Comparative study of mine dewatering control systems

S Taljaard

orcid.org 0000-0002-5162-6340

Dissertation submitted in fulfilment of the requirements for the degree *Master of Engineering in Mechanical Engineering* at the North-West University

Supervisor: Dr JF van Rensburg

Graduation May 2018

Student number: 23550589

ABSTRACT

TITLE: Comparative study of mine dewatering control systems

AUTHOR: Stéphan Taljaard

SUPERVISOR: Dr JF van Rensburg

DEGREE: M.Eng. (mechanical engineering)

KEYWORDS: mine dewatering, pump, dam, level, control, control system

Deep-level mines use a series of dams and pumps known as a dewatering system to remove water from underground. The dewatering process can be automated and then controlled on different levels in the control system: programmable logic controllers (PLCs), supervisory control and data acquisition (SCADA), or third-party control software interacting with the SCADA.

Based on personal industry experience, automation engineers often have preferences or bias towards the use of certain levels of control in control systems, but these preferences are not always based on comparative knowledge of all three control systems. Investigation of literature revealed mathematical techniques aimed at optimising pumping schedules and studies using a single control system for pumping optimisation. Experimental studies comparing the three control systems used for dewatering process control were not found.

In order to experimentally compare the control systems, a scoring evaluation method was developed. After verification and validation of the accuracy of mathematical modelling and simulation of the dewatering process, the method is applied to facilitate the comparison. The comparison methodology incorporates the control systems' features and control performance. The features are identified as beneficial to the control of dewatering systems, and includes aspects such as alarm handling capability and a built-in simulation/testing environment. Control performance is assessed from simulation results, and takes into account electrical operating cost, dam levels controlled within limits and frequency of pump starts.

The developed methodology was applied on three case studies provided by deep-level gold mines from a mining group in South Africa. For the case studies, PLC control uses a similar control algorithm to SCADA control, except for the fact that SCADA control considers the downstream dam level. In its state at the time of comparison, this control algorithm did not allow for customisation, whereas third-party control software did.

It was found that control of complex dewatering systems could be better optimised using control systems offering the capability of incorporating more complex/site-specific logic in the control algorithm. Complex dewatering systems require a non-generic approach. For simple dewatering systems, higher complexity control systems are not strictly required, but might offer the advantage of more advanced features which aid in the implementation and use of the control system.

For the mining group providing the case studies, third-party control software showed to be the optimal choice of control system to use. This is followed by SCADA control and then PLC control. Furthermore, it was found that the mines' SCADA control algorithm as developed by the case studies' mine is likely to cause unnecessary starting and stopping of pumps. This leads to increased frequency of pump maintenance and preventable expenditure.

The findings are directly applicable to dewatering systems using the same control algorithms as in the study. The developed methodology can be applied to any dewatering system, or system containing at least one dam and pump.

ACKNOWLEDGEMENTS

I would like to make use of this opportunity to thank the following people or institutions for their contributions towards the success of this study:

- My mentor, Dr Handré Groenewald, for your support, encouragement and assistance in proofreading this dissertation. Thank you for your attention to detail; your critique was always appreciated.
- My supervisor, Dr Johann van Rensburg, for your guidance, feedback, active discussions and words of encouragement.
- Enermanage (Pty) Ltd and its sister companies, for financial support to complete this study.
- My friends and family, your support and encouragement carried me through the completion of this dissertation.
- My colleague and friend, Gerrit Cloete, for your positive attitude, breakfasts and coffee breaks. Thank you for helping me keep my sanity, challenging my ideas, answering (and enduring) my many questions. It has been a valuable learning experience working alongside you.
- My friends, Jaco Mostert and Yvette van Tonder, for your support, advice and always having a listening ear.
- My mother and brother, Marthinette and Marco Taljaard, for your love, continued support, encouragement, and advice and assistance.
- My grandmother, Mara van der Colff, for your assistance in proofreading this dissertation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A	Area	m^2
h	Height	m
L	Dam level	%
L_f	Dam level (fraction)	
M	Mass	kg
p	Pump(s)	
Q	Volumetric flow	m^3/s; L/s
t	Time	seconds
V	Volume	m^3; ML
V_c	Capacity	m^3
ρ	Density	kg/m^3
Δ	Change	
$\dot{}$	Flow of -	

LIST OF ABBREVIATIONS

BPC	Best profile cost
CS1	Case study 1
CS2	Case study 2
CS3	Case study 3
DSM	Demand-side management
I/O	Input/output
M&V	Measurement and verification
OLE	Object Linking and Embedding
OPC	Open Platform Communications (standard)
PLC	Programmable logic controller
REMS	Real-Time Energy Management System
RMSD	Root mean square deviation
SCADA	Supervisory control and data acquisition
SMS	Short message service
SPC	Simulated profile cost
ToU	Time-of-use
UL HL	Downstream (upper-level) dam higher limit
UL LL	Downstream (upper-level) dam lower limit
VSD	Variable speed drive

CHAPTER 1

INTRODUCTION AND BACKGROUND

1. INTRODUCTION AND BACKGROUND

1.1. BACKGROUND

1.1.1. DEWATERING OF MINES

Deep-level mines use water underground for cooling and mining operations such as drilling, cleaning, sweeping, and removal of ore in the form of slurry pumping (Vosloo, 2008:11; Oberholzer, 2014:3–4). Underground, natural water seeping from rock surfaces (known as fissure water) might be present and routed to water holding dams (Vosloo, 2008:11). These dams are also known as “sumps” or “ponds”. To prevent flooding of the mine, the water accumulated underground needs to be removed (Vosloo, 2008:11–12; Oberholzer, 2014:5). The mine’s dewatering system is used to pump water from underground. Water might be used as service water along the way up, or when it reaches the surface, it is filtered (if needed) and cooled for re-use. Cooled water is then used for mine cooling and other water-consuming activities as listed before.

In the dewatering process, a series of reservoirs or dams are used, along with pumps pumping water between these dams (Vosloo, 2008:11–13). This is known as the “pumping system” or “dewatering system” of the mine. Figure 1 (on the next page) illustrates a simplified layout of a dewatering system. A dewatering system is divided into one or more “pumping levels” or “pumping stations”. (Multistage) centrifugal pumps are widely used for water pumping applications (Grundfos, 2004:9; Brito, 2011; Oberholzer, 2014). The pumps are used to pump the accumulated water from the dam(s) on its level to the dam(s) on a higher pumping station. Multi-station dam systems with differing elevations per pumping station are known as a “cascading dam systems”. The dams where the water is pumped from can be arranged in two configurations:

- separate – one or more dams that are not interconnected, and
- interconnected – more than one dam located close to each other, interconnected and equalised so that water volume is shared, effectively forming one larger-sized dam.

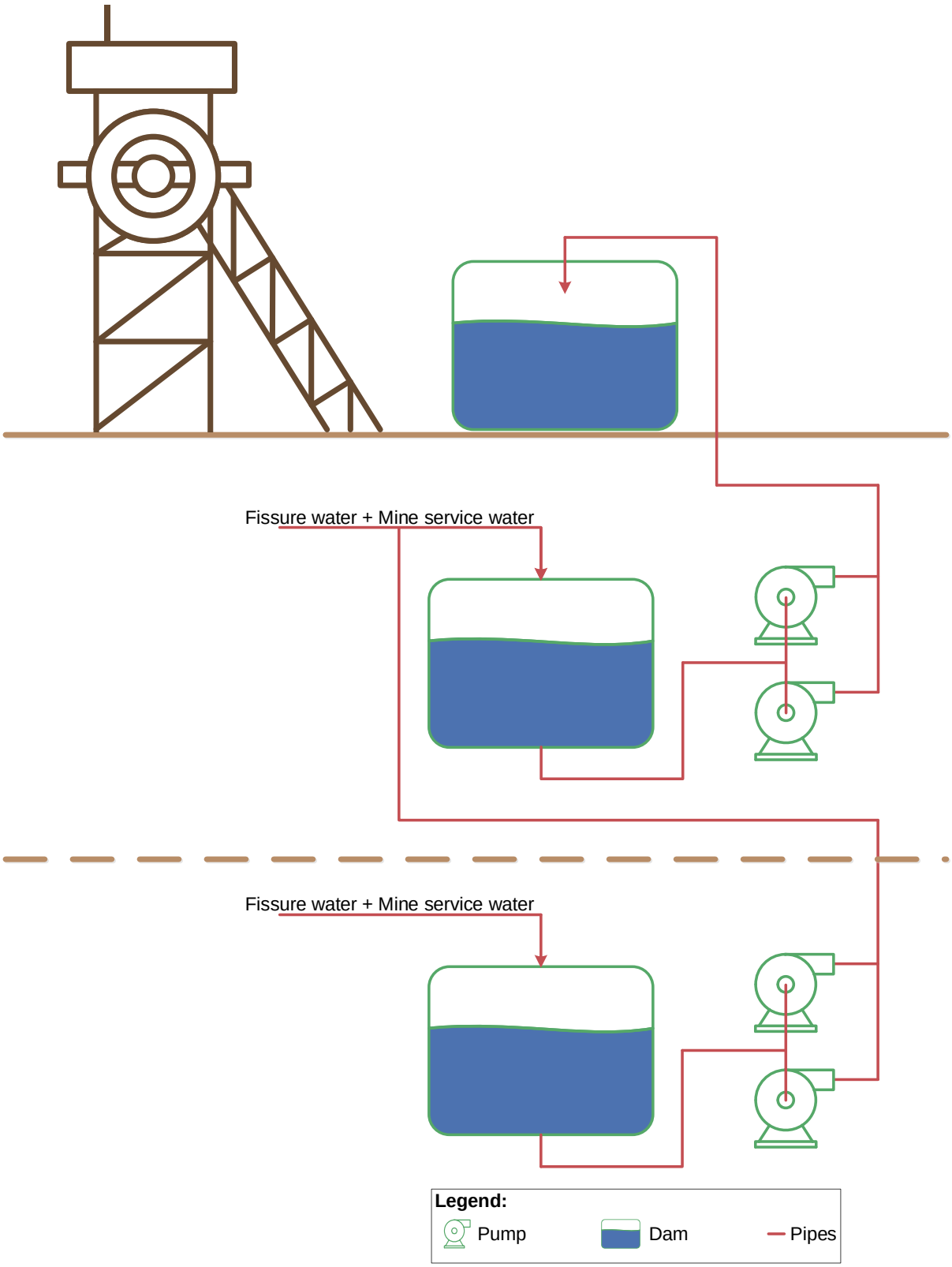


FIGURE 1 – SIMPLIFIED MINE DEWATERING SYSTEM LAYOUT

This process of pumping water from one level to the next is repeated until the water reaches the surface. Figure 2 represents a typical performance curve of a centrifugal pump. This figure illustrates the relationship between the liquid head[1], efficiency, and power consumption of a pump. Particularly, it is noticed that as the head increases (left y-axis), the flow delivered by the pump decreases (x-axis; along the blue head-flow curve) and that the power consumption increases. As flow increases, the efficiency of the pump increases up to the best efficiency point and decreases beyond this point. By making use of multiple cascading dams, the system as a whole is split into separate pumping sections, reducing the total head between the sections. This leads to improvements in the pumping system's efficiency and energy consumption (Oberholzer, 2014:5).

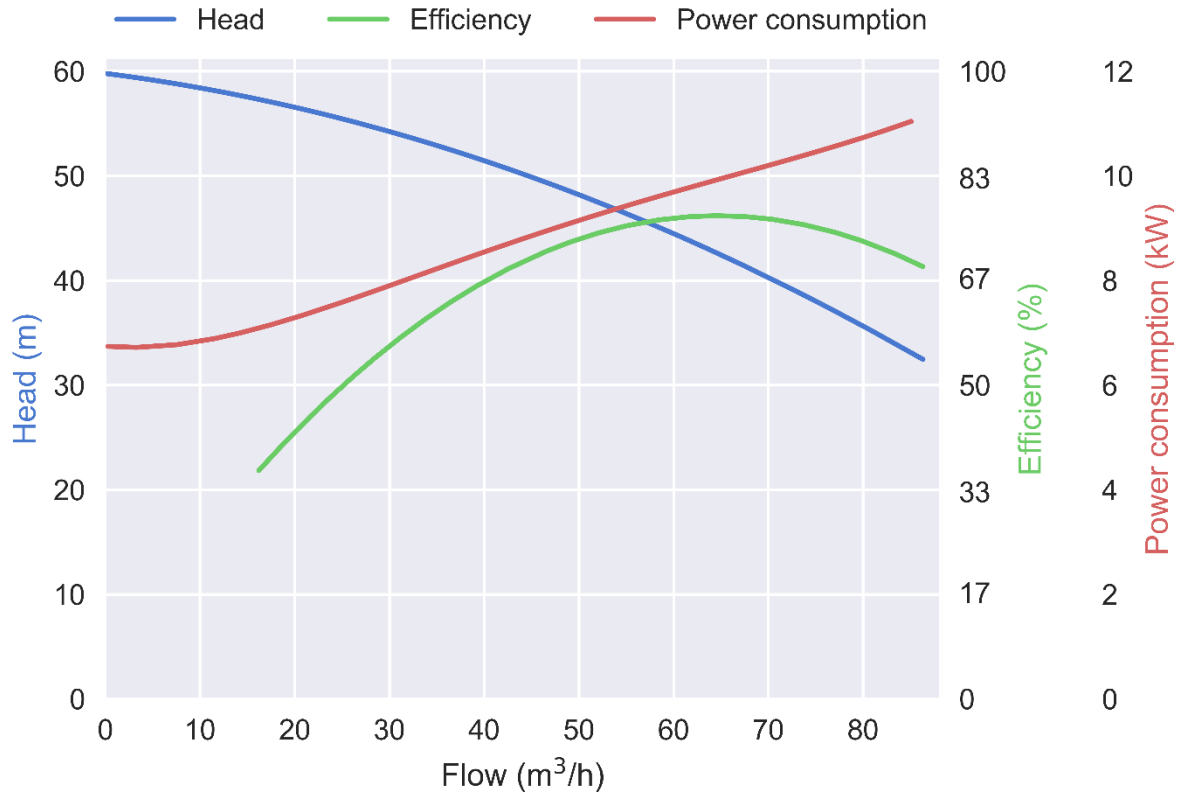


FIGURE 2 – TYPICAL PUMP CURVE
ADAPTED FROM GRUNDFOS INDUSTRY (2004:9).

Mine dewatering systems can be controlled manually by pump attendants at the pumping stations and/or by control room operators if pumps are remotely controllable. This requires the operators to have sufficient training and knowledge of the system, as well as its requirements and constraints. Dewatering systems can also be controlled automatically by computerised

[1] "Head" refers to "the pressure of a liquid expressed as the equivalent height that a column of the liquid would exert" (Schaschke, 2014:176).

systems (known as “automated control”), working towards a predictable manner of controlling dam levels. Automating the control of the dewatering system has advantages, such as safety of equipment and humans in and around the pumps (Oberholzer, 2014:84-90), cost savings through demand-side management (DSM) (Oberholzer, 2014:10), reduced labour and operating costs (Rautenbach, 2007:16) and reduced maintenance because of lessened pump on/off switching (Oberholzer, 2014:12). In times of labour unrest, one can also be assured that the mine dewatering system will run and operate as normal and uninterrupted (Oberholzer, 2014:12).

Typically, automated decision-making and control of pumps are performed by the following control systems:

- **Control via programmable logic controllers (PLCs):**
  Pumps and instrumentation are wired directly to the PLCs. The PLCs control the status of the pumps.
- **Supervisory control and data acquisition (SCADA) control with PLC implementation:**
  Control via the on-site SCADA system that gives start and stop commands to the PLCs.
- **Third-party control software:**
  Control via third-party software that does some externalised decision-making instead of the SCADA.

1.1.2. PLC IMPLEMENTATION

With the PLC implementation, PLCs are not only responsible for switching pumps on or off, but also for deciding to do so according to a schedule or algorithm. One of two approaches is generally followed:

1. **Single PLC:**
   A single PLC per pumping station is used (Figure 3). Instrumentation is wired directly to the PLC. This PLC is responsible for safely switching on/off all the pumps on its pumping station, as well as monitoring/reporting process and equipment variables (sensor readings or “pump condition”) to the SCADA if required. This approach has the risk of losing automatic control capability if the PLC fails, as there is no redundancy in place.

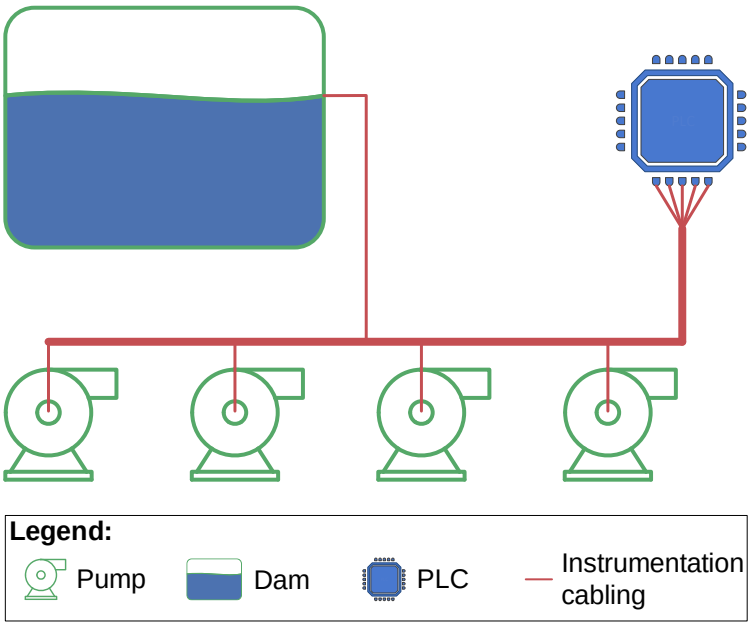


FIGURE 3 – PLC IMPLEMENTATION: SINGLE PLC PER PUMPING STATION

2. **Master PLC:**

   Each pump has its own PLC, which is in turn connected to a master PLC for each pumping station (Figure 4). Each pump PLC is responsible for safely switching its pump on and off, and reporting its pump status to the master PLC. The master PLC serves as a "scheduling PLC": it gives start and stop commands to the pump PLCs based on a pump scheduling algorithm. The pump PLC reports process and equipment variables (sensor readings) to the SCADA if so configured.

   It is possible for the master and pump PLCs to operate independently (albeit without following a schedule). This serves as redundancy in the sense that pumps will still be operable if the master PLC or communication network fails.

When performing automated control of pumping systems, it might be beneficial to use sensor readings from other pumping stations. This is for example, reading a downstream dam level and deciding how many pumps to run upstream from this dam. Although technically possible, it is impractical to access process parameters from another pumping station (Horowitz *et al.*, 2005:2103); hundreds of meters of instrumentation cabling will likely be needed to allow for the wiring between the pumping stations. PLCs also have small amounts of memory which can be problematic if extensive programming is required (Vosloo, 2008).

