()+V3_P1*min_variability))*batch_size_factor || after(t3_P1,sec)]
Add_Educt3__Agitation
entry:
EventFrameP2_Agitation=1;
N1_PV=100;
valve_in=1;
valve_out=2;
qin=q3_P2;
outputs=OP3_P2_ID;
[VolumeP>=(V3_P2*(1+control_variability*(V3_P2*(max_variability-min_variability)*rand()+V3_P2*min_variability))*batch_size_factor]
Add_Educt3__Agitation_DisturbanceID7_State3
entry:
valve_in=0;
Hatch_Digital=1;
EventFrameP5_Wat_Control_Operator=1;
outputs=OP4_P3_ID
exit:
EventFrameP5_Wat_Control_Operator=0;
Hatch_Digital=0;
[VolumeP>=(V4_P2*(1+control_variability*(V4_P2*(max_variability-min_variability)*rand()+V4_P2*min_variability))*batch_size_factor]
[chance_disturbanceOP3_ID7>probabilityThreshold_DisturbanceOP3_ID7]
[chance_disturbanceOP3_ID7<probabilityThreshold_DisturbanceOP3_ID7]
Add_Educt3__Agitation_DisturbanceID7_State2
entry:
valve_in=1;
valve_out=0;
qin=q4_P2;
outputs=OP4_P3_ID;
EventFrameP4_Add_Solid_Through_Hatch=1;
exit:
EventFrameP4_Add_Solid_Through_Hatch=0;
after(4_P1,sec)
Add_Educt3__Agitation_DisturbanceID7_State1
entry:
EventFrameP3_Wait_Field_Operator=1;
valve_in=0;
outputs=OP4_P1_ID;
%Because of the disturbance
EventFrameOP2_Add_Educt2=0;
EventFrameP2_Agitation=0;
exit:
EventFrameP3_Wait_Field_Operator=0
[chance_disturbanceOP3_ID1>=probabilityThreshold_DisturbanceOP3_ID1]
Add_Educt3__Agitation_DisturbanceWaiting %DisturbanceID1
entry:
valve_in=0;
output=OP3_ID;
exit:
%Begin the count
EventFrameOP2_Add_Educt2=2;
EventFrameOP3_Add_Educt3=2;
EventFrameP2_Agitation=2;
after(abs(randn()*delayFactorOP3),sec)
Add_Sdkds
entry:
%DisturbanceID1: Delay of a single phase end
If Disturbance_EnablerOP4_ID1==1
chance_disturbanceOP4_ID1=rand(); %chance of disturbance
EventFrameOP4_Add_Sdkds=1;
else
chance_disturbanceOP4_ID1=2;
EventFrameOP4_Add_Sdkds=1;
end
after(t4_P3,sec)
Add_Solids__Wait_Field_Operator
entry:
EventFrameP3_Wait_Field_Operator=1;
valve_in=0;
outputs=OP4_P1_ID;
exit:
EventFrameP3_Wait_Field_Operator=0
after(t4_P1,sec)
Add_Solids__Add_Solid_Material_Through_Hatch
entry:
valve_in=1;
valve_out=2;
qin=q4_P2;
outputs=OP4_P2_ID;
EventFramePF4_Add_Solid_Through_Hatch=1;
exit:
EventFramePF4_Add_Solid_Through_Hatch=0;
[VolumeP>=(V4_P2+control_variability*(V4_P2*(max_
[chance_disturbanceOP3_ID1>=probabilityThreshold_DisturbanceOP3_ID1]||EventFrameP2_Agitation==0;)