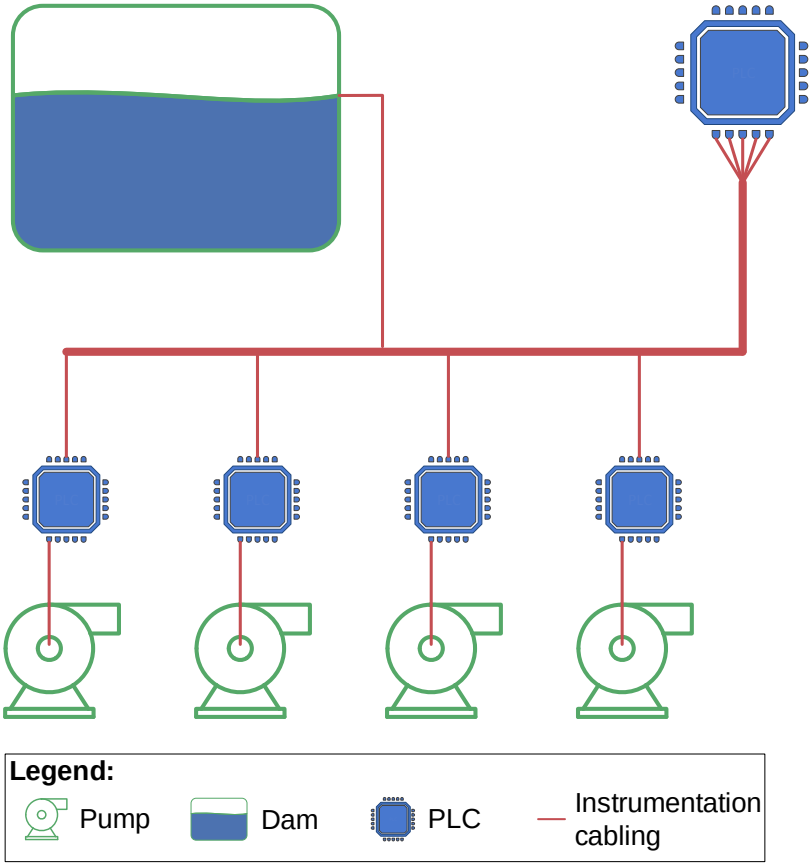


FIGURE 4 – PLC IMPLEMENTATION: MASTER PLC AND PUMP PLCS PER PUMPING STATION

1.1.3. SCADA CONTROL WITH PLC IMPLEMENTATION

With the SCADA implementation, the SCADA issues start/stop commands to the underground PLCs, which in turn start/stop the relevant pumps (Figure 5). SCADAs are generally quite expandable, which allows SCADA developers to add additional logic or functionality to the control system. The SCADA can be configured to include parameters such as time-based schedules for pumps. All data is centralised, making it easier to access most process variables anywhere across the SCADA.

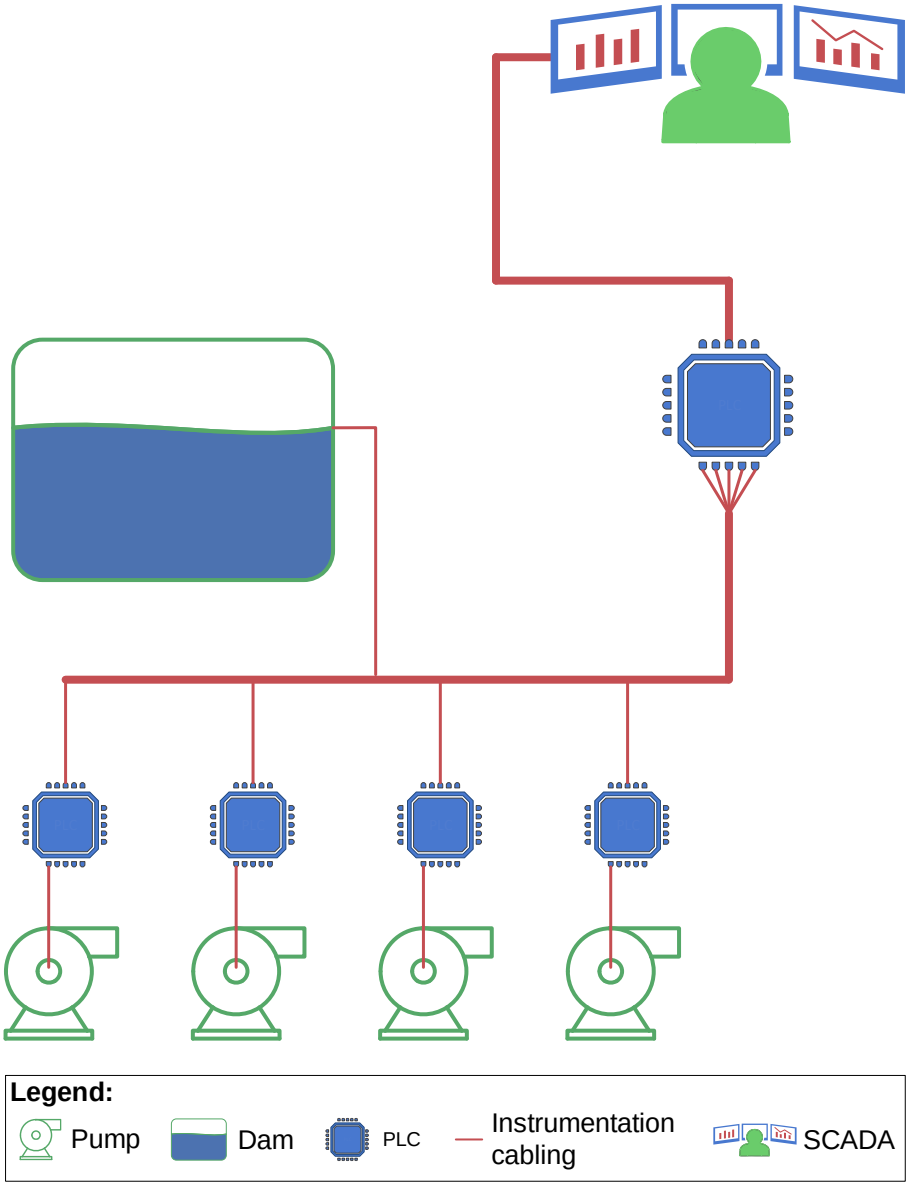


FIGURE 5 – CONTROL VIA SCADA

1.1.4. THIRD-PARTY CONTROL SOFTWARE

The third-party control software implementation is similar to the SCADA implementation, except for the fact that it externalises the decision-making from the SCADA. Third-party control software connects to the SCADA system via a real-time process data connectivity standard, such as the Open Platform Communications standard (OPC) (Figure 6). This allows real-time access to read and write to SCADA “tags” (timestamped data). In essence, this control software can be seen as an “add-on” to the SCADA.

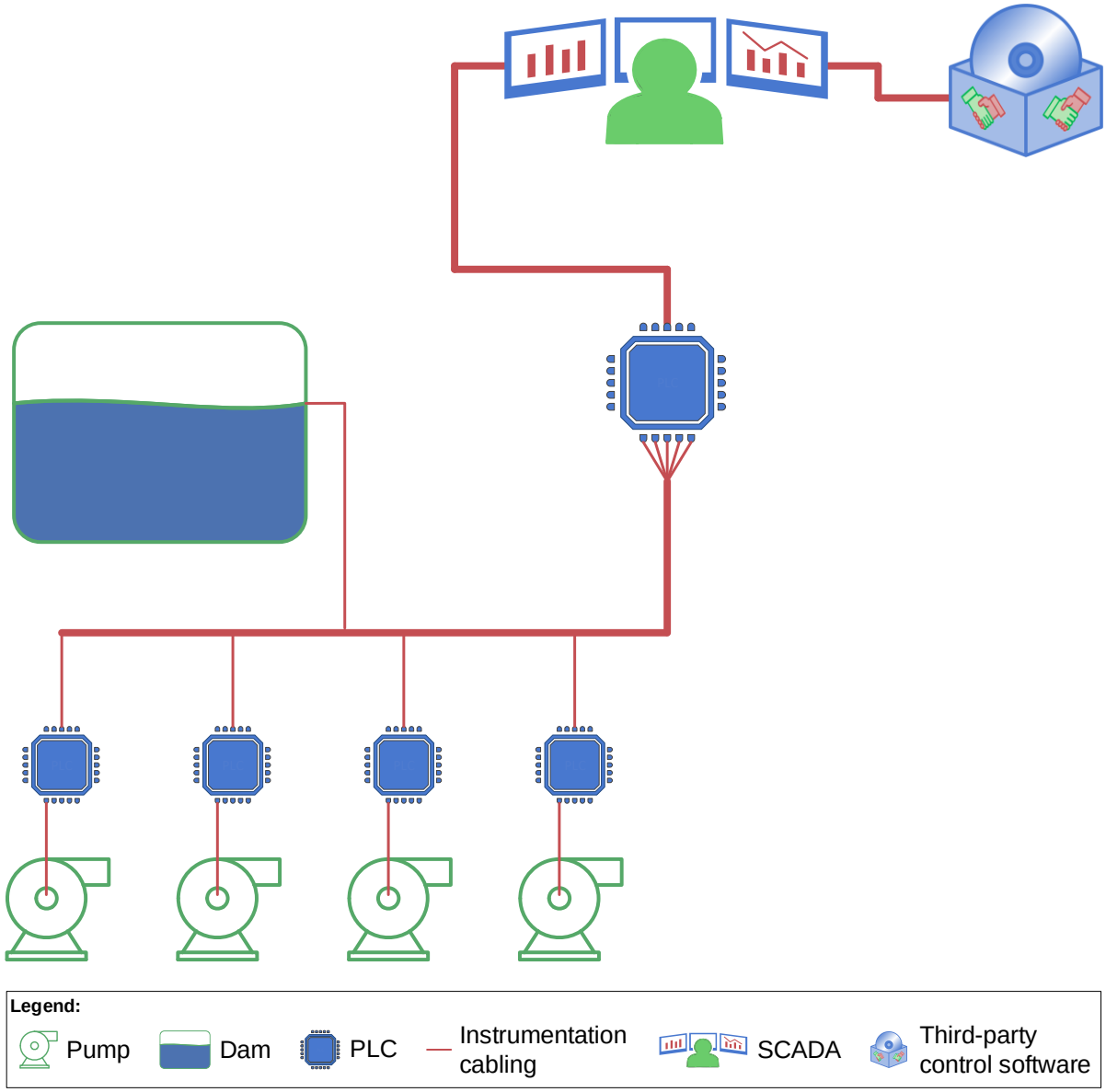


FIGURE 6 – THIRD-PARTY CONTROL SOFTWARE

1.2. PROBLEM STATEMENT

Any of the three control systems discussed in Sections 1.1.2 to 1.1.4 can be implemented for the automated control of mine dewatering pumps. Differences do, however, exist in their capabilities and use. The dewatering system that the control systems will be employed for, along with the needs, requirements and standards that instrumentation/automation engineers may have, will dictate the overall requirements for the control system to use.

Based on personal industry experience, automation engineers are often biased, or have preferences towards the use of certain control systems, but these preferences are not always based on comparative knowledge of all three control systems. Investigation of available

literature reveals numerous studies investigating the optimisation of pump control schedules using mathematical techniques, without focusing on the actual implementation of these techniques[2]. Furthermore, one finds studies doing optimisation using a single control system only[3]. There is no literature available that contain an experimental comparison of the control systems.

A need exists for a thorough evaluation of each of these control systems to be able to determine which will be the optimal choice for the control of dewatering systems of varying complexities. In the evaluation, special emphasis should be given to the practical application of the control systems. A case study-specific, simulation-driven approach is suggested to investigate the workings of these systems and arrive at a conclusion of the ideal approach towards automation to follow.

1.3. SCOPE OF THE INVESTIGATION

It may be possible to develop a control system to perform similarly/on the same level as another. This could be impractical since a considerable amount of resources might be required. Whether the actual system capabilities, development time, skills required, etc., allows for that, will determine the practicality and feasibility of developing the system to such an extent. Because of this, the control systems will be considered as being x-factored, indicating the number of main control variables being considered by the system. This rather changes the focus from the control systems themselves towards a control philosophy dictating how many variables are being considered.

For control system comparisons, only discrete (on/off) control of pumps are considered. Only centrifugal pumps are considered in this study – energy recovery devices such as three-chamber pump systems and U-tubes are not considered. Controlling the rotational speed of pumps by means of electrical control equipment such as variable speed drives is not considered. The long-term effects of implementation of the control systems, availability/ breakdown of pumps and environmental impact (water quality) are not considered. Furthermore, the cost of development, implementation and upkeep of the control systems falls out of scope for this investigation.

---

[2] (Brion & Mays, 1991; Ormsbee *et al.*, 2009; Pasha & Lansey, 2009; Bene, 2013; Hasan *et al.*, 2013; Zhuan & Xia, 2013; Behandish & Wu, 2014; Puleo *et al.*, 2014)

[3] (Pezeshk & Helweg, 1996; Dieu, 2001; Cembrano *et al.*, 2004; Richter, 2008; Shankar, 2008; Aydogmus, 2009; Vosloo *et al.*, 2012; Nortjé, 2012; Oosthuizen, 2012; Van Niekerk, 2013, 2014; Breytenbach, 2014; Smith, 2014; Cilliers, 2014; Grobbelaar, 2014; Oberholzer, 2014; De Jager, 2015; Van der Merwe, 2016)

1.4. AIM AND OBJECTIVES

The aim of the study is to develop and use a method by means of which mine dewatering control systems can be compared.

Three automated pump control systems will be considered: control via PLC, control via SCADA, and third-party software. To limit the scope of each, the control systems will be considered as being x-factored (refer to Section 1.3). The control systems will be evaluated, and the aim achieved, through the following objectives of this study:

1. Compare the three control systems (PLC, SCADA and third-party control software). The comparison will be done by means of:
    a. Evaluation and comparison of control system features that improves ease-of-use and/or are beneficial to the control of the dewatering system:
        i. Simulation/testing environment.
        ii. Optimised pumping schedules.
        iii. Control and automated operation.
        iv. Monitoring and reporting.
        v. Alarm handling.
        vi. Skill level required.
    b. Comparison of the performance of the control systems. Performance of each control system will be by means of simulating the output of the control system and evaluating this with a specific focus on operating cost, dam levels and unnecessary pump cycling that occurs.
2. Draw conclusions as to how the complexity of the dewatering system affects the choice of control system to implement.

1.5. DOCUMENT OUTLINE

CHAPTER 2: LITERATURE REVIEW

In this dissertation, Chapter 2 entails a literature review. Time-of-use electrical tariff structures in South Africa are discussed. This is followed by a discussion of mine dewatering and pump automation. Emphasis is given to control systems used in the automated control of pumps.

CHAPTER 3: DEVELOPMENT OF CONTROL SYSTEM COMPARISONS

Chapter 3 is dedicated to the development of the method for comparing the control systems. This chapter contains the derivation of the mathematical model used for simulating the control

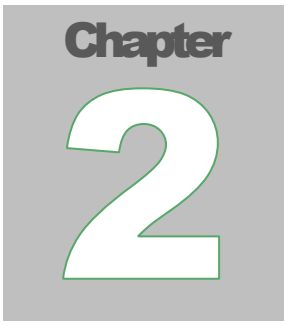
systems and explains the decision-making algorithms of each. The computerised implementation of the mathematical model and simulation is verified. Additionally, in this chapter the validation methodology for ensuring the accuracy and correctness of case study simulations are developed. This chapter lastly describes the methodology by means of which the study itself is validated.

CHAPTER 4: RESULTS AND DISCUSSION

In Chapter 4, the results that are obtained from following the method developed in Chapter 3 are presented. The method is applied on three case studies, and the specific results from each are reported. All results are discussed in this chapter. Each case study's simulation is validated and the study's validity is evaluated further.

CHAPTER 5: CONCLUSION AND RECOMMENDATIONS FOR FURTHER STUDY

Chapter 5 summarises and concludes the study. Key points from the study evaluation are mentioned and recommendations for areas of further research are made.



COMPARATIVE STUDY OF MINE DEWATERING CONTROL SYSTEMS

CHAPTER 2

LITERATURE REVIEW

2. LITERATURE REVIEW

2.1. INTRODUCTION

In this chapter, Eskom's Megaflex tariff structure is explained. This tariff structure is widely used by many of Eskom's industrial clients. The time-of-use-based pricing of this tariff structure is aimed at discouraging clients from using electrical energy supplied by Eskom during peak hours. Background information regarding mine dewatering and control systems is provided. Lastly, the chapter contains an overview and evaluation of previous studies related to the use of PLC, SCADA control with PLC implementation and third-party control software.

2.2. TIME-OF-USE PRICING STRUCTURE IN SOUTH AFRICA

Many industrial energy consumers receiving electrical energy from Eskom use Eskom's Megaflex tariff structure. This tariff structure divides days into "high demand" and "low demand" seasons, and further into peak, standard and off-peak time slots, corresponding to the national electricity demand (Eskom, 2017:2,3,7) (Figure 7). Public holidays are treated as Saturdays or Sundays. Different tariffs are charged based on the season and time slot in effect (Figures 8 and 9). Allocating separate tariffs to time slots, like with the Megaflex tariff structure, is referred to as a time-of-use (ToU) tariff structure. The time slots are known as ToU time slots or ToU periods.

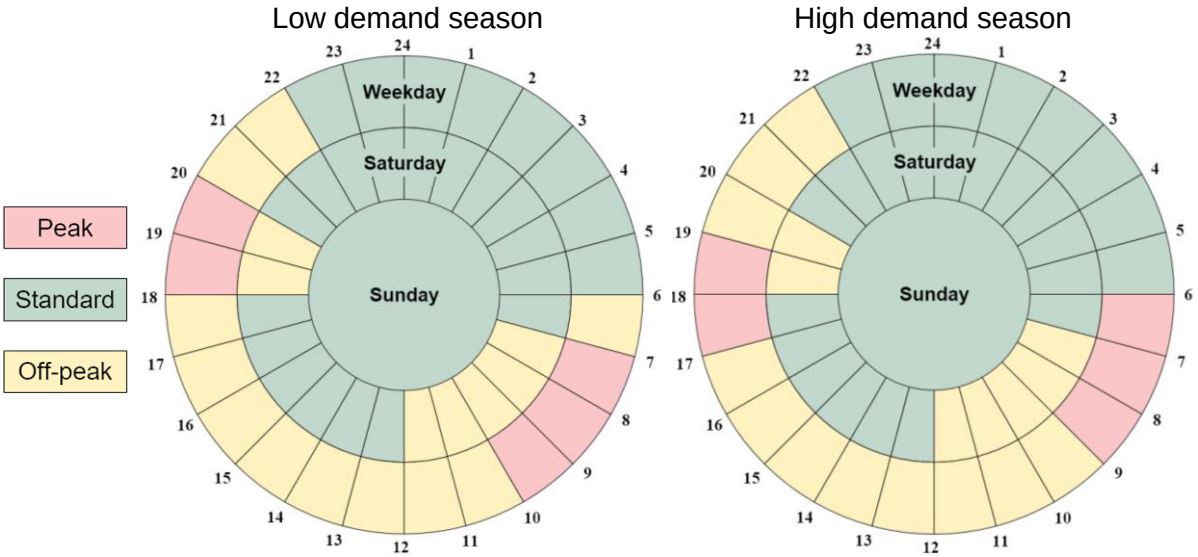


FIGURE 7 – ESKOM MEGAFLEX TIME SLOTS ADAPTED FROM ESKOM (2017:7).

Eskom's Megaflex tariff is further structured to charge customers based on the voltage of their electricity supply and transmission zone[4]. Megaflex tariffs are available in Appendix A (page 95). Figures 8 and 9 reflect the Megaflex weekday tariffs for the transmission zone ≤300 km and a voltage supply of 500 V – 66 kV. The indicated tariff is the active energy cost, which is the c/kWh charge for each unit of energy consumed**.** From Figures 8 and 9, it is clear that there is a considerable difference between the tariffs applicable to the different ToU periods, especially for the high demand season.