Cooling_DisturbanceWaiting %DisturbanceID1
entry:
qin=0;
output=OP9_ID;
after(abs(.randn()*delayFactorOP9),sec)
[chance_disturbanceOP9_ID1<probabilityThreshold_DisturbanceOP9_ID1]
[VolumeP<=(V9+control_variability*(V9*(max_variability-min_variability)*rand()+V9*min_variability))*batch_size_factor
Cooling
entry:
EventFrameOP8_Post_Reaction=0;
output=OP9_ID
EventFrameOP9_Cooling=1;
%DisturbanceID1: Delay of a single phase end
If Disturbance_EnablerOP9_ID1==1
chance_disturbanceOP9_ID1=rand(); %chance of disturbance
else
chance_disturbanceOP9_ID1=2;
end
If Disturbance_EnablerOP9_ID13==1
chance_disturbanceOP9_ID13=rand(); %chance of disturbance
else
chance_disturbanceOP9_ID13=2;
end
exit:
N1_PV=0;
Cooling_ID19
Cooling_Off
entry:
N1_PV=N1_PV
[chance_disturbanceOP9_ID13>=probabilityThreshold_DisturbanceOP9_ID13]
Cooling_On
entry
N1_PV=N1_PV*(1-rand());
Cooling_FlowRate
Cooling_State1
entry:
valve_in=0;
valve_out=1;
qin=q9*1.21;
after(0.1,sec)
Cooling_State2
entry:
after(0.1,sec)
after(0.1,sec)
Cooling_State3
entry:
valve_in=0;
valve_out=1.50*valve_out;
disturbanceOP9_ID1>=probabilityThreshold_DisturbanceOP9_ID1]

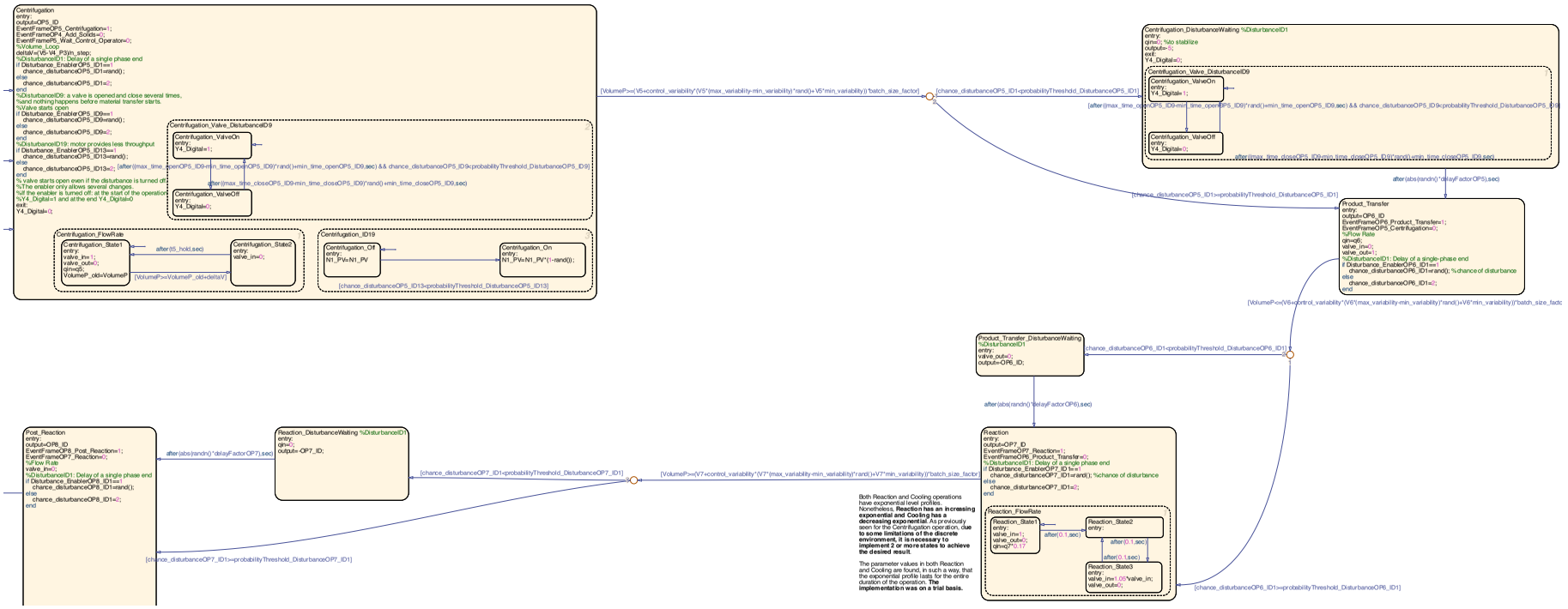


Figure A.1: Stateflow® environment for the implementation of the Benchmark Model 1.0

A.2 Benchmark Model 2.0

Worth remembering, the first benchmark model's purpose was to correctly identify batch phases while complexity arose on both level profiles and disturbances. For this model, the goal is for the algorithm to identify phases with simple profiles while material moves through different units—*batch tracking*.

A.2.1 Model Description

The Benchmark Model 2.0 consists of several operations wherein the entire process is viewed at the plant-level in 4 units: $Unit\ 1$, $Unit\ 2.1$, $Unit\ 2.2$, and $Unit\ 2.3$, as depicted in the process flow diagram provided in Figure A.2. These last three units functioning is in parallel, which means, depending on the number of the batch being produced, only one is active. Liquid raw materials and solids are being used for the reaction step, with each being associated with a single unit with exception of Educt2 and Educt3—as $Units\ 2.1$ and 2.2 have the same characteristics (raw materials, duration, and volumetric constraints). Only the unit occurrence is affected. Once again, the process model here presented is purely conceptual wherein the goal is to have a working benchmark model that captures the complexity of a plant-level production where batches are produced in multiple units (for batch tracking).

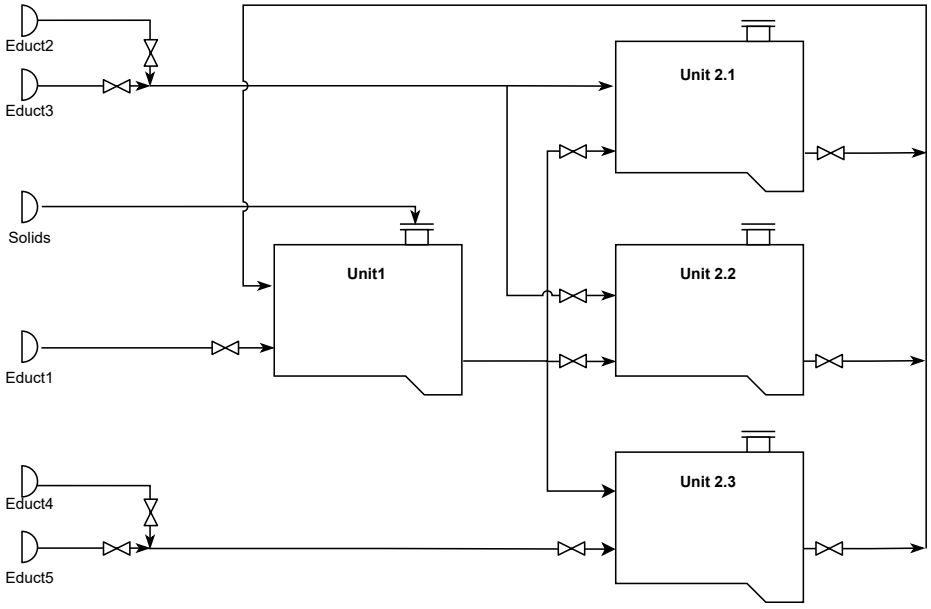


Figure A.2: Process Flow Diagram of the Benchmark Model 2.0.

Although some non-linear profiles are added to the recipe, the level measurement of each unit is rather simple compared to $Model\ 1.0$. The process recipe for $Model\ 2.0$ consists of fewer phases and operations with Figure A.3 illustrating the path, operations (dark blue), and phases (light blue). A batch is produced by moving forward from $Unit\ 1$ to one of the other three left remaining.

It is worth mention, simulations are only limited to mass balances as discussed for the last model (chemical reactions and kinetics present no value here). Also, vessel properties as height (logarithmic following linear) and maximum volume remain with the same values as before.