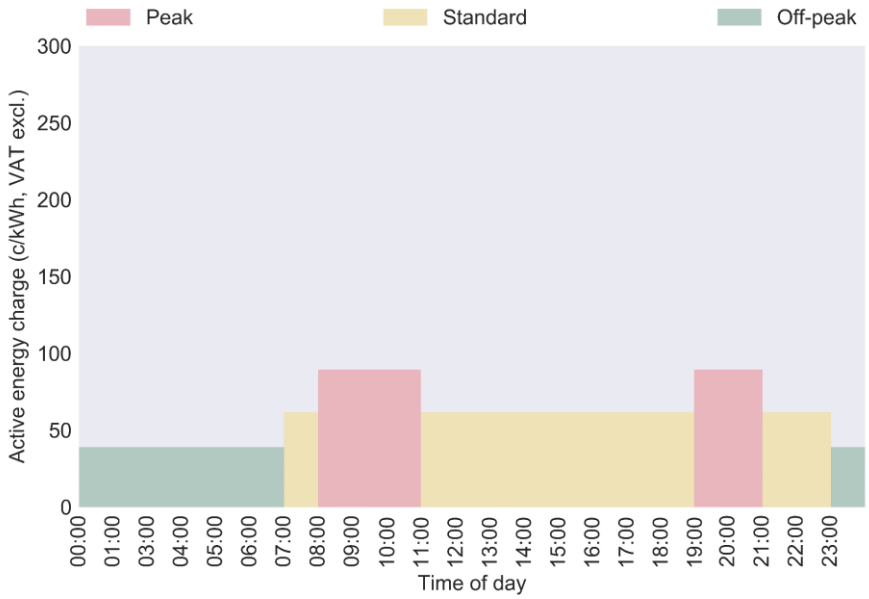


FIGURE 8 – ESKOM MEGAFLEX TARIFF (≤300 KM, 500 V – 66 KV, LOW SEASON)

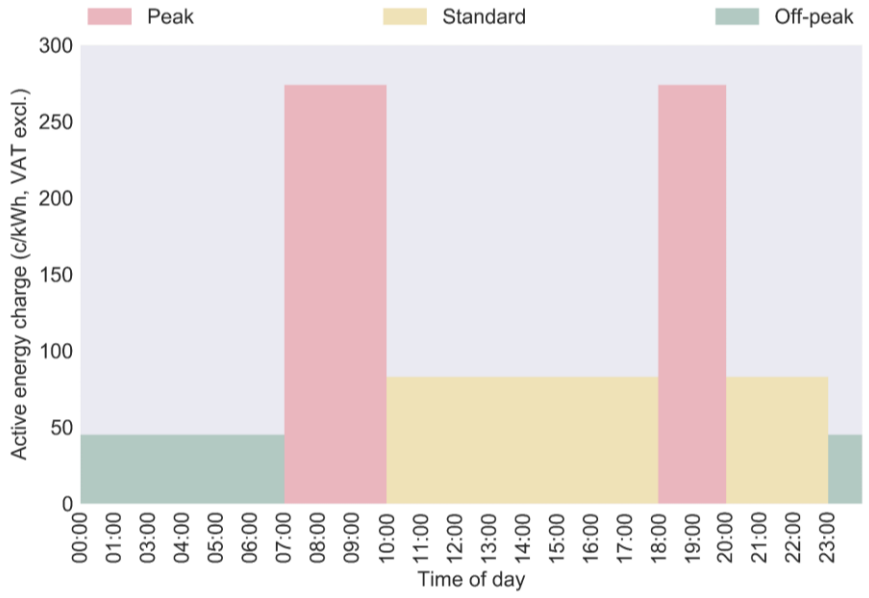


FIGURE 9 – ESKOM MEGAFLEX TARIFF (≤300 KM, 500 V – 66 KV, HIGH SEASON)

[4] Transmission zone: the distance from a central geographical point in South Africa, affecting loss factors and charged accordingly.

Examining the energy tariffs, if possible, one would prefer to use energy in non-peak time slots (i.e. off-peak or standard) instead of in the peak time slots. This is exactly the purpose of load-shift DSM projects. Demand load is shifted from the peak ToU periods, not necessarily leading to energy savings, but electricity cost savings instead. Eskom has this pricing structure in place to encourage (industrial) clients to rather use energy in the non-peak periods, lowering the national maximum demand (that Eskom has to supply).

Because mine dewatering systems contain dams, the storage capacity provided creates the opportunity to perform load-shift on the pumps. If dam levels can be lowered during non-peak periods, pumps can be switched off during peak periods. If dam levels are sufficiently low and the process allows, it might be possible to keep pumps switched off for the entire peak period. Considerable cost savings are realisable by performing load-shift on mine dewatering pumps.

2.3. MINE DEWATERING SYSTEMS

Once water has been used underground, it is often treated before being stored in dewatering dams. These dams are also known as clean, clear, or hot water dams. Underground water treatment consists of a combination of neutralisation, coagulation/flocculation, settling, and disinfection (DWAF, 2008:38–42). Neutralisation is the process of adding lime, soda ash or caustic soda to the water, which chemically neutralises the water's acidity and improves the functioning of flocculants added afterwards. The neutralised pH also has a less corrosive effect on the dewatering pumps.

The addition of coagulants or flocculants enables the settling of suspended solids from the water in underground settlers. Settlers have two outflows: clear water and mud. These flow to separate dams and are removed from the mine using separate systems. Separating the solids from the water prevents mud build-up in the clear water dams and lessens the wear on dewatering pumps' internal components. Clear water is pumped upwards for removal from the mine. In some systems, clear water is pumped to intermediate dams for underground re-use. Mud is pumped out from underground or hauled out in the form of filter press cakes. The mud is ore-loaded, so it is treated in beneficiation plants. Clear water is sometimes treated above or below ground to reduce the health hazard in the case of unintentional ingestion. Chlorine, bromine, chlorine dioxide, calcium hypochlorite or sodium hypochlorite are disinfectants commonly used (DWAF, 2008:89–96).

2.3.1. CLEAR WATER DAMS

Clear water dams often have a large enough capacity to allow storage of a full day's accumulated water (DWAF, 2008:42). Large dam capacities also provide higher suction

pressure for the dewatering pumps. Dam capacities vary in the range from 1 ML to 5 ML. Old mining tunnels or cavities are sometimes converted into water holding dams, as is the case with Case study 2 presented in Section 4.4 and Appendix C.

Clear water dams have minimum and maximum dam levels between which the dams should be controlled. Pumping water from a dam with a dam level lower than the minimum level has the danger of mud (that bypassed the settlers and settled at the bottom of the dewatering dam) entering and damaging the pumps (Cilliers, 2014:28). A lower water level also increases the risk of the occurrence of pump cavitation, leading to decreased pump effectiveness and damage to the pump. Maximum dam levels are usually determined to allow sufficient time to react in case of emergencies (for example pumps tripping when started).

2.3.2. MINE DEWATERING PUMPS

Centrifugal pumps are the most common type of pump encountered in the mining industry (DWAF, 2008:42). Dewatering systems are split to contain multiple pumping stations or dewatering levels to enable water to be removed from the mine. Even with multiple dewatering levels, differences in elevation between two levels are typically high (often as high as 1000 m (DWAF, 2008:48)). For this reason, multistage centrifugal pumps are used. “Multistage” refers to the use of more than one impeller in the pump, allowing the pump to impart more energy (pressure) to the water (increasing the distance water can be pumped).

Pumps have two main modes of control: on/off (or discrete) control, and modulating control (Horowitz *et al.*, 2005:2101–2106). Discrete control entails switching a pump on or off when a dam level reaches a certain value. Pumps controlled in this manner are referred to as constant-speed pumps. Modulating control consists of altering the effective flow delivered by a pump. This is by means of

- bypassing (returning) some of the water to before the pump,
- throttling the output of the pump by using a throttling valve, or
- adjusting pump rotation speed by using variable speed drives (VSDs).

Between the modulating control options, VSD control is preferred, as this method is more energy-efficient.

A “cycle” of a pump refers to the start and stop of a pump. Some applications may cause a pump to start and stop repeatedly in a short time period. This may cause damage to the pump, motor, pipes or other fittings (Grundfos, 2014). This damage is partly caused by the higher than usual electrical current required by the pump motor during start-up, leading to higher than normal motor temperatures (Horowitz *et al.*, 2005:2101). Exactly what constitutes “a short time period” was not found in literature. Horowitz *et al.* (2005:2101) state that large,

non-submersible pumps usually have a maximum number of starts per hour of two to four. From personal industry experience, dewatering pump operators prefer to not cycle a pump within a time span of fewer than 1-1.5 hours. Puleo *et al.* (2014) and Cembrano *et al.* (2000) also did not cycle pumps within time spans of less than an hour.

Within the context of gold mining in South Africa, the term “cycling” of a pump is used more to refer to the starting and stopping of a pump *in quick succession*, rather than *simply* the starting and stopping of a pump.

With discrete pump control, special attention should be given to the cycling of pumps, especially because increased pump cycling leads to increased maintenance costs (Lansey & Awumah, 1994).

2.4. CONTROL SYSTEMS

2.4.1. PROGRAMMABLE LOGIC CONTROLLER

A programmable logic controller (PLC) is a small computer optimised for control of machines and processes and use in the industrial environment (Mehta & Reddy, 2014:37–50). PLCs contain multiple built-in or modular input and output ports (I/O ports), to which sensors and final control elements can be connected. PLCs read a voltage or current signal from input devices and interpret these inputs according to a program written in the PLCs. Examples of input devices are sensors such as switches, temperature sensors, pressure sensors and level sensors. PLCs output information/instructions to external devices. Examples of output devices are relays, solenoid-operated valves and direct current motors. Using the PLC’s communication interface, data is received or transmitted to other PLCs/SCADA.

2.4.2. SUPERVISORY CONTROL AND DATA ACQUISITION

Supervisory control and data acquisition (SCADA) systems are used for monitoring and remote control of industrial processes (Mehta & Reddy, 2014:237). Using a SCADA system, operators are able to make set point adjustments, turn control devices on or off, access process alarms and gather process data over any distance.

A SCADA system consists of a SCADA server (or master terminal unit, MTU) at its centre (Figure 10). An operator interacts with the MTU by using a screen and computer input devices, such as a keyboard and mouse. The MTU interacts with control hardware known as remote terminal units (RTUs) located some distance away from the MTU. RTUs are connected to the MTU by means of wired and/or wireless communication technology. PLCs are examples of

RTUs – controllers installed at or near field devices for interaction and control with them. Field devices, such as sensors and valves, are connected to and interacted with the RTU by means of wires.

As part of the SCADA system, the historian stores timestamped data (tags) for later access of historical data (in the form of graphical trends or database queries) (Bagri *et al.*, 2014).

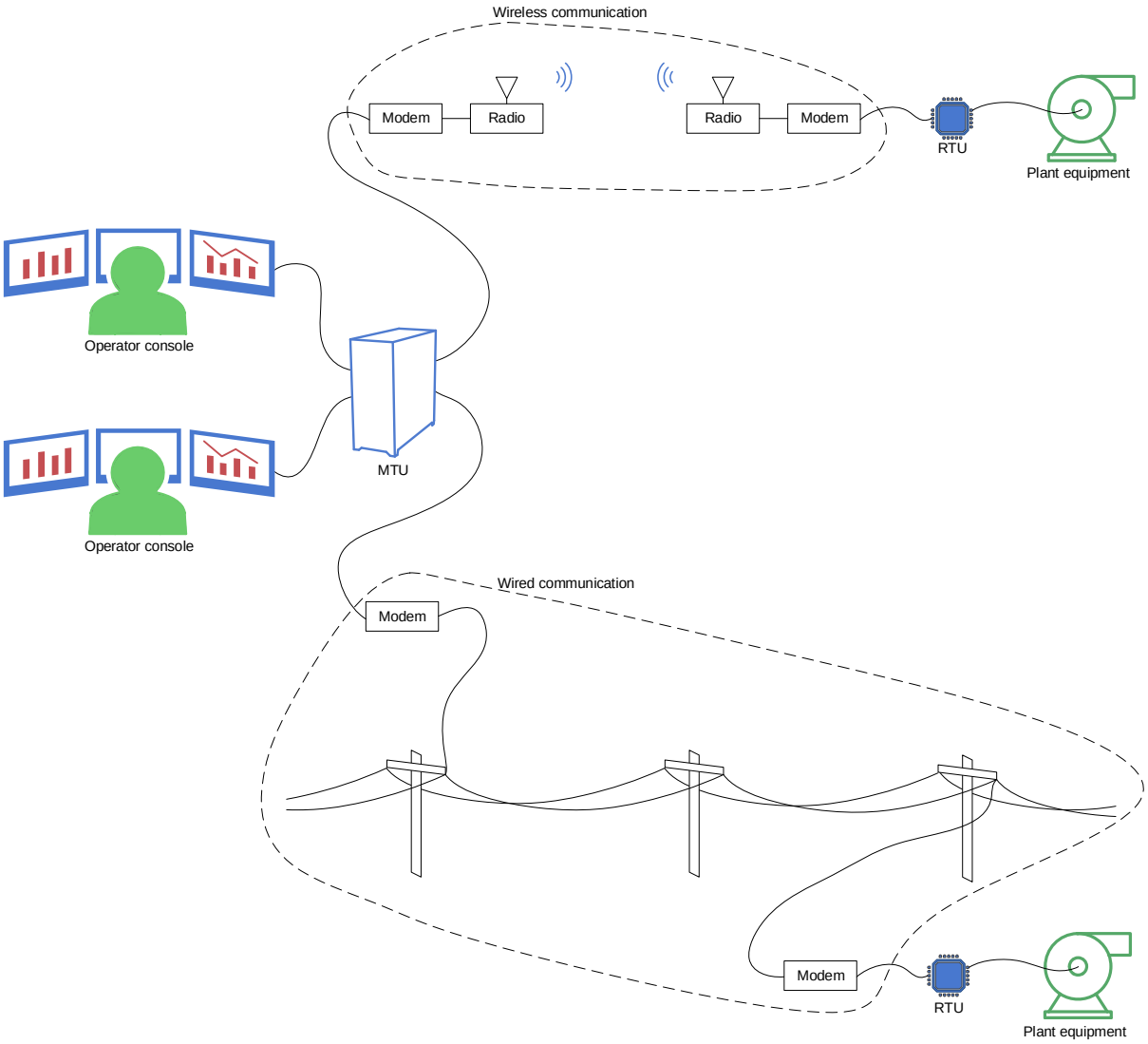


FIGURE 10 – SCADA SYSTEM LAYOUT
ADAPTED FROM Boyer (2004:13–14) AND Mehta & Reddy (2014:238)

2.4.3. THIRD-PARTY CONTROL SOFTWARE

The Open Platform Communications (formerly known as Object Linking and Embedding (OLE) for process control) (OPC) standard was developed to enable interoperability between automation providers and devices, such as PLCs from different vendors and process control software running on different operating systems (Mehta & Reddy, 2014:459). OPC enables

focus on the *use* of devices or software instead of having to focus on the *communication between* these (Mehta & Reddy, 2014:460) (Figure 11).

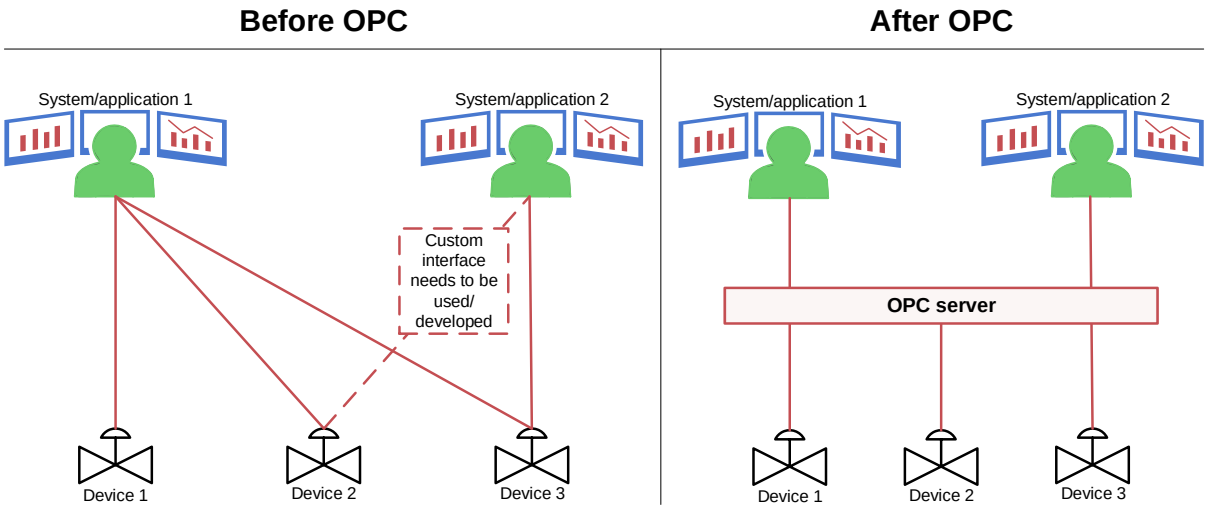


FIGURE 11 – OPC ALLOWS INTEROPERABILITY BETWEEN APPLICATIONS ADAPTED FROM MEHTA & REDDY (2014:460–461)

2.4.4. CONTROL SYSTEM FEATURE REQUIREMENTS

The following control system features have been identified to be beneficial when controlling dewatering systems. These features were gathered from personal industry experience and from the work done by Boyer (2002:364–365), Mehta & Reddy (2014:286–300) and Rautenbach (2007).

SIMULATION/TESTING ENVIRONMENT

This feature refers to the built-in capability to test the control of the water pumping system. This enables easy testing of the control philosophy. A simulation or testing environment can aid in preventing the deployment and use of incorrect control parameters, which could lead to incorrect control of the system, such as dams overflowing or pumps running dry.

OPTIMISED PUMPING SCHEDULE

This feature refers to the capability to calculate and/or perform optimised control of pumps based on an optimised pumping schedule. This schedule should be such that it reduces the running cost of the system (in terms of electricity and maintenance cost).

CONTROL AND AUTOMATED OPERATION

The control system should be able to control the components of the pumping system in one way or another. Control tasks should be completed without needing 24-hour human assistance. Emergencies should not cause failure of the control system.

MONITORING AND REPORTING

This refers to the capability of the control system to provide monitoring capability for operators to be able to observe and/or control the process. Furthermore, this feature refers to the capability of automatic logging, managing and reporting of data relevant to the system being controlled.

ALARM HANDLING

This refers to the control system's ability to raise warnings in the event that process parameters reach certain limits. These alarms attract the operator's attention to elicit rapid response to keep the process within control.

SKILL LEVEL REQUIRED

In order to set up the control system, programming of some sort is required. This "feature" refers to the skill level required by the person doing setup or making maintenance changes on the control system.

2.5. PREVIOUS STUDIES

Many studies were investigated for possible relation to this study. Table 1 lists the critical evaluation of related studies investigating pumping systems or pump schedule optimisation, and/or the use of PLC/SCADA/third-party control software. In the context of the current study, these refer to automatic control using PLCs, SCADA, and third-party control software interacting with a SCADA, respectively.

Table 1 also lists the control systems used/investigated (experimentally) in each study. In addition, the table lists whether control systems were compared in each study, and whether a well-argued choice of control system was made. As described before, many studies focussed only on the theoretical optimisation of pumping schedules. These studies, along with studies making use of simulations, were identified. The table also identifies studies wherein control using the considered control system was actually implemented or if it is implementable.

TABLE 1 – CRITICAL LITERATURE EVALUATION

	Study	Control system					
		PLC	SCADA	Third-party	Reason/comparison	Theoretical	Implementation
A	Van der Merwe (2016)			●		●	●
B	Els (2015)			●		●	●
C	De Jager (2015)			●		●	●
D	Cilliers (2014)			●		●	●
E	Breytenbach (2014)			●		●	●
F	Grobbelaar (2014)			●			●
G	Oberholzer (2014)			●		●	●
H	Smith (2014)			●		●	
I	Van Niekerk (2014)			●		●	●
J	Van Niekerk (2013)			●	●[5]	●	●
K	Nortjé (2012)			●		●	●
L	Oosthuizen (2012)			●		●	●
M	Vosloo *et al.* (2012)			●		●	●
N	Botha (2010)			●			●
O	Aydogmus (2009)		●				
P	Richter (2008)			●		●	●
Q	Shankar (2008)		●		●[6]		●
R	Vosloo (2008)			●	●		●
S	Cembrano *et al.* (2004)			●			●
T	Dieu (2001)		●				●

[5] Used Real-Time Energy Management System (REMS) as alternative to PLC or SCADA control for reasons listed by Vosloo (2008).
[6] Comparison/reasons for implementing versus manual control.

U	Cembrano *et al.* (2000)			●			
V	Pezeshk & Helweg (1996)		●			●	
	The following studies focused only on the theoretical optimisation of pump schedules: • Behandish & Wu (2014) • Puleo *et al.* (2014) • Bene (2013) • Hasan *et al.* (2013) • Zhuan & Xia (2013) • Ormsbee *et al.* (2009) • Pasha & Lansey (2009) • Brion & Mays (1991)					●	

From Table 1 it is clear that numerous published studies are available using third-party control software, almost exclusively in the South African environment. None of these studies mentioned why third-party control software was the chosen method for implementation, especially compared to PLC or SCADA control. The studies investigating manual versus automatic control did mention the reasons, as that was the purpose of those studies.

Examination of available literature did not reveal any comparative studies investigating or listing differences between the control systems PLC, SCADA and third-party control software based on experimental results.

It was found that previous studies aimed at optimisation of pumping systems (be it mine dewatering or national water distribution networks) mainly focussed on schedule and running cost optimisation. Many of these studies investigated theoretical approaches towards optimisation, often using advanced mathematical techniques. These studies are considered to be “theoretical” because they go into detail about the mathematical model and solving/decision-making algorithms used for optimisation; these being simulated but not implemented. Many of these optimisation methods were found to be computationally expensive and require knowledge of future process parameters, making them unfit for real-time process control.

Furthermore, most studies use either PLC, SCADA or third-party control systems, but specific and clear reasons for this choice are not listed. In the context of this study, reasons for using one automated control system *instead of another* is especially sought after. Some studies investigating manual versus automatic control did list differences and justification of using automated control, but the listed differences are irrelevant to this study, which investigates *types of automatic control*. In the context of this study, those investigations refer to automated

control more as a generic type of automated control (i.e. it can be PLC, SCADA or third-party). Table 2 lists a description or summary of the studies in Table 1 and their findings.

TABLE 2 – BRIEF DESCRIPTION OF EVALUATED LITERATURE

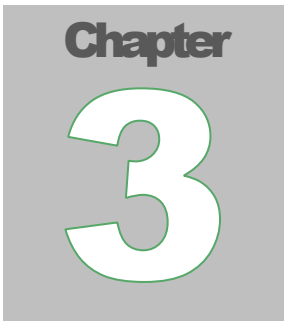
	Study and summary/applicability to this study
A	Van der Merwe (2016) used case studies provided by projects implemented in 2008, using third-party control software to optimise dewatering system control. The software did not automatically control the pumps but recommended operating schedules, which were manually followed.
B	Els (2015) focussed on shifting energy load to non-peak hours of a water distribution network's pumps. He provided a brief overview of industrial control systems. Advantages were given of a specific industrial energy management control software, but not in the context of comparison to PLC or SCADA control. The water distribution network was simulated by this software and the software was then used to generate pump schedule recommendations.
C	De Jager (2015) used third-party control software to control a mine dewatering system. The study mostly focussed on the effect of pump availability and briefly mentioned control implementations.
D	Breytenbach (2014) used third-party control software for automated control of a water distribution system.
E	Cilliers (2014) used third-party control software for optimised load-shift on pumping systems. In this study, however, dynamic control ranges were used instead of conventionally constant control ranges.
F	Grobbelaar (2014) mentioned control modes: manual, PLC and SCADA. Grobbelaar explained the working of a PLC and SCADA and listed advantages of control using SCADA. This dissertation focuses on maintenance and maintenance procedures of DSM projects, not the actual implementation thereof. Grobbelaar made use of third-party control software but did not list any reasons for this choice.
G	Oberholzer (2014) investigated best practices towards pump automation, with special focus on the practical aspects of its implementation (topics such as instrumentation, interlocks, start-up, shutdown and trip procedures). Oberholzer did not focus on the control system being used. He listed advantages of using an automated system compared to manual control.

H	Smith (2014) mentioned the advantages of using automated control instead of manual control. Smith investigated and mentioned the benefits of automation (in general); parties whose involvement are needed to ensure the success of an automation project; factors influencing the success of a pump automation project; instrumentation needed for automatic control; and start-up, shut-down and trip conditions. Smith concluded that automatic control ensures operation and efficiency of the pumping system (compared to manual, manual scheduled, and manual surface control).
I	Van Niekerk (2014) used third-party control software for performing automated load-shifting on pumping systems. He provided a brief explanation as to how PLC, SCADA and third-party control works.
J	Van Niekerk (2013) simulated compressed air systems and dewatering systems. Simulations were used to improve DSM project performance and to rectify problems that were encountered during the implementation of existing mine DSM projects. He cited Vosloo (2008), who stated that Real-Time Energy Management System (REMS) is an alternative and more feature-rich control and simulation option than using PLC or SCADA control.
K	Nortjé (2012) focused on the implementation of a third-party control system on a national water distribution system. Nortjé discussed details regarding, PLC, SCADA and third-party control implementations for illustrative reasons.
L	Oosthuizen (2012) compared automatic control of dewatering systems with manual control. He found that automatic control is the optimal method to use.
M	Vosloo *et al.* (2012) used third-party control software for dewatering control on a gold mine. This control leads to a 65% demand reduction in peak and a 13% saving on annual operational cost.
N	Botha (2010) uses third-party control software for water supply optimisation at a gold mine, but without stating reasons for this choice (especially compared to using PLC or SCADA control).
O	Aydogmus (2009) used fuzzy-logic-based level control using SCADA (via PLC). SCADA was used for controlling and monitoring the operation of the process and provides access to all inputs and outputs. Control was by using the SCADA to change set points or to switch a pump on or off.

P	Richter (2008) compared manual control with automatic control. He listed advantages and disadvantages of both, and then continued to use third-party control as his implementation of “automatic control” without listing reasons for this specific choice.
Q	Shankar (2008) automated boiler operation using SCADA control. He explained how a PLC works. Shankar also explained advantages and disadvantages of manual control, PLC control and SCADA control. In his justification of choice of the control system, automatic (or PLC), control was compared to manual control.
R	Vosloo (2008) mentions control via PLC and SCADA. Vosloo found that SCADA systems are incapable of controlling integrated mine water networks. He found that SCADAs do not have simulation capabilities and are unable to perform complex calculations and optimisation, leading to problems such as pump cycling or incorrect control of dam levels. Vosloo found that PLCs have little memory and no database capability, meaning that performing simulations and complex models/calculations are not possible.
S	Cembrano *et al.* (2004) used third-party software running alongside the SCADA. This software was used for real-time optimisation and control on urban drainage systems.
T	Dieu (2001) successfully used SCADA for wastewater treatment plant control.
U	Cembrano *et al.* (2000) used third-party software running alongside the SCADA for control and optimisation of a water distribution network. It was also found that automated control performs better than manual control.
V	Pezeshk & Helweg (1996) used third-party software KYPIPE for real-time optimisation (scheduling of pumps).
	The following studies focused only on the theoretical optimisation of pump schedules: • Behandish & Wu (2014) • Puleo *et al.* (2014) • Bene (2013) • Hasan *et al.* (2013) • Zhuan & Xia (2013) • Ormsbee *et al.* (2009) • (Felder *et al.*, 2015:94, 571) • (Kobayashi & Salam, 2000; Gauch *et al.*, 2003; Piñeiro *et al.*, 2008; Pedregosa *et al.*, 2011)

2.6. CONCLUSION

Eskom's Megaflex tariff structure was explained in this chapter, stating that time-of-use tariff structures like Megaflex are utilised to drive load-shift initiatives. An overview of previous studies related to the use of PLC, SCADA and third-party control software was also provided and analysed to determine their applicability towards this study. It was found that many previous studies focussed only on the theoretical optimisation of pumping systems or the use of third-party control software without explaining why this was the choice compared to PLC and SCADA control.



CHAPTER 3

DEVELOPMENT OF CONTROL SYSTEM COMPARISONS

3. DEVELOPMENT OF CONTROL SYSTEM COMPARISONS

3.1. INTRODUCTION

In this chapter, a methodology is developed that will facilitate the comparison of different control systems for mine dewatering. The methodology consists of two parts:

- comparison of the performance of the control systems by means of simulations,
- and comparison of the features of the control systems.

3.2. METHODOLOGY OVERVIEW

In order to answer the question as to which dewatering pump control system is the optimal choice for different applications, comparison of the control systems is required. For this comparison, a multi-pronged approach is suggested, consisting of the following steps:

1. **Pumping system identification.**
   Multiple dewatering systems are identified for investigation. These systems should differ in complexity to be able to investigate each control system's behaviour under different circumstances.
2. **Develop a base case simulation and test its integrity.**
   Each dewatering system is simulated without allowing the control system to interact with and control the system. This will allow testing the accuracy of the model, simulation and assumptions.
3. **Perform control system simulations.**
   The interaction and control of each control system are simulated as working on each dewatering system. The simulations demonstrate how the control systems will attempt to control the pumps.
4. **Scoring.**
   Simulation results are evaluated and a score is given to the performance of the control system. Control system features are also scored for comparison purposes.
5. **Score combination and evaluation.**
   Performance and feature scores are be combined to make up a total score for each control system for each dewatering system. From there, conclusions can be made as to which control system works best for different levels of dewatering system complexity.

3.3. PUMPING SYSTEM IDENTIFICATION: SITE SURVEY

The first step in the developed investigation process is to identify a viable dewatering/pumping system ("the site") to be investigated. The following requirements or initial screening criteria will help to identify a viable candidate system:

- The pumping system should have at least one dam where water is being pumped from.
- At least one pump should be present, pumping water from the dam(s) in the criterion above.
- Historical data or a means to collect new data should be present.

After successful initial identification and selection of the site, data should be collected to be able to simulate the system. The following questions should be answered by the data collected:

- How many dewatering levels does the system have?
- How many dams are located on each level?
- What are the capacities of these dams?
- Are the dams interconnected?
- What are the fissure/additional water inflows into the dams?
- What are the dam level limits (minimum and maximum) per dam?
- What happens when a dam exceeds its level limits (i.e. overflows or empties)? How severe are these occurrences; does it warrant penalisation? Which dam level values warrant penalisation?
- How many pumps are installed per pumping station?
- What is the flow and power consumption per pump?
- How many pumps can run at a given moment (considering electrical and mechanical capacity)?
- Which Eskom tariff structure is applicable to the site?

3.4. MODEL AND SIMULATION DEVELOPMENT

3.4.1. MODELS

In order to simulate each system, a mathematical model for the dam level is developed:

GENERAL MASS BALANCE

The first step is to model the accumulation or change of water level in a dam located on a pumping station. This is done by means a water mass balance. Drawing a system boundary

across each dewatering level, a water mass balance for each level is as follows (Felder *et al.*, 2015:94,571):

$$\text{Accumulation} = \text{input} - \text{output} + \text{generation} - \text{consumption} \quad (1)$$

Because pumping systems are non-reactive processes, the generation and consumption terms can be dropped:

$$\text{Accumulation} = \text{input} - \text{output} \quad (2)$$

Considering the process as transient, this equation becomes:

$$\frac{dM}{dt} = \dot{m}_{\text{in}} - \dot{m}_{\text{out}} \quad (3)$$

where $\frac{dM}{dt}$ represents a change of mass of water contained within the dam with respect to time, and $\dot{m}_{\text{in}}$ and $\dot{m}_{\text{out}}$ denote mass flows of water into and out of the system (in units of mass per time), respectively. For simulation purposes, multiple interconnected dams per dewatering level are viewed as one dam with a larger volume (Figure 12):

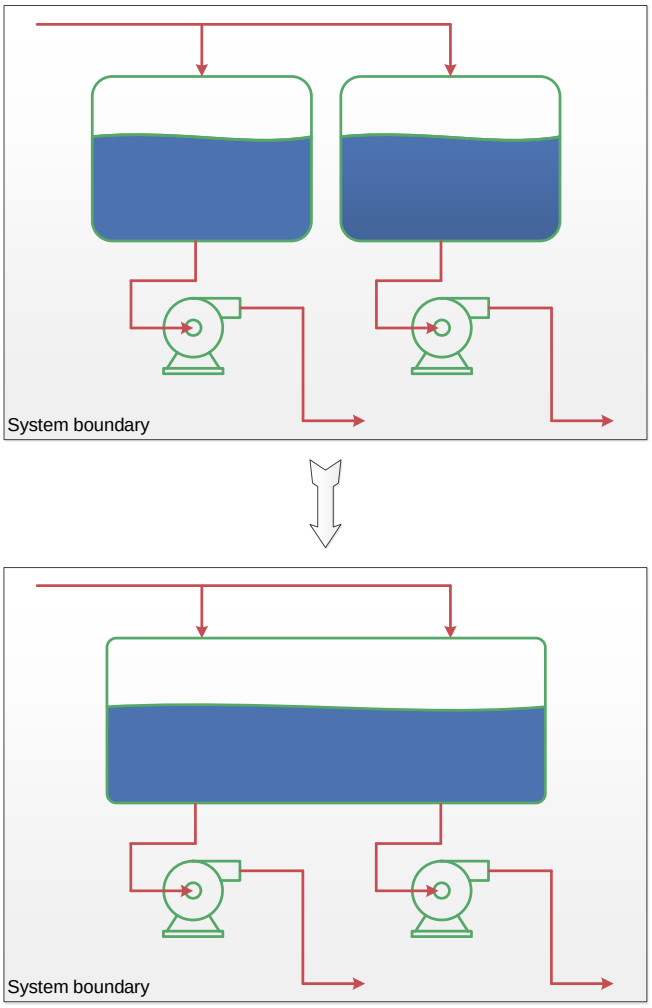


FIGURE 12 – DEWATERING LEVEL SYSTEM BOUNDARY

Expanding the left-hand side of Equation (3) gives

$$\frac{dM}{dt} = \frac{d(\rho V)}{dt} \tag{4}$$

where V denotes the volume of water in each dam and ρ the density of water. The volume can be expressed as either a constant dam area (A) multiplied by a varying height (h), or as a percentage or fraction (defined hereafter as "dam level", L and L_f, respectively) multiplied by the dam capacity (denoted hereafter as V_c):

$$\frac{dM}{dt} = \frac{d(\rho V)}{dt} = \frac{d(\rho A h)}{dt} = \frac{d\left(\rho L_f V_c\right)}{dt} \tag{5}$$

The latter option used in this study, since generally, dam capacities are known and dam cross-sectional areas are not known (especially since old haulages or inclines are often renovated into dams, leading to cross-sectional areas differing along the height of the dam).

Expanding the right-hand side of Equation (3) gives

$$\dot{m}_{\text{in}} - \dot{m}_{\text{out}} = \rho_{\text{in}} Q_{\text{in}} - \rho_{\text{out}} Q_{\text{out}} \tag{6}$$

with Q_{in} and Q_{out} representing the volumetric flow of water into and out of the system. ρ, again, represents the density of water, with the subscript indicating whether it applies to the water flowing into or out of the dam.

Thus, Equation (3) can be written as

$$\frac{d\left(\rho L_f V_c\right)}{dt} = \rho_{\text{in}} Q_{\text{in}} - \rho_{\text{out}} Q_{\text{out}} \tag{7}$$

Because of liquid water's low compressibility, the density of water is assumed invariable with respect to temperature and pressure. Cancelling the density term across the equation gives the following final general mass balance around each dewatering level:

$$\frac{dL_f}{dt} = V_c(Q_{\text{in}} - Q_{\text{out}}) \tag{8}$$

Solution of the Equation (8) will be done iteratively, in one-second-intervals, so an analytical equation akin to Euler's method is used instead of differential equations:

$$\frac{\Delta L_f}{\Delta t} = V_c(Q_{\text{in}} - Q_{\text{out}})$$

$$\Delta L_f = \Delta t V_c(Q_{\text{in}} - Q_{\text{out}}) \tag{9}$$

$$L_{f,\,\text{old}} - L_{f,\,\text{new}} = \Delta t V_c(Q_{\text{in}} - Q_{\text{out}})$$

$$L_{f,\,\text{new}} = L_{f,\,\text{old}} + \Delta t V_c(Q_{\text{in}} - Q_{\text{out}}) \tag{10}$$

In the above equations, L_f represents a dam level as a fraction of its total capacity. Equation (10) can be rewritten in terms of the dam level as a percentage, L:

$$L_{\text{new}} = L_{\text{old}} + \Delta t \frac{100\%}{V_c}(Q_{\text{in}} - Q_{\text{out}}) \tag{11}$$

PUMP SCHEDULING ALGORITHM

As mentioned in Section 1.3, the control systems will be considered as being x-factored, indicating the number of main control variables being considered by the system. In this study, the PLC implementation is an example of 1-factor control (where only the dam being pumped from (the upstream dam) dam level is considered). The example for 2-factor control is SCADA control (which considers both the levels of the upstream, as well as downstream dam). Third-party control software is the specific example for n-factor control in this study.

1-FACTOR AND 2-FACTOR DECISION-MAKING ALGORITHM

Decision-making for the 1-factor and 2-factor control systems is very similar. The algorithm presented in this section is the one considered for 1-factor and 2-factor control, as it is the algorithm used by the case study mines' SCADA for pump scheduling (available in Appendix E, page 128). The 1-factor control system considers only the dam level for the dewatering dams that it is employed for, whereas the 2-factor model also considers the level of the downstream dam(s). Refer to Figure 13 (page 36), which is a graphical representation of this algorithm.

Part 1: Setup

Step 1a: Calculate the average dam level.

Only one dam level sensor is used per dam. Values are included in the calculation of the average level if the values are positive and non-zero, as well as "good" quality as defined by the SCADA.

Step 1b: Downstream dam higher and lower limits ("UL HL" and "UL LL") are set. UL HL is the maximum dam level for the downstream dam so that pumps are not

recommended to run if the dam is “full”. UL LL is the downstream dam level at which pumps can be recommended to run again[7].

The reason for two level limits is to serve as hysteresis. If only one value is used, recommendations will cycle pumps to keep dams at this one dam level.

This is only the case for 2-factor control. For 1-factor control, this step is skipped, since downstream dam levels are not taken into consideration.

Step 1c: Set the dam level (percentage) amount of hysteresis that should be used to determine if all pumps are to be switched off.

Step 1d: A table is set up, specifying when pumps should be switched on. This table lists dam levels for each Eskom ToU period when one, two, three, and so forth, pumps are required to run. As an example, refer to Table 3, below:

TABLE 3 – MINE SCADA PUMP SCHEDULER SETUP TABLE

	Peak	Standard	Off-peak
1 pump required at	80%	65%	30%
2 pumps required at	85%	75%	40%
3 pumps required at	90%	85%	50%
4 pumps required at	95%	95%	90%

Part 2: Decision-making

Here, the previously set-up table (Table 3) is used to determine the number of pumps that should be running.

Step 2a: 1-factor control: Skip this step.
2-factor control: If the downstream dam level is higher than, or exceeds UL HL, then no pumps are recommended to run.

Step 2b: 1-factor control: Continue to Step 2c.
2-factor control: Continue to Step 2c only if the downstream dam level is lower than or equal to UL LL.

[7] Example: If the downstream dam is considered to be “full”, the current dewatering level’s pump schedule recommendation will be to run zero pumps. Conversely, the proposed schedule by the SCADA pump controller object on the downstream or upper level will be to *not* run zero pumps. If this is followed, the downstream dam level will decrease. Only when this level decreases and reaches UL LL, will the pump controller object for the current level start giving non-zero recommendations again.