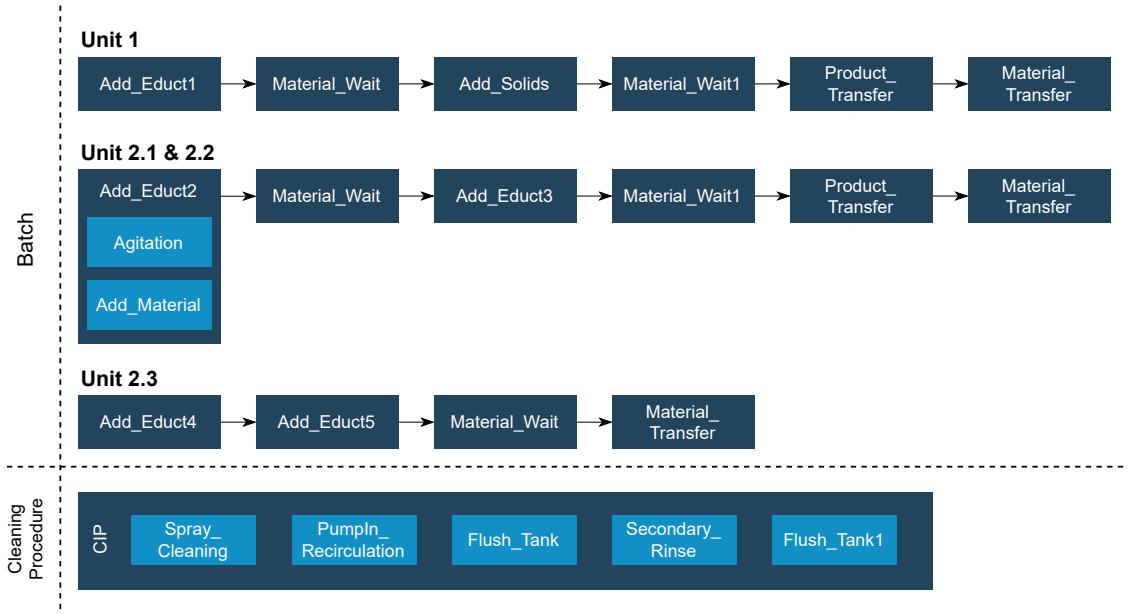


Figure A.3: Benchmark Model 2.0 Recipe including all operations and phases.

After establishing the recipe, duration, volume conditions, and units occurrence are specified for the model implementation. Once more, the flow rate is adjusted as a degree of freedom to connect volume with time. A description of the operational procedures in terms of constraints and rates is provided in Table A.1. Between batch production, an inactivity period between 5 min and 2 h occurs.

Table A.1: Unit, operations, and phases in the second model including transition trigger conditions, nominal durations, level profiles, and occurence.

Unit	Event Label		Transition Trigger Condition	Nominal Duration	Level Profile	Occurence (per batch)	Procedure Duration
1	Add_Educt1		L1_PV>=30%	5 min	Linear	All batches	∼1.4h (85 min)
	Material_Wait		after (10,min)	10 min	Linear		
	Add_Solids		L1_PV>=60%	20 min	Noise is added to the signal		
	Material_Wait1		after(5,min)	5 min	Linear		
	Product_Transfer		L1_PV<=50%	30 min	Exponential		
	Material_Transfer		L1_PV<=0%	15 min	Linear		
2.1 & 2.2	Add_Educt2	Add_Material	L1_PV>=60%	5 min	Linear	Unit 2.1: For BatchID #1, #2, #4, and so on	∼1h
		Agitation	L1_PV>65%	7 min	Noise is added to the signal		
	Material_Wait		after (15,min)	15 min	Linear		
	Add_Educt3		L1_PV>=70%	3 min	Linear		
	Material_Wait1		after (20,min)	20 min	Linear		
	Product_Transfer		L1_PV<=50%	7 min	Linear	Unit 2.2: For Batch ID	
	Material_Transfer		L1_PV<=0%	5 min	Linear		
2.3	Add_Educt4		L1_PV>=20%	3 min	Linear	#3, and so on For BatchID #5, and so on	47min
	Add_Educt5		L1_PV>=30%	20 min	Exponential		
	Material_Wait		after (5,min)	15 min	Linear		
	Material_Transfer		L1_PV<=0%	5 min	Linear		
	CIP	Spray_Cleaning	L1_PV>=10%	30 min	Noise is added to the signal	20% of occurrence after the end of a batch	∼1.4h
		PumpIn_Recirculation	after(14,min)	14 min	Noise is added to the signal		
		Flush_Tank	L1_PV<=0%	5 min	Noise is added to the signal		
		Secondary_Rinse	L1_PV>=5%	30 min	Noise is added to the signal		
		Flush_Tank1	after(2.5,min)	2.5 min	Noise is added to the signal		

A.2.2 Model Implementation

The implementation of the nominal model and disturbances follow the same steps already discussed for the first benchmark batch process. As such, Section 3.3 is meant to be followed keeping in mind small changes:

- No heating and cooling behaviors are considered, as well, as the behavior of a height sensor;
- For the noise enabler, all parameters were set to the same values with exception of $Post\_Reaction$ since said operation is not discussed for this model's purpose.

For this hybrid system, it is worth mention that the output variables regarding level sensor and batch number differ—these outputs are also exported for each unit. Table A.2 provides an overview of the output variables for this simulation.

Table A.2: Output Variables in the discrete and continuous environments of the simulation for the Benchmark Model 2.0.