Step 2c: Determine which Eskom ToU period is currently in effect. This determines the column in the table that should be used.

Step 2d: Determine how many pumps are available, with SCADA communication, on the current dewatering level.

Step 2e: Counting from one to the number of available pumps will now start. Some decisions are made in the following steps, and if needed, decision-making will repeat from this step (Step 2e).

Step 2f: For the current count, read the dam level that is required for this number of pumps to run (from Table 3). If this level from the table is smaller than or equal to the current average dam level, then this count is how many pumps are tentatively required to run.

Step 2g: The recommendation from Step 2f is checked to determine if it should be altered to switch off pumps. If the current average dam level is lower than the level read from the table in Step 2f minus the hysteresis set in Step 1c, the tentative recommendation changes to zero pumps running.

Step 2h: Now the final recommendation is made:

- If the tentative recommendation is zero pumps, then the final recommendation is also to switch off all pumps.
- If the tentative recommendation is *not* to run zero pumps, then the number of "excessive" pumps is determined. If less than two too many are running (compared to the tentative recommendation from Step 2f), go back to Step 2e, incrementing the count.
- More than two too many pumps indicate that pumps need to be switched off. To avoid sudden drastic changes, pumps will be switched off one by one: the final recommendation is to run one pump more than the tentative number of pumps from Step 2f.
- If the final recommendation has been made, return to the start of the algorithm (Step 2a).

The above algorithm can be graphically illustrated as indicated in Figure 13, on the next page.

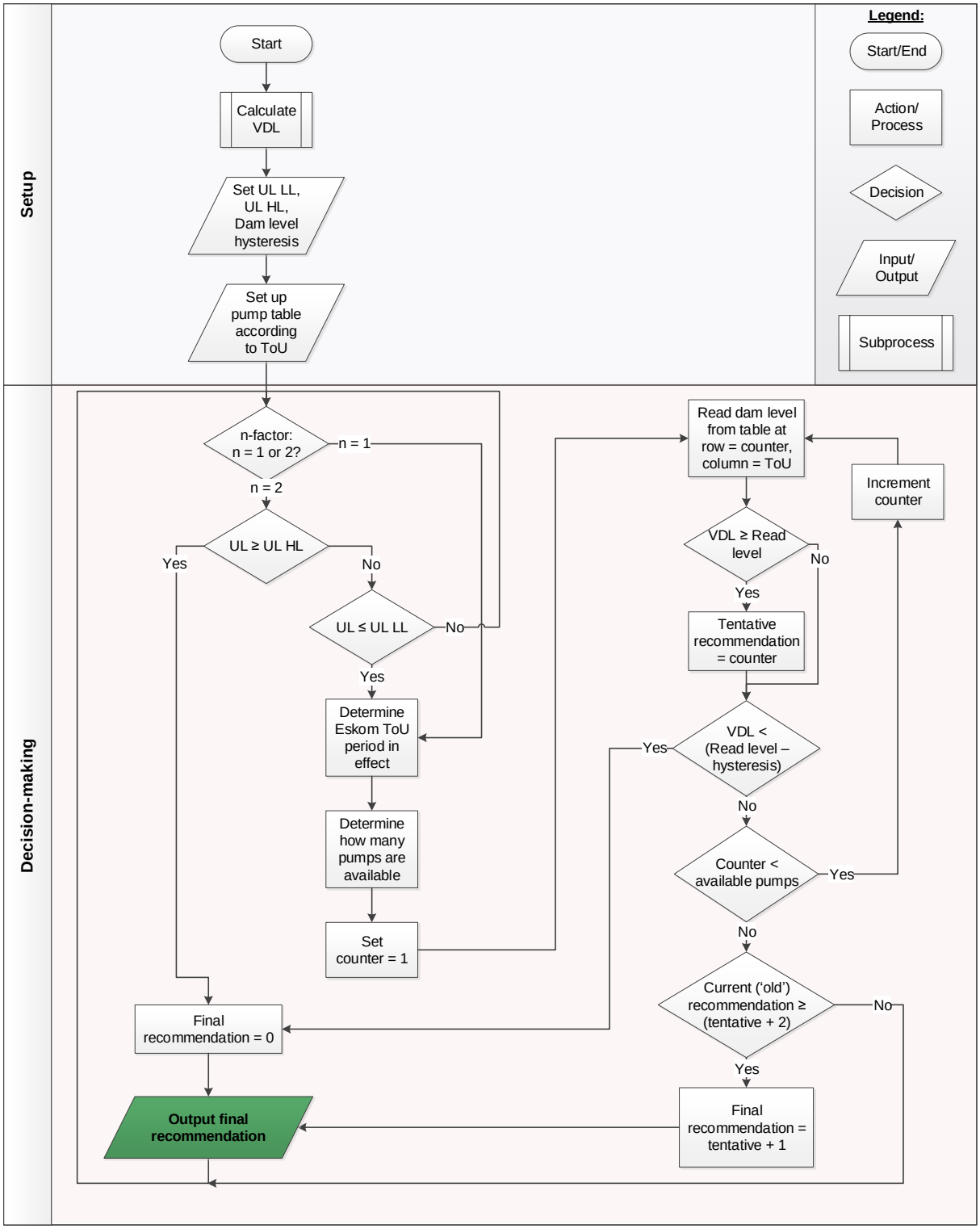


FIGURE 13 – MINE SCADA PUMP SCHEDULER ALGORITHM (→ 1-FACTOR AND 2-FACTOR CONTROL)

n-FACTOR DECISION-MAKING ALGORITHM

An industrial pump dewatering simulation and control software package, Real-Time Energy Management System (REMS), was selected for simulating n-factor control. Reasons for making use of REMS include availability and knowledge of this software to the author.

To set up a simulation or control model in REMS is relatively easy, seeing as the software is developed especially for this purpose. The building of the model is intuitive because the model layout represents the actual site layout. The software has built-in libraries of objects that the user can insert and customize via simple clicks of the mouse.

The REMS pump controller performs the algorithm below for selecting a number of pumps to run. The algorithm is also presented graphically in Figure 14. For each dam level being controlled with its pumps pumping from this dam:

- Define the minimum and maximum numbers of pumps that are allowed to run[8].
- Define a "control range", "bottom offset" and "top offset"[8].
- The pump controller calculates a "lower" and "upper bound" (Equations (12) and (13)). The controller will attempt to control the dam level between these two values.

$$\text{Lower bound} = \begin{cases} \text{Eskom non-peak period:} & \text{Minimum} \\ \text{Eskom peak period:} & \text{Maximum} - \text{control range} \end{cases} \quad (12)$$

$$\text{Upper bound} = \begin{cases} \text{Eskom non-peak period:} & \text{Minimum} + \text{control range} \\ \text{Eskom peak period:} & \text{Maximum} \end{cases} \quad (13)$$

- From these bounds, "pump starting" and "pump stopping" values are calculated, where p represents the number of pumps that should be running:

$$\text{Pump starting values} = \text{Upper bound} + (p-1) \times \text{top offset} \quad (14)$$

$$\text{Pump stopping values} = \text{Lower bound} - (p-1) \times \text{bottom offset} \quad (15)$$

  When the dam level reaches a pump starting or stopping value, the amount of pumps this value is for, should be running. Thus, if the dam level rises and reaches the upper bound, one pump will be started. If the level then continues to rise and reaches the upper bound + $1 \times$ top offset, another pump will be started. This process continues until the maximum number of pumps is running. Similarly, if the dam level is decreasing and reaches the lower bound, one pump will be stopped. From there on, every time an additional bottom offset value is reached, an extra pump will be stopped, until the minimum number of pumps is running.

[8] The minimum and maximum number of pumps, as well as control range and offset values can be fixed values or values dynamically calculated by REMS in a custom programmable tag. The 1-factor and 2-factor control systems only allow for fixed values.

This logic allows all pumps to be switched off during an Eskom peak period if the dam level is sufficiently low. At the end of the peak period, all pumps might be switched on if the dam level is higher than the minimum + control range + (maximum pumps − 1) × top offset.

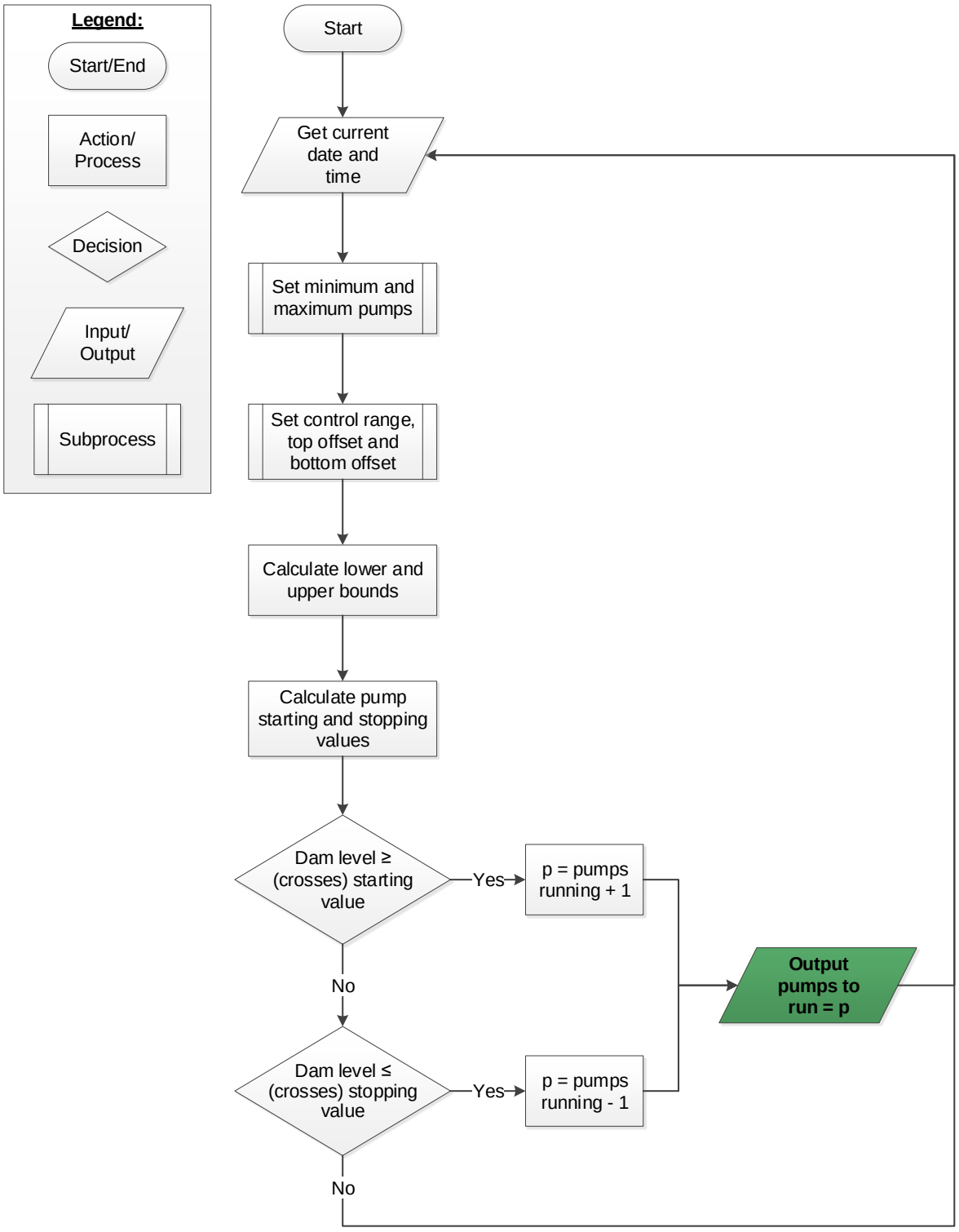


FIGURE 14 – REMS PUMP CONTROLLER ALGORITHM (→ n-FACTOR CONTROL)

3.4.2. SIMULATION

The model developed previously (Equation (11)) is solved for each system. An iterative approach is followed, where a new set of values is calculated for each dewatering system every simulated second. After calculations, control decisions are made according to the algorithms described in Section 3.4.1 (page 33), ready to be effective the next second.

Initial values of the mathematical model (Equation (11)) are chosen at the start of the simulation. A single day is simulated for each case study. The initial values for the simulations are the same as the actual values that the respective days start with (dam levels and number of pumps running).

The developed model was simulated using the programming language Python[9], along with additional modules NumPy[10] and Pandas[11]. All source code is available in Appendix F, and electronically on GitHub (http://bit.ly/2rqp9IE), which provides an optimal browsing experience of data processing Jupyter notebooks and source code.

The rest of Section 3.4.2 is devoted to a more thorough explanation of the simulation development.

GENERIC SIMULATION MODULE DEVELOPMENT

REMS has a built-in customisable simulation mode. It is possible to incorporate the mine SCADA pump scheduling algorithm (for 1-factor and 2-factor control) into REMS, through the customisation of REMS's pump controller objects. However, these objects are designed to follow REMS's n-factor control algorithm, and incorporating another algorithm will be performed in a clumsy and inelegant manner. For this reason, it was decided that it was best to control simulations manually using the model derived from first principles (Equation (11)).

The model was solved iteratively using the Python programming language. Python was selected as programming language because of its learning ease and the availability of third-party modules (though none is strictly needed). This provides complete control over all aspects of the simulations. Because of the generic approach followed, as becomes known in the rest

[9] Python v3.6.2: Python Software Foundation. Python language reference, version 3.6. Available at https://docs.python.org/3.6/reference/index.html

[10] NumPy v1.13.1: Van der Walt, S., Colbert, S. C. & Varoquaux, G. 2011. The NumPy Array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30. DOI: 10.1109/MCSE.2011.37.

[11] Pandas v0.20.3: McKinney, W. 2010. Data structures for statistical computing in Python. (*In* van der Walt, S. & Millman, J. *eds*. Proceedings of the 9th Python in science conference. Austin, TX: SciPy. p. 51–56).

of this section, simulations are performed using only a single line of code. This provides a quicker method of performing multiple simulations.

In order to make multiple dewatering systems' simulations easier to perform, it was decided to follow a generic approach towards simulation creation. A custom `pumping system` module was created. This module serves three purposes:

- Creation of `pumping level` objects.
- Creation of a `pump system` object and adding to the aforementioned `pumping level` objects to this `pump system`.
- Performing of dewatering system simulations in a specified `simulation mode`.

The three above-mentioned purposes are explained below:

PUMPING LEVEL OBJECT

The `pumping level` object is an object containing all variables and information related to a specific pumping level. It contains a list of attributes specific to a level:

- Name.
- Capacity.
- Initial level (initial value at the start of a simulation).
- Pump delivery flow.
- Pump power consumption.
- Initial number of pumps running.
- Inflow of water. This can be specified as a constant value or a 30-minute flow profile (a `NumPy` array with shape `(24, 2)`).
- The name of a level that is fed by this level (optional).
- `Validation mode` values: a list of pump status values that the simulation should follow in `validation mode` (explained in Section 3.6). This list has 86 400 values – the number of seconds in a day.
- `1-factor` and `2-factor control mode` values:
  - Pump schedule table (used for 1-factor and 2-factor control). Refer to Table 3, page 34, as an example.
  - Hysteresis.
  - Downstream (upper-level) dam lower limit (UL LL).
  - Downstream (upper-level) dam higher limit (UL HL).
- `n-factor control` values:
  - Minimum amount of pumps.
  - Maximum amount of pumps.

- Minimum dam level.
- Maximum dam level.
- Control range.
- Bottom offset.
- Top offset.

The `pumping level` object's code is available is available in Appendix F (page 144), and electronically on GitHub (http://bit.ly/2rqp9IE).

PUMP SYSTEM OBJECT AND PERFORMING SIMULATIONS

The `pump system` object is an object containing a list of all `pumping levels` making up this pumping system. The developed module can be used for any system containing at least one dam and one pump – not just mine dewatering systems.

The `pump system` object also contains the core code – the simulation. The simulation consists of a `for loop` with a `step size` of `1`. Each iteration represents one second. During each iteration, the current dam levels and pump statuses are read and a decision is made on how many pumps should be recommended to run. After implementing this recommendation, the new dam level is recalculated based on the derived model (Equation (11)).

The decision of how many pumps to run is based on the control system that is specified (the `simulation mode`). The simulations have four available modes:

- `Validation mode`
  Instead of making use of some algorithm to decide how many pumps should run, this mode uses the same running schedule as was followed in reality (this is explained in more detail in Section 3.6). This means that validation mode skips the decision-making part of the code and goes directly to the model calculation part.
- `1-factor mode`
  This mode follows the decision-making algorithm explained in Section 3.4.1, page 33.
- `2-factor mode`
  This mode follows the decision-making algorithm explained in Section 3.4.1, page 33.
- `n-factor mode`
  This mode follows the decision-making algorithm explained in Section 3.4.1, page 36.

After completing the simulation for one day (86 400 iterations or seconds), the simulation results are optionally saved to a `csv` file. This is performed using the third-party library `pandas`, because of its exceptional and easy to use data structure and tabular data handling capabilities.

The `pump system` object's code is available is available in Appendix F (page 146), and electronically on GitHub (http://bit.ly/2rqp9IE).

* * *

Using the developed generic `pump system simulation module` enables multiple simulations to be performed quickly and easily. A `pumping system` object is created only once and from there on, any simulation can be run by only using a single line of code. The generic approach provides encapsulation from the core code, allowing the user to focus only on `pump system` setup.

3.4.3. Simulation/model assumptions

The following assumptions were made for simulation of the control systems:

- All pumps are available at all times.
- For comparison purposes, the Eskom demand season in effect is trivial.
- Start-up and shutdown of pumps, as well as changes in delivered flow as a pump is spinning up or down, are instantaneous.
- Pumps have constant delivery flows.
- Multiple pumps running simultaneously deliver the sum of the pumps' flows.
- Pumps can be started and stopped simultaneously.
- Pump efficiency is implicitly taken into account by the pump flows.
- Data polling/refresh rate is 1 second.

The simulation testing process (Sections 3.5 and 3.6) ascertains the validity of the above assumptions.

3.5. Model and simulation verification

Verification is concerned with whether the developed model and simulation have been implemented correctly. It is important to verify the correctness of the simulation, before drawing conclusions from its outputs.

The simulation module developed in Section 3.4.2 consists of two main parts:

- Model calculation, which calculates the change in dam level.
- Pump scheduling, based on 1, 2 and n-factor control decision-making algorithms.

The two parts of the simulation can be verified separately.

3.5.1. VERIFICATION OF MODEL IMPLEMENTATION

In order to verify if the model describing the change in dam level is derived and implemented correctly, a set of simple simulations can be performed:

- Model verification simulation 1: dam with constant inflow.
- Model verification simulation 2: dam with variable inflow.
- Model verification simulation 3: dam with inflow and a running pump.

SIMULATION M1: DAM WITH CONSTANT INFLOW

A verification simulation is performed with the following characteristics:

- Dam: one 1 ML dam.
- Initial dam level: 0%.
- Inflow: constant 10 L/s.

The expected dam level after one day (86 400 seconds) can be calculated and compared with the simulation results. For the above-mentioned characteristics, the amount of water flowing into the dam in one day can be calculated as

$$\frac{10\ \text{L}}{\text{s}} \times \frac{1\ \text{day}}{} \left| \begin{array}{c|c|c} \frac{24\ \text{h}}{1\ \text{day}} & \frac{60\ \text{min}}{1\ \text{h}} & \frac{60\ \text{s}}{1\ \text{min}} \end{array} \right.$$

$$\frac{10\ \text{L}}{\text{s}} \times \frac{86\ 400\ \text{s}}{}$$

$$= 864\ 000\ \text{L}$$

This amount of water contained within the dam can be converted to a dam level (percentage) as follows:

$$\frac{864\ 000}{1\ 000\ 000} \times 100\% = 86.4\%$$

A simulation is performed with the same input values as listed (refer to Appendix F, Section F.3.2 for the programmatic simulation setup). The dam level profile as in Figure 15 is obtained from the simulation:

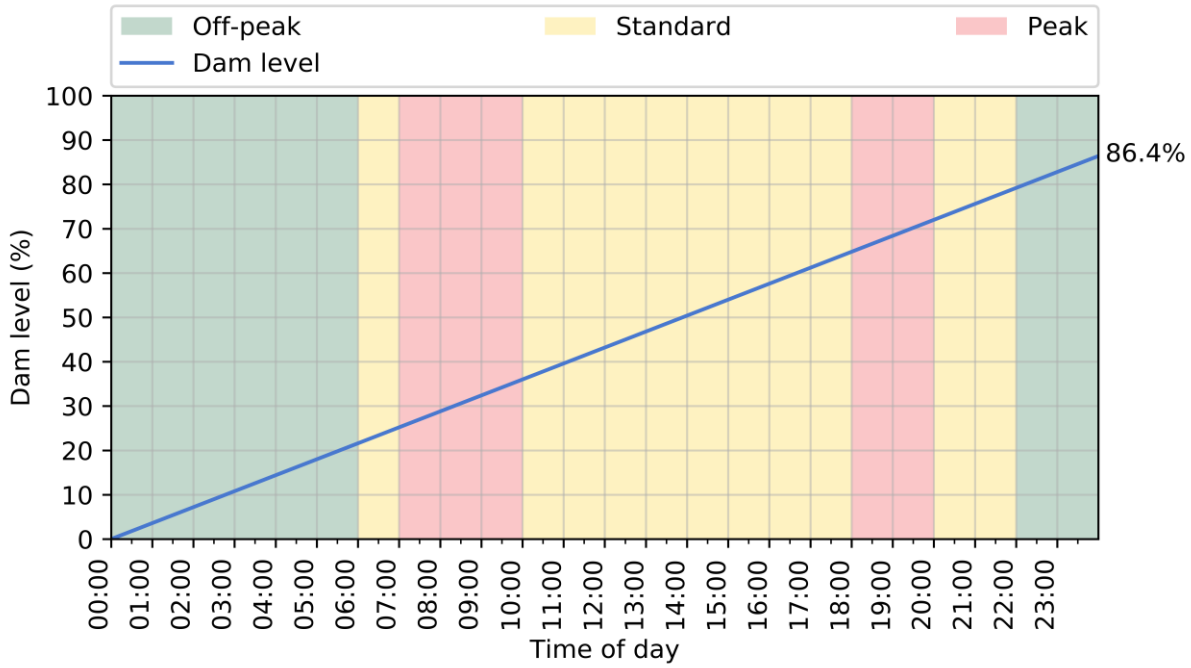


FIGURE 15 – MODEL VERIFICATION SIMULATION M1 OUTPUT: DAM WITH CONSTANT INFLOW

From Figure 15, it can be seen that the simulation has a final dam level of 86.4%. This value corresponds with the expected value. This shows that the model and simulation implementation correctly characterises the constant inflow of water into a dam.

SIMULATION M2: DAM WITH VARIABLE INFLOW

A verification simulation is run with the following characteristics:

- Dam: one 1 ML dam.
- Initial dam level: 10%.
- Inflow: 5 L/s for the first half of the day and 10 L/s for the second half.

The expected dam level after one day can be calculated and compared with the simulated result. For the above-mentioned characteristics, the amount of water flowing into the dam in one day can be calculated, this time directly using the derived model (Equation (11)):

$$L_{\text{new}} = L_{\text{old}} + \Delta t \frac{100\%}{V_c}(Q_{\text{in}} - Q_{\text{out}})$$

$$\Rightarrow 10\% + \frac{5\ \text{L}}{\text{s}} \left| \frac{86\,400\ \text{s}}{2} \right| \frac{100\%}{1\,000\,000\ \text{L}} + \frac{10\ \text{L}}{\text{s}} \left| \frac{86\,400\ \text{s}}{2} \right| \frac{100\%}{1\,000\,000\ \text{L}}$$

$$= 74.8\%$$

A simulation is performed with the same values (refer to Appendix F, Section F.3.3 for the programmatic simulation setup). The dam level profile as in Figure 16 is obtained from the simulation:

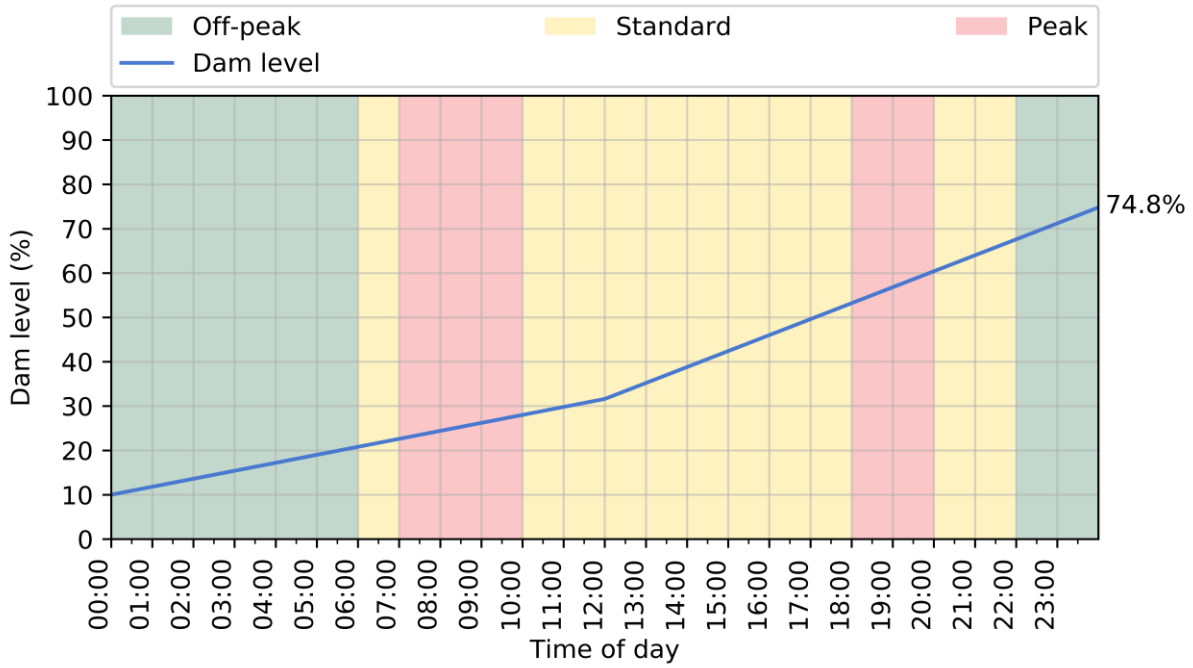


FIGURE 16 – MODEL VERIFICATION SIMULATION M2 OUTPUT: DAM WITH VARIABLE INFLOW

From Figure 16, it can be seen that the simulation has a final dam level of 74.8%. This value corresponds with the expected value. This shows that the model and simulation implementation correctly characterises variable inflow into a dam.

SIMULATION M3: DAM WITH INFLOW AND A RUNNING PUMP

A verification simulation is run with the following characteristics:

- Dam: one 1 ML dam.
- Initial dam level: 10%.
- Inflow: constant 50 L/s.
- Pump: one pump constantly on, with a delivery flow of 40 L/s.

The expected dam level after one day can be calculated and compared with the simulated result. For the above-mentioned characteristics, the amount of water flowing into the dam in one day can be calculated:

$$L_{\text{new}} = L_{\text{old}} + \Delta t \frac{100\%}{V_c}(Q_{\text{in}} - Q_{\text{out}})$$

$$\Rightarrow 0\% + \frac{50\ \text{L}}{\text{s}} \left| \begin{matrix} 86\,400\ \text{s} \\ \end{matrix} \right| \frac{100\%}{1\,000\,000\ \text{L}} - \frac{40\ \text{L}}{\text{s}} \left| \begin{matrix} 86\,400\ \text{s} \\ \end{matrix} \right| \frac{100\%}{1\,000\,000\ \text{L}}$$

$$= 86.4\%$$

A simulation is performed with the same values (refer to Appendix F, Section F.3.4 for the programmatic simulation setup). The dam level profile as in Figure 17 is obtained from the simulation:

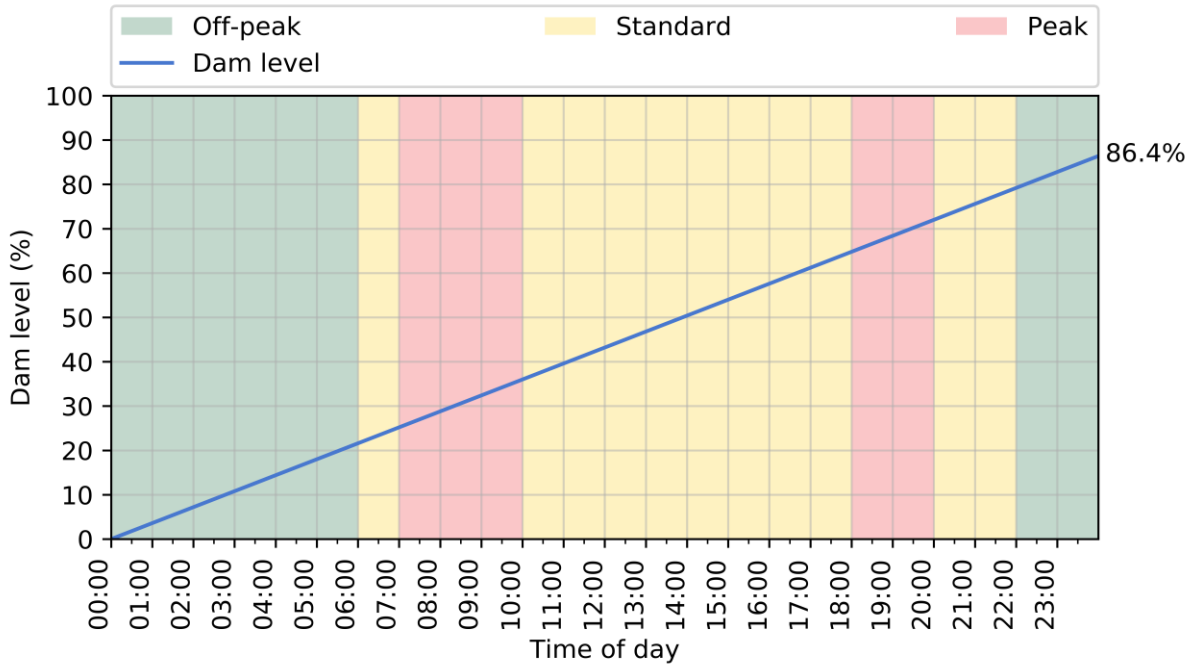


FIGURE 17 – MODEL VERIFICATION SIMULATION M3 OUTPUT: DAM WITH INFLOW AND A RUNNING PUMP

From Figure 17, it can be seen that the simulation has a final dam level of 86.4%. This value corresponds with the expected value. This shows that the model and simulation implementation correctly characterises the effect of a pump pumping from a dam.

3.5.2. VERIFICATION OF PUMP SCHEDULING DECISION-MAKING ALGORITHM IMPLEMENTATIONS

Simple simulations for the SCADA and REMS decision-making algorithms will reveal if these algorithms were correctly implemented.

SIMULATION D1: MINE SCADA ALGORITHM (1-FACTOR AND 2-FACTOR CONTROL)

A verification simulation is run with the following characteristics:

- Dam: one 1 ML dam.
- Initial dam level: 0%.
- Inflow: constant 20 L/s.
- Pumps: Two pumps available with a delivery flow of 15 L/s each. The following pump scheduler setup table (Table 4) shows when pumps should be switched on or off as per the mine SCADA algorithm (page 36).
- Initial number of pumps running: 0.

TABLE 4 – VERIFICATION SIMULATION D1: PUMP SCHEDULER TABLE

	Peak	Standard	Off-peak
1 pump required at	70%	70%	70%
2 pumps required at	75%	75%	75%

The first pump is expected to be switched on when the dam level reaches 70%. When the dam level reaches 75%, the second pump is expected to be switched on. As per the mine SCADA pump scheduling algorithm, a pump is only switched off when “two too many pumps” are running. This means that a pump will be switched off when the dam level reaches 65% (70% – hysteresis). Coincidentally, when this 65% value is reached, this is the value that triggers all pumps to be switched off. To state this again, at 65%, all pumps are expected to be switched off.

A simulation is performed with the same values (refer to Appendix F, Section F.3.5 for the programmatic simulation setup). The dam level profile as in Figure 18 is obtained from the simulation:

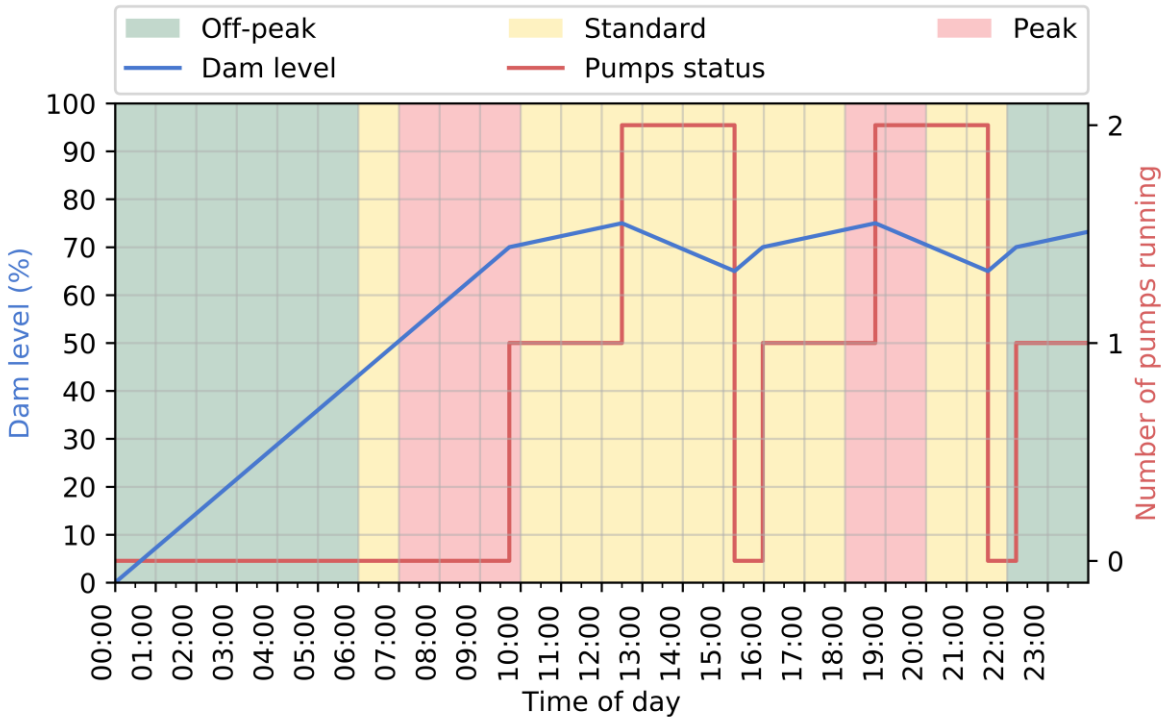


FIGURE 18 – DECISION-MAKING VERIFICATION SIMULATION D1 (MINE SCADA ALGORITHM) OUTPUT

From Figure 18, it can be seen that the dam level rises up to 70% while no pumps are running. When the dam level reaches 70%, one pump is switched on. The dam level rises further up to 75%, after which another pump is switched on. With two pumps running, the outflow from the dam caused by the pumps exceed the inflow, causing the dam level to decrease. When 65% is reached, both pumps are switched off. This pattern repeats for the rest of the day.

The simulation showed to follow the expected pump scheduling behaviour. This means that the mine SCADA pump scheduling algorithm has been correctly implemented and can be used for 1-factor and 2-factor control simulations.

SIMULATION D2: REMS ALGORITHM (n-FACTOR CONTROL)

A verification simulation is run with the following characteristics:

- Dam: one 1 ML dam.
- Initial dam level: 0%.
- Inflow: constant 20 L/s.
- Pumps: Two pumps available with a delivery flow of 15 L/s each.
- Minimum number of pumps: 0.
- Maximum number of pumps: 2.
- Minimum dam level: 15%.
- Maximum dam level: 77%.
- Control range: 5%.
- Bottom offset: 5%.
- Top offset: 2.5%.

When performing a simulation with the above inputs, during non-peak hours, it is expected that one pump should be switched on when the dam level reaches 20% (minimum + control range). The second pump is expected to be switched on at 25% (minimum + 2 × control range). Furthermore, the first pump is expected to be switched off at 15% (the minimum dam level) and the second at 10% (minimum – bottom offset). During Eskom peak hours, if the dam level is lower than 72% (maximum – control range), all pumps are expected to be switched off.

A simulation is performed with the same values (refer to Appendix F, Section F.3.6 for the programmatic simulation setup). The dam level profile as in Figure 19 is obtained from the simulation:

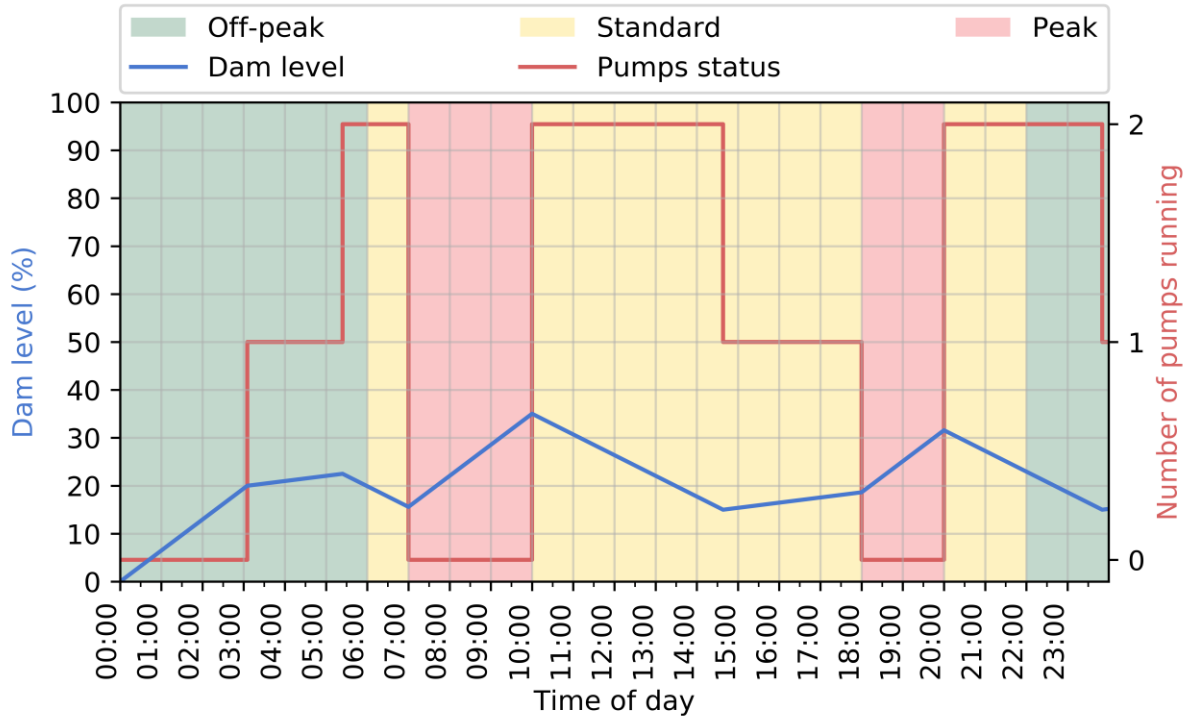


FIGURE 19 – DECISION-MAKING VERIFICATION SIMULATION D2 (REMS ALGORITHM) OUTPUT

From Figure 19, it can be seen the simulation followed the expected pump scheduling behaviour. This shows that the REMS pump scheduling algorithm has been correctly implemented and can be used for n-factor control simulations.