Environment	Variable Category	Variable Label	Description	Type of Output Values
Discrete	Time Start/End Points	BatchID	Number of batches produced	Integer
		BatchID_Unit1 BatchID_Unit2_1 BatchID_Unit2_2 BatchID_Unit2_3	ID of the batch being produced for each of the units. Example, when Unit 2.1 is producing a batch, the output will be the ID of said batch.	Integer
		CleaningID	Number of cleaning operations	Integer
		EventFrameLabel	Label of each phase and operation with disturbances scenarios. Provides an output of the current recipe	String
	Valve Positions	Y1_Digital	Valve is only opened on Add_Educt1 for Unit1	Binary
		Y2_Digital	Valve is only opened on Add_Educt2 for both Unit 2.1 and Unit 2.2	Binary
		Y3_Digital	Valve is only opened on Add_Educt3 for both Unit 2.1 and Unit 2.2	Binary
		Y4_Digital	Valve is only opened on Add_Educt4 for Unit 2.3	Binary
		Y5_Digital	Valve is only opened on Add_Educt5 for Unit 2.3	Binary
		Hatch_Digital	Hatch opens on Add_Solids for Unit 1	Binary
Continuous	Sensor Values	L1_PV	Level Sensor. Percentage of how much the vessel if filled	Float (0 to 100)

Table A.2 continued from previous page

Environment	Variable Category	Variable Label	Description	Type of Output Values
		L1_Unit1_PV L1_Unit2_1_PV L1_Unit2_2_PV L1_Unit2_3_PV	As L1_PV but the level profile of each unit	Float (0 to 100)
		N1_PV	Motor Sensor (given in rpm) turned on from Agitation to Product Transfer on both Unit 2.2 and Unit 2.3	Float (0 to 100)

A.2.3 Results of Benchmark Model Simulation Disturbance-Free

A simple campaign is visualized in a Gantt chart notation in Figure A.4. Here, a campaign of eleven batches (color-code) is shown. To be kept in mind: batches 1, 2, and 4 are tracked on units 1 and 2.1, batch 3 has units 1 and 2.2; and, finally, batch 5 will be tracked following units 1 and 2.3.

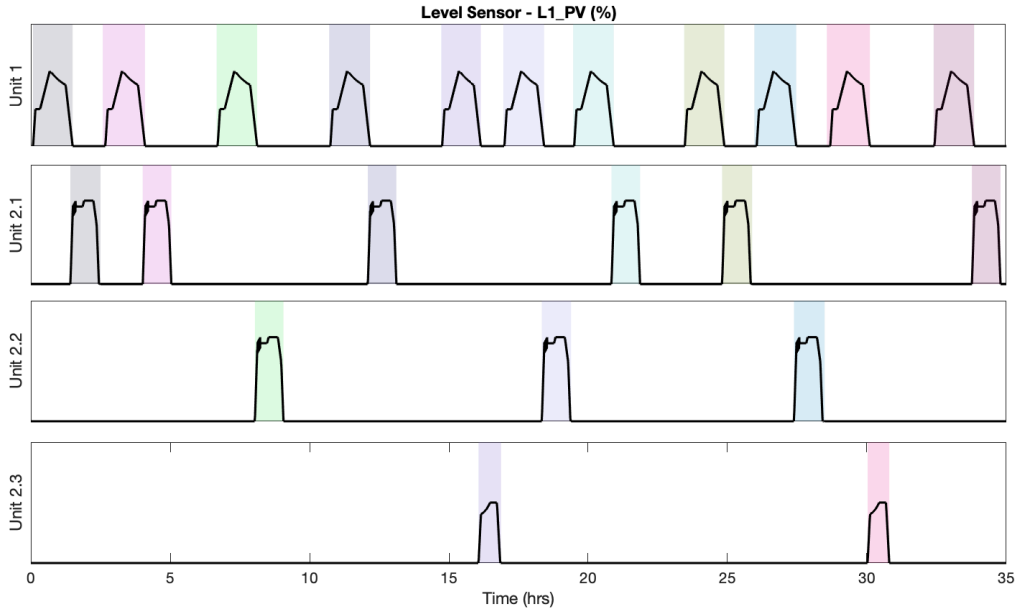


Figure A.4: Batch Tracking for the Benchmark Model 2.0. Each color represents a batch.