* * *

From the verification simulations as presented in Section 3.5, it is concluded that the dam level model and pump scheduling algorithms have been correctly implemented. Successful verification ensures that simulations using the developed simulation module can be used to characterise the working of actual pumping systems.

3.6. VALIDATION METHODOLOGY FOR CASE STUDY SIMULATIONS

Validation is concerned with whether the correct dewatering system is being simulated. It is important ensure that the simulation model is indeed representative of the actual dewatering system being considered, before drawing conclusions from its outputs.

Before control simulations can be run and conclusions made on a dewatering system, the integrity of the base case simulation for this system first needs to be determined. The base case simulation is a simulation of the system under the same control conditions as was followed for the actual day. The simulation is *not* controlled by the control systems of this study. The validation methodology will ensure that the dewatering system is simulated within an acceptable range of accuracy. Upon successful validation of the simulation, the simulation

can be used to test how the different control systems will interact with the process variables of the dewatering system.

Historical data is used to validate that the model and computerised simulation thereof is correct. For validation, a simulation is run using the initial values of a selected day from historical data. Instead of using pump scheduling algorithms to determine the number of pumps that should be running, the actual pump statuses from that day is used. This means that if the same control is done by the simulation as was done in reality, dam profiles obtained through the simulation should match the actual dam profiles of that day.

In order to test the degree of similarity between simulated and actual results, the root mean square deviation (RMSD; Appendix B) is used (Kobayashi & Salam, 2000; Gauch *et al.*, 2003; Piñeiro *et al.*, 2008; Pedregosa *et al.*, 2011). This value represents the mean deviation of predicted/simulated/calculated values with respect to the observed/actual values, in the same units as the variable under evaluation. The lower this value, the closer the simulated and actual values are to each other, or stated otherwise, the more accurate the simulation.

3.7. SCORING

Due to differences between dewatering systems, the three control systems and how well they are able to control the systems are not directly comparable. In order to facilitate this comparison, a scoring system is developed. For each dewatering system, each control system will be given a score out of 100 points. Afterwards, the scores can be compared to be able to draw conclusions. The developed scoring consists of two main scores: a “performance score” and a “feature score”.

3.7.1. PERFORMANCE SCORE

Each simulation has pump running schedules, dam level and power consumption profiles as outputs. These outputs are used to calculate its corresponding performance score. The performance score consists of three parts:

(a) the cost score,
(b) the level score, and
(c) the pump cycling score.

These are all factors considered as important when controlling pumping systems.

COST SCORE

From each dewatering level’s simulated pump power consumption profiles, the entire dewatering system’s power profile can be determined. Seeing as the simulation is run in one-

second resolution, the simulated power profile is also in one-second resolution. In line with Eskom's Measurement and Verification (M&V) practices, the power profile dataset is resampled to 30-minute resolution. Using the Eskom tariff structure applicable to the site, the electricity cost of the power profile can be calculated. This is called the “simulated profile cost” (“SPC”).

The next step is to calculate the “best profile cost” (“BPC”). Seeing that many factors play a role in determining the optimal running profile of any site, some simplification is needed.

In order to calculate the optimal cost, the maximum number of pumps allowed to run simultaneously is used. From this a maximum possible power consumption can be derived. Each Eskom ToU period consists of a fixed amount of time, so it is possible to calculate the maximum amount of energy that can be used for each ToU period if the maximum number of pumps are run for the entire period. The amount of energy required to remove x amount of water from the mine is known from the simulated power profile. This amount is then filled into the Eskom ToU periods, in order of least to most expensive. In doing this, total amount of energy needed to dewater the system for one day has been divided into three ToU periods, and the cost can again be calculated using the Eskom tariff structure. This is illustrated in Figure 20:

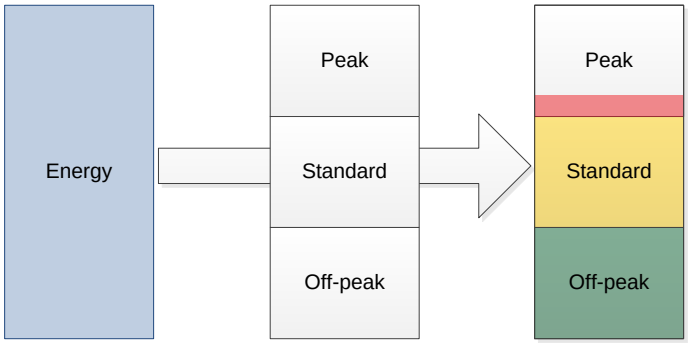


FIGURE 20 – DIVIDING ENERGY INTO ESKOM TOU PERIODS

The cost score can now be calculated by dividing the BPC by the SPC, resulting in a value between zero and one:

$$\text{Cost score} = \frac{\text{Best profile cost (BPC)}}{\text{Simulated profile cost (SPC)}} \tag{16}$$

LEVEL SCORE

The level score is an indication of the capability of the control system to control the dam levels within their control ranges. Calculation of the level score is simple: the number of dams that stayed within penalisation limits is divided by the total number of dams, resulting in a score

between zero and one (Equation (17)). A value of one indicates that all dam levels were controlled in a satisfactory manner. A value of zero indicts that none of the dams were controlled satisfactorily.

$$\text{Level score} = \frac{\text{Number of dams controlled within penalisation limits}}{\text{Total number of dams}} \tag{17}$$

PUMP CYCLING SCORE

The pump cycling score is an indication of the capability of the control system to schedule the pumps in such a way as to not have a negative impact on life or maintenance cost/frequency of the pump. Determination of the pump cycling score is simple, as shown by Equation (18). If no pump cycling occurred (as defined in Section 2.3.2), a score of one is awarded. If unnecessary/preventable pump cycling occurred, a score of zero is awarded.

$$\text{Pump cycling score} = \begin{cases} \text{No cycling: } 1 \\ \text{Cycling: } 0 \end{cases} \tag{18}$$

COMBINED PERFORMANCE SCORE

The three scores calculated above are now combined into the total performance score:

$$\text{Performance score} = \frac{\text{Cost score} \times w_1 + \text{Level score} \times w_2 + \text{Pump cycling score} \times w_3}{w_1 + w_2 + w_3} \tag{19}$$

where w_i represents a user-definable weights. In other words, the performance score is a weighted average of its components.

3.7.2. FEATURE SCORE

The feature score serves as a means of evaluating each control system's features and ease of use, as this also plays a part in deciding which control system to use and how it will be used.

From features identified in Section 2.4.4, a feature table is set up (Table 5). In this table, each feature is given a score between zero and three, indicating how well that feature is performed. To calculate the total feature score, the score achieved by evaluation the control system is divided by the maximum achievable score, resulting in a value between zero and one.

Each feature consists of subcomponents for evaluation or is evaluated relative to other control systems' features. Table 5 summarises the features identified. Items listed in grey are the descriptions of the contributing factors adding up to the total score of each feature (underlined).

TABLE 5 – CONTROL SYSTEM FEATURE SCORE BREAKDOWN

	Score awarded	Maximum possible score
Simulation / testing environment - Yes = 3. - No = 0.	• • •	3 3 0
Optimised pumping schedule - No = 0. - Follow/recommend, leading to reduced electricity cost. If not, -1. - Workload balancing. If not, -1.	•	3
Control and automated operation - Able to control = 1. - Human assistance not needed = 1. - Emergency situation handling = 1.	• • • •	3 1 1 1
Monitoring and reporting - Monitoring = 1. - Logging = 1. - Reporting = 1.	• • • •	3 1 1 1
Alarm handling - Scoring relative to other control systems.	•	3
Alarm handling - Scoring relative to other control systems.	•	3
	•	18
	•/18	

3.7.3. COMBINED TOTAL SCORE

Similarly to how the performance score is built up from its components, the total score consists of the performance score and feature score:

$$\text{Total score} = \frac{\text{Performance score} \times w_4 + \text{Feature score} \times w_5}{w_4 + w_5} \tag{20}$$

Though trivial, this score is multiplied by 100:

$$\text{Total score} = 100 \times \frac{\text{Performance score} \times w_4 + \text{Feature score} \times w_5}{w_4 + w_5} \tag{21}$$

3.8. STUDY VALIDATION METHODOLOGY (FURTHER VALIDATION)

To ascertain the validity of the study, the study will be evaluated in terms of its internal and external validity. The study is internally valid if it is content and construct valid, i.e. the control system performance, or total score, (the "construct") is measured by the elements in Section 3.7, and the measurement procedure is valid. External validity refers to whether the conclusions of the study can be generalised.

3.9. CONCLUSION

This chapter focused on developing a methodology by means of which to compare different control systems for mine dewatering. The control systems are considered as being 1-, 2- or n-factored (x-factored).

For comparison purposes, control systems are evaluated by means of simulation. After verifying the integrity of the simulation, simulation results are used to calculate a performance score, which represents the control system's ability to control the dewatering system parameters within control ranges. The control systems are also scored based on the features they offer. The feature score and performance scores are combined, resulting in a total score, which can be used to directly compare the control systems.

COMPARATIVE STUDY OF MINE DEWATERING CONTROL SYSTEMS

CHAPTER 4

RESULTS AND DISCUSSION

4. RESULTS AND DISCUSSION

4.1. INTRODUCTION

In Chapter 3, a method for comparing the control systems was developed. This comparison consists of calculating performance and feature scores, which describe how the control systems are able to control the dewatering system. In this chapter, this method is implemented on and results presented for three case studies. Feature scoring results are independent of the case studies (i.e. not site-specific), therefore this is not repeated for each case study.

4.2. FEATURE SCORING RESULTS

This section presents the feature scoring results for the different control systems. For each control system, each feature is evaluated and given a score varying from zero to three, along with an explanation as to why the score was given. Afterwards, the scores are summarised and the total feature scores are presented.

A gold mining group provided the three case studies considered in this study. The control systems as will be considered for use by the mines are evaluated.

4.2.1. 1-FACTOR FEATURE SCORING

PLC implementation is the example of 1-factor control in this study.

SIMULATION/TESTING ENVIRONMENT

It is possible to simulate and verify the setup of inputs and outputs (the functionality) of the PLC by using simulation software, such as Siemens's SIMATIC S7-PLCSIM Advanced. However, PLCs do not provide an environment for performing process and process control simulations. There is no built-in method of simulating what the effect of control parameter changes would be. The control philosophy can not be easily tested. Thus, a score of 0/3 is awarded.

OPTIMISED PUMPING SCHEDULE

With this control system, mostly the same pumping schedule as with 2-factor control is followed. If the input parameters to the pump scheduling algorithm are fine-tuned, it is possible that this algorithm will output optimised pumping schedules. However, the PLC only recommends *a number* of pumps to run and not *which* pumps to run. This means that this control system is unable to balance the workload of the pumps. A score of 2/3 is awarded.

CONTROL AND AUTOMATED OPERATION

This control system is able to control the components of the pumping system. In emergencies, scheduling will continue, or pumps can be switched to local operation mode, ignoring scheduling instructions from the PLC. In the case of incorrect programming or setup on a PLC, this might lead to incorrect process control. Even if control can be temporarily disabled, this will not be possible for control room operators. There will, therefore, be a delayed response in disabling control. A score of 2/3 is awarded.

MONITORING AND REPORTING

If display devices are installed, PLCs are able to provide monitoring capability to operators. PLCs do not offer data management or database capability: data logging and reporting are not offered. A score of 1/3 is awarded.

ALARM HANDLING

PLC as control system is able to raise audible and visual alarms at the pump station if the required hardware is installed. Since the other control systems offer more in terms of alarm handling capability, a score of 1/3 is awarded for PLC control.

SKILL LEVEL REQUIRED

In general, PLCs support a range of programming technologies or methods (Mehta & Reddy, 2014:24). This increases the possibility that engineers are familiar or comfortable with performing PLC programming. Mehta & Reddy (2014:37) state that specific computer programming knowledge is not necessarily a requirement – PLC programming uses a simple and intuitive logic-based form of language (for example “ladder diagrams”). Because no high skill level is required (Mehta & Reddy, 2014:52), a score of 3/3 is awarded.

4.2.2. 2-FACTOR FEATURE SCORING

SCADA control (with PLC implementation) is the example of 2-factor control in this study. The case studies are from a mining group in South Africa that makes use of Wonderware System Platform (previously known as ArchestrA).

SIMULATION/TESTING ENVIRONMENT

The SCADA used in the case studies does not offer a built-in simulation capability. There is no means of simulating what the effect of control parameter changes would be. Thus, a score of 0/3 is awarded.

OPTIMISED PUMPING SCHEDULE

If the input parameters to the pump scheduling algorithm are fine-tuned, it is possible that this algorithm will output optimised pumping schedules. However, only *a number* of pumps to run will be recommended and not *which* pumps to run. This means that this control system is unable to balance the workload of the pumps. A score of 2/3 is awarded.

CONTROL AND AUTOMATED OPERATION

This control system is able to control the components of the pumping system. In emergencies, scheduling will continue, or if so preferred, pumps can be switched to local operation mode, ignoring scheduling instructions from the SCADA/PLC. Control room operators should be able to disable control via the SCADA. A score of 3/3 is awarded.

MONITORING AND REPORTING

The mine SCADA provides detailed monitoring capability of the process. It is not only possible to monitor process variables, but also information such as past start and stop times of each pump. The SCADA has a linked historian where vast amounts of data can be stored. Historical data is accessible using built-in trending functionality. A score of 3/3 is awarded.

ALARM HANDLING

The SCADA control system has better alarm management capabilities than the PLC implementation. In its current state, on-screen alarms can notify control room operators of any system parameters that need attention. Because this control is linked to PLC control, the benefits from PLC alarm handling are included. A score of 2/3 is awarded.

SKILL LEVEL REQUIRED

SCADA programming requires more programming knowledge than PLC's (compared to PLC that uses simplified methods). The mines' SCADA has a built-in script editor that makes programming easier (Figure 21). This script editor provides a user-friendly environment to create scripts using its QuickScript .NET programming language.

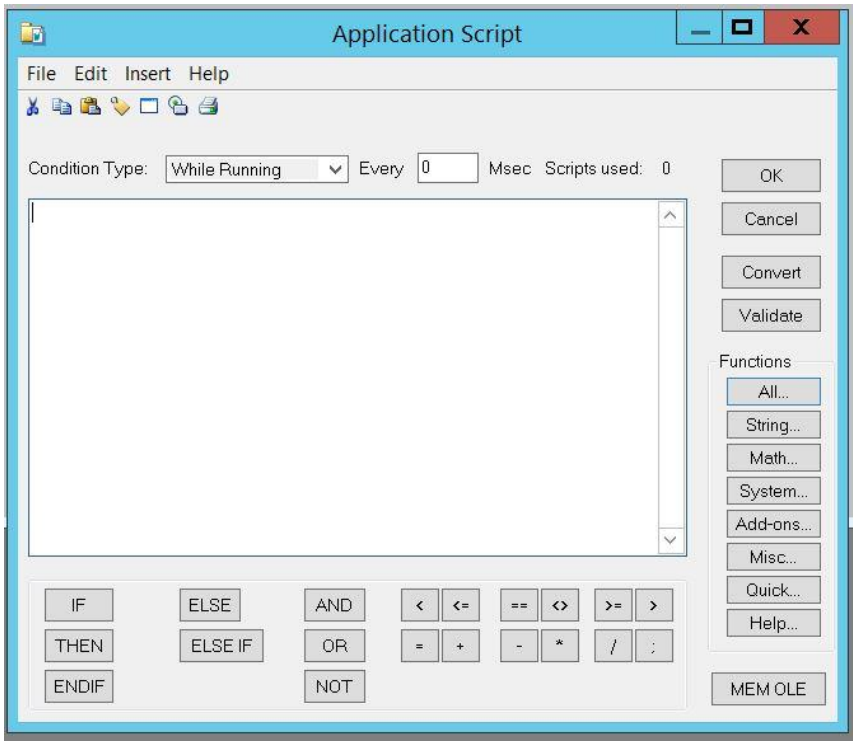


FIGURE 21 – SCADA SCRIPT EDITOR WITH CODE-FILLING BUTTONS (WONDERWARE, 2017)

The third-party control software used for the n-factor control in this study offers a slightly better script editor, because code snippets can be inserted in such a way that the user does not require specific programming knowledge.

In terms of skill level requirements, SCADA is inferior to PLC and third-party control software. A score of 1/3 is awarded.

4.2.3. n-FACTOR FEATURE SCORING

Third-party control software, Real-Time Energy Management System (REMS), is the example of n-factor control in this study. Reasons for making use of REMS include availability and knowledge of this software to the author.

SIMULATION/TESTING ENVIRONMENT

The third-party software has a built-in dynamic state simulation environment, capable of simulating water use and accumulation for mine dewatering systems. It can be used for management and simulation of dam levels. This provides the capability to test the effect of control parameters before deployment, possibly preventing erroneous changes. A score of 3/3 is awarded.

Optimised pumping schedule

This control system offers continuous optimisation of the pump schedule based on current system parameters. This control system also has a positive effect on the maintenance of pumps, because it automatically balances the workload of the pumps (Rautenbach, 2007). A score of 3/3 is awarded.

Control and automated operation

REMS is capable of automatically controlling components of the mine dewatering system by sending commands to the SCADA it is linked to. In the case of incorrect scheduling by REMS, or in emergencies, control via REMS can easily be disabled by control room operators. A score of 3/3 is awarded.

Monitoring and reporting

REMS provides SCADA-like process visualisation for use in control rooms. This enables operators to view process parameters and control the dewatering system. REMS provides extensive data logging capability. All parameters needed for control of the system are automatically logged. The control system provides trending capability for these parameters. A score of 3/3 is awarded.

Alarm handling

Similarly to the 2-factor control system, alarm handling for this control system also includes alarms based on the PLC-only control system. REMS additionally provides audible and visual alarms for use in the control room, as well as SMS and e-mail alarm capability. REMS provides more in terms of alarm handling capabilities than the SCADA in its current state. Thus, a score of 3/3 is awarded.

Skill level required

REMS provides the capability for the user to program custom tags, and thus customisable behaviour, into the control system. This is done using JavaScript. However, it is not necessarily required that the user be highly trained in using JavaScript, as REMS provides code snippets containing most of the code for just pasting into the programming environment (Figure 22). In any case, no programming is strictly required – building simulation/control models in REMS is done using built-in components. All code is encapsulated from the user and the user only needs to input control parameters. A score of 3/3 is thus awarded.