Contrary to the first benchmark model, no results regarding the learning of the algorithm are presented. Nonetheless, this implementation covers not only the nominal process but as well the disturbances implementation being provided to the partners in this form.

B

Benchmark Multiple Disturbances Cycles

Table B.1: Benchmark Cycles with multiple disturbances scenarios.

ID/Cycle	A	B	C	D	E
1/2	OP1/OP5/OP7: probability=0.5 OP3: probability=0.3 OP8/OP11: probability=0.4	OP1/OP5: probability=0.8 OP4/OP6/OP10: probability=0.7	OP1/OP7: probability=0.3 OP3/OP4/OP6: probability=0.8 OP5: probability=0.9 OP8/OP11: probability=0.7	OP1/OP10: probability=0.2 OP2: probability=0.7 OP4/OP7/OP8: probability=0.5 OP5: probability=0.9 OP3/OP6/OP9: probability=0.8 OP11: probability=0.3	OP2/OP9: probability=0.7 OP3/OP4/OP6: probability=0.8 OP7/OP8: probability=0.6 OP5: probability=0.9 OP10: probability=0.2
3/4			probability=0.2	probability=0.2	probability=0.2
5	n_batch_campaign=10 variability_campaign=70% 2<variability_end_op<5%	n_batch_campaign=10 variability_campaign=80% 5<variability_end_op<10%	n_batch_campaign=10 variability_campaign=70% 1<variability_end_op<4%	n_batch_campaign=10 variability_campaign=30% 1<variability_end_op<2%	n_batch_campaign=10 variability_campaign=30% 1<variability_end_op<2%
9	OP1: probability=0.3 10<open valve<20sec 20<close valve<30sec	OP1: probability=0.6 5<open valve<10sec 10<close valve<20sec OP5: probability=0.5 1<open valve<2.5min 20<close valve<60sec	OP1: probability=0.7 5<open valve<10sec 10<close valve<15sec OP5: probability=0.7 1<open valve<2.5min 20<close valve<60sec	OP1: probability=0.8 3<open valve<6sec 8<close valve<15sec OP5: probability=0.6 0.8<open valve<2.5min 40<close valve<60sec	OP1: probability=0.8 2<open valve<4sec 6<close valve<30sec OP5: probability=0.7 1<open valve<2.5min 20<close valve<40sec
10	probability=0.5 10<open valve<20sec 20<close valve<30sec	probability=0.7 10<open valve<20sec 20<close valve<30sec		probability=0.5 5<open valve<10sec 10<close valve<15sec	probability=0.5 5<open valve<10sec 10<close valve<15sec
11		sample_timeY5=4h sample_timeY6=2h	sample_timeY5=4h sample_timeY6=2h	sample_timeY5=3h sample_timeY6=5h	sample_timeY5=2h sample_timeY6=5h

Table B.1 continued from previous page

ID/Cycle	A	B	C	D	E
12	Block1: variance=0.1 sample_time=0.05h Block2: variance=0.05 sample_time=0.1h	Block1: variance=0.1 sample_time=0.05h Block2: variance=0.05 sample_time=0.1h	Block1: variance=0.1 sample_time=0.02h Block2: variance=0.05 sample_time=0.1h	Block1: variance=0.1 sample_time=0.01h Block2: variance=0.03 sample_time=0.1h	Block1: variance=0.1 sample_time=0.03h Block2: variance=0.02 sample_time=0.1h
13	OP5: probability=0.5			OP5: probability=0.8 OP9: probability=0.2	OP5: probability=0.3 OP9: probability=0.5
14			X	X	X
15			X		
17			slope=0.1/t_batch	slope=0.01/t_batch	
18		n_batches_reset=5 slope=0.2/t_batch			n_batches_reset=5 slope=0.01/t_batch
19		1<duration<2h seed=3; sample_time=0.5h		1<duration<2h seed=2; sample_time=0.5h	0.2<duration<1h seed=2; sample_time=1h
20			1<duration<8h	5<duration<8h	
20A					8<duration<11h seed=0 sample_time=0.5
20B		1<duration<3h amplitude=0.5; frequency=2		4<duration<8h amplitude=0.05; frequency=1	15<duration<20h amplitude=1; frequency=4
21 (for all cycles)	Block added: sample_time=0.01h -0.05<noise random number<0.05			Block being multiplied: sample_time=0.05h 1-0.001<noise random number<1+0.001	