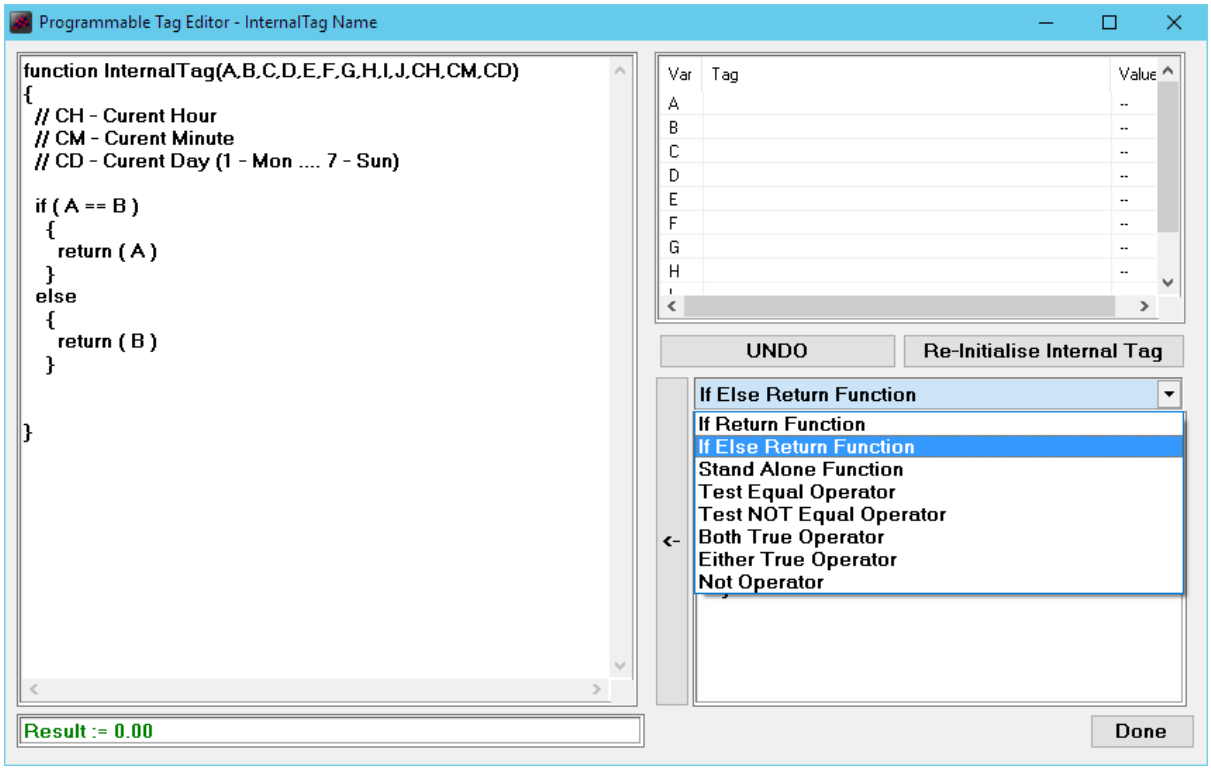


FIGURE 22 – REMS SCRIPT EDITOR WITH CODE SNIPPET FUNCTIONALITY (TEMM INTERNATIONAL (PTY) LTD, 2017)

4.2.4. TOTAL AND SUMMARISED FEATURE SCORES

For each of the control systems, features were evaluated and scored (Sections 4.2.1 to 4.2.3). Table 6 lists these scores in summarised tabular format. Each feature consists of subcomponents for evaluation or is evaluated relative to other control systems' features. Items listed in grey are the descriptions of the contributing factors adding up to the total score of each feature (underlined).

TABLE 6 – AWARDED CONTROL SYSTEM FEATURE SCORES

	1-factor	2-factor	n-factor
Simulation / testing environment	**0**	**0**	**3**
- Yes = 3.			
- No = 0.	0	0	3
Optimised pumping schedule	**2**	**2**	**3**
- No = 0.			
- Follow/recommend, leading to reduced electricity cost. If not, -1.			
- Workload balancing. If not, -1.	-1	-1	

Control and automated operation	**2**	**3**	**3**
- Able to control = 1.	1	1	1
- Human assistance not needed = 1.	0	1	1
- Emergency situation handling = 1.	1	1	1
Monitoring and reporting	**1**	**3**	**3**
- Monitoring = 1.	1	1	1
- Logging = 1.	0	1	1
- Reporting = 1.	0	1	1
Alarm handling	**1**	**2**	**3**
- Scoring relative to other control systems.			
Skill level required	**3**	**1**	**3**
- Scoring relative to other control systems.			
	$^{9}/_{18}$	**$^{11}/_{18}$**	**$^{18}/_{18}$**

4.3. CASE STUDY 1: MINE WITH ONE DEWATERING LEVEL

4.3.1. DESCRIPTION OF CASE STUDY AND SPECIFICS

A South African gold mine with one dewatering level provided Case study 1 (CS1).

At this mine, fissure water enters two dams at 43.5-level (43.5L) after having mud settled out by two settlers (Figure 23). 43.5L is located roughly 3.5 km below ground. Four pumps are installed at 44-level (44L), which pump water from the 43.5L dams to two dams at 30L. Two pumps can run simultaneously. The pumps installed on 44L have a rated power of 1 900 kW each.

From 30L, water is gravity-fed freely to another nearby shaft though underground interconnecting pipes. The dams installed at 44L and 30L have a capacity of 5 ML each. Both levels have two dams, but at each level, only one is used at a time.

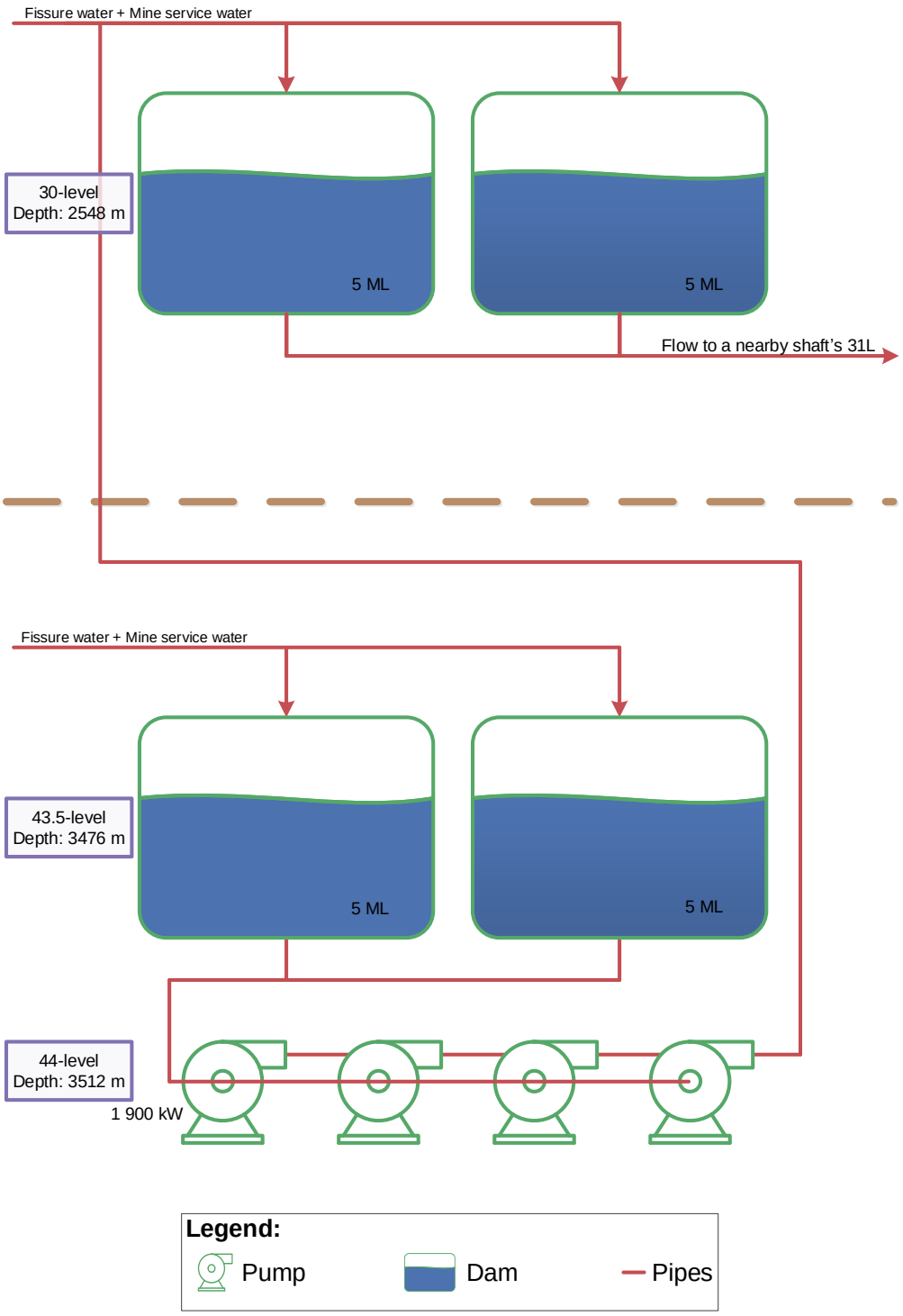


FIGURE 23 – CASE STUDY 1 (CS1) DEWATERING SYSTEM LAYOUT

Table 7 (on the next page) contains the information required to be able to simulate the dewatering system. This information answers the questions posed in Section 3.3 (page 30). Seeing as the water in the 30L dams is gravity-fed to another shaft, 30L does not need to form part of the simulation.

TABLE 7 – CASE STUDY 1 (CS1) SIMULATION INFORMATION

	44-level	30-level
Number of dams	2	2
Dam capacity	5 ML	5 ML
Dams interconnected	Possible. Only one used at a time.	Possible. Only one used at a time.
Additional inflow	Varies throughout the day	
Dam level limits (preferred)	Minimum: 30% Maximum: 80%	
Penalisation if dam(s) run empty/overflow	Yes	No
Dam penalisation limits	≤25%, ≥85%	N/A
Pumps installed	4	
Power consumption per pump	1 900 kW	
Flow per pump	143.0 L/s	
Maximum pumps running simultaneously	2	
Eskom tariff structure	Megaflex ≤300 km, 500 V – 66 kV	

4.3.2. SIMULATION RESULTS

Before control simulations can be run and conclusions made on the CS1 dewatering system, the integrity of the base case simulation for this system first needs to be determined. The base case simulation is a simulation of the system under the same control conditions as was followed for the actual day. The simulation is *not* controlled by the control systems of this study.

VALIDATION OF BASE CASE SIMULATION

One day is simulated for the CS1 dewatering system. For this day, mostly two pumps were running (Figure 24). A simulation was run using this exact number of pumps at the times that these pumps were running, hence the two lines overlapping. In the figure, green shaded areas represent Eskom off-peak periods, yellow areas represent standard periods, and red areas represent peak periods. The root mean square deviation (RMSD) is 0 pumps, which is to be expected.

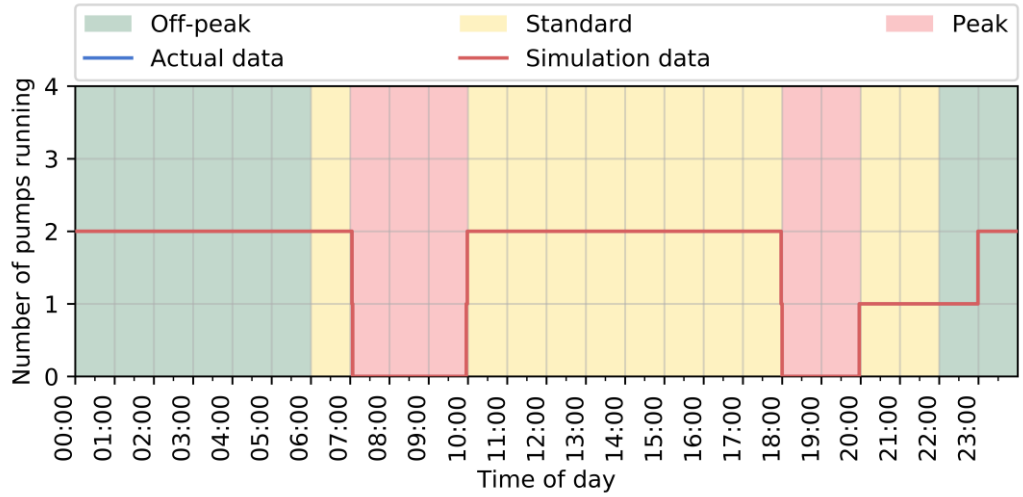


FIGURE 24 – CS1 SIMULATION VALIDATION: PUMP STATUSES

Fissure water inflow into 44L was calculated in 30-minute resolution. Using these values, along with the pump statuses from above and the starting dam level, dam level changes were simulated and compared to the actual dam levels for this day (Figure 25). It is seen that the simulated dam level matches the actual dam level quite closely. The RMSD is 1.25%.

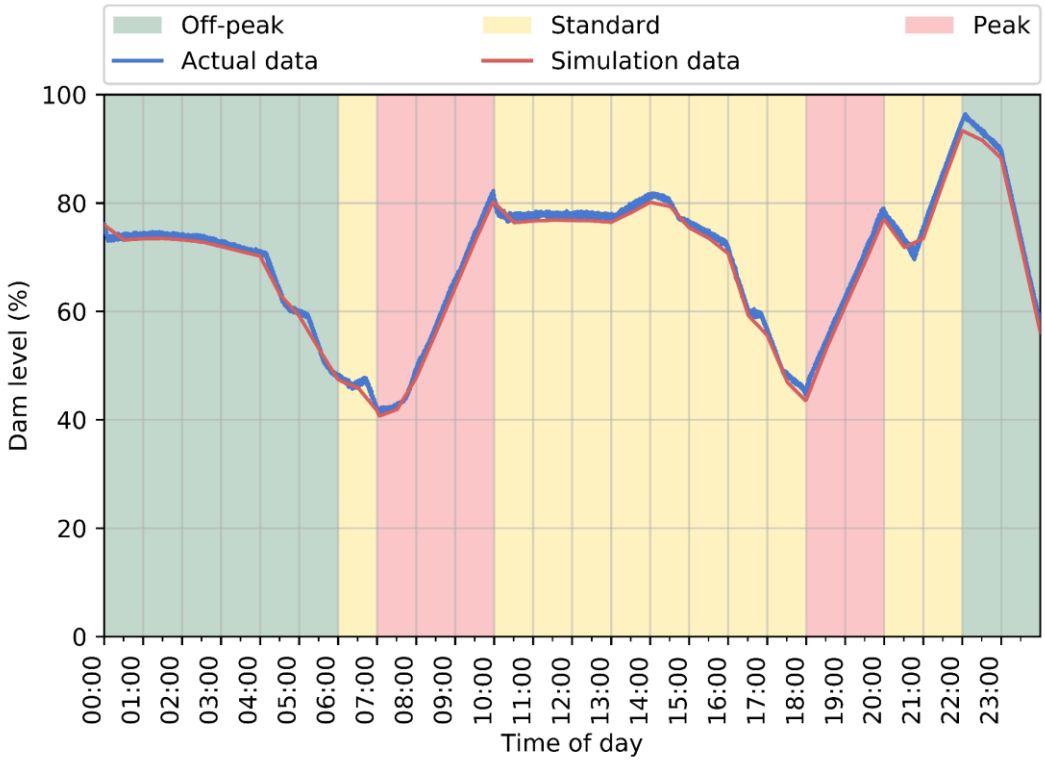


FIGURE 25 – CS1 SIMULATION VALIDATION: DAM LEVEL

Differences between simulated and actual dam levels, such as from 06:30 to 07:00 and 22:00 to 22:30, are because of abrupt changes in fissure water inflow to the dam which is not perfectly captured by the 30-minute resolution data. However, with the mean deviation of predicted values with respect to the observed ones of 1.25%, this is not of too much concern.

With the small differences between the simulated and actual values, the base case simulation can be deemed accurate enough to continue onto the next stages: inspecting how the different control systems will interact and attempt to control the process variables.

1-FACTOR CONTROL SIMULATION

Figures 26 and 27 graphically illustrate the results of the 1-factor simulation of CS1. Figure 26 indicates the 44L dam level as well as the total number of pumps running at any time. Figure 27 indicates the power consumption of the dewatering system pumps.

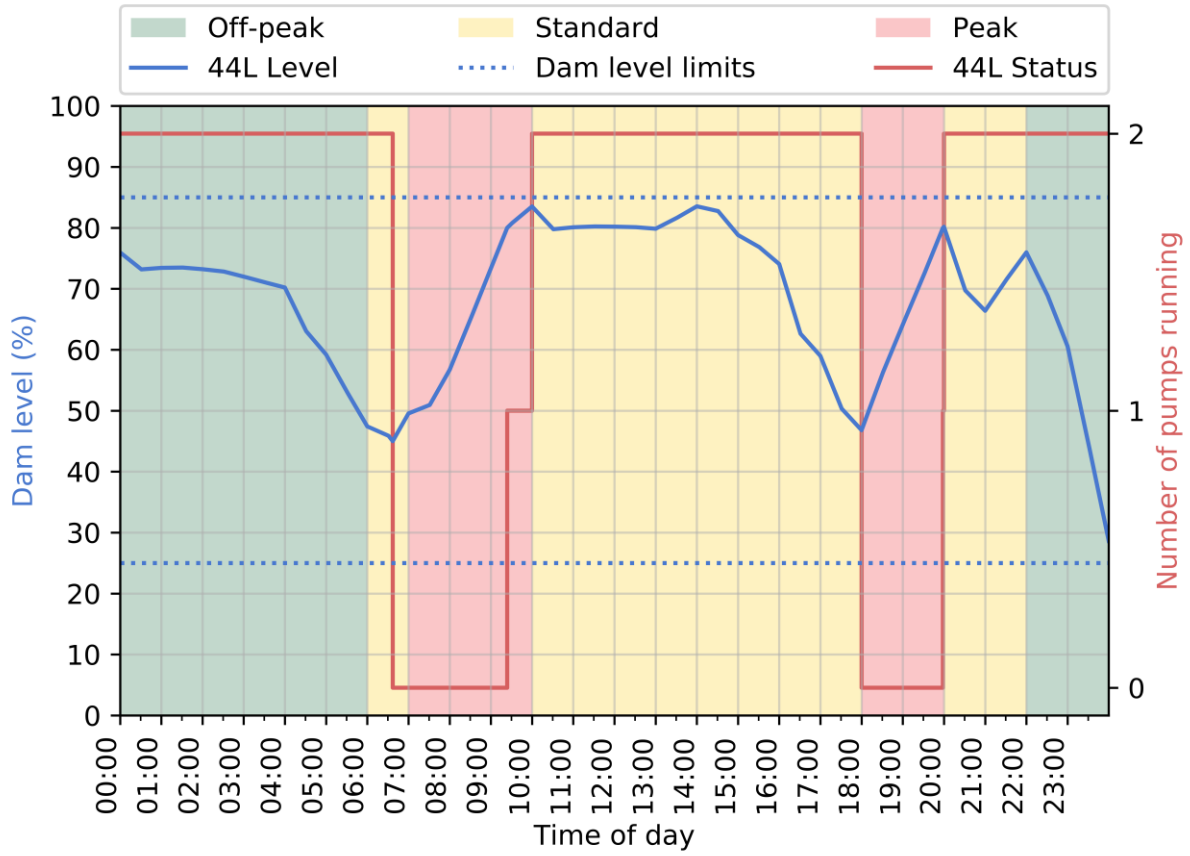


FIGURE 26 – CS1 1-FACTOR SIMULATION RESULTS: DAM LEVEL AND PUMP STATUSES

Figure 26 shows that mostly two pumps were running during Eskom non-peak hours. The control system was able to have zero pumps running for most of the duration of the Eskom peak hours. Just before 10:00, one pump was started as the dam level reached 80%. The fact that it was possible to switch off pumps during morning and evening peak hours means that the control system was able to perform morning and evening load-shifts. The effect of the load-shift is also visible in Figure 27, where power consumption decreased during peak hours.

From Figure 26, it is seen that pumps were controlled with no intermittent pump cycling. 44L's dam level stayed within control limits.

With the pumps running as shown in Figure 26, the power profile as indicated in Figure 27 is obtained for the CS1 dewatering system. This power profile is in one-second resolution. This data is resampled to 30-minute resolution, as per Eskom M&V practices (Figure 28). The 30-minute data points are represented by the blue cross markers. In Figure 28, the 30-minute data can be illustrated by drawing straight or stepped lines between the data points. From this point forward, power profiles in this study will be shown in 30-minute intervals only, with stepped lines between the data points.

Using the Eskom Megaflex ≤300 km, 500 V – 66 kV tariff, the “best profile cost” (BPC) for the operation of this simulated day is R30 579.18. The “simulated profile cost” (SPC) for this day is R37 748.91.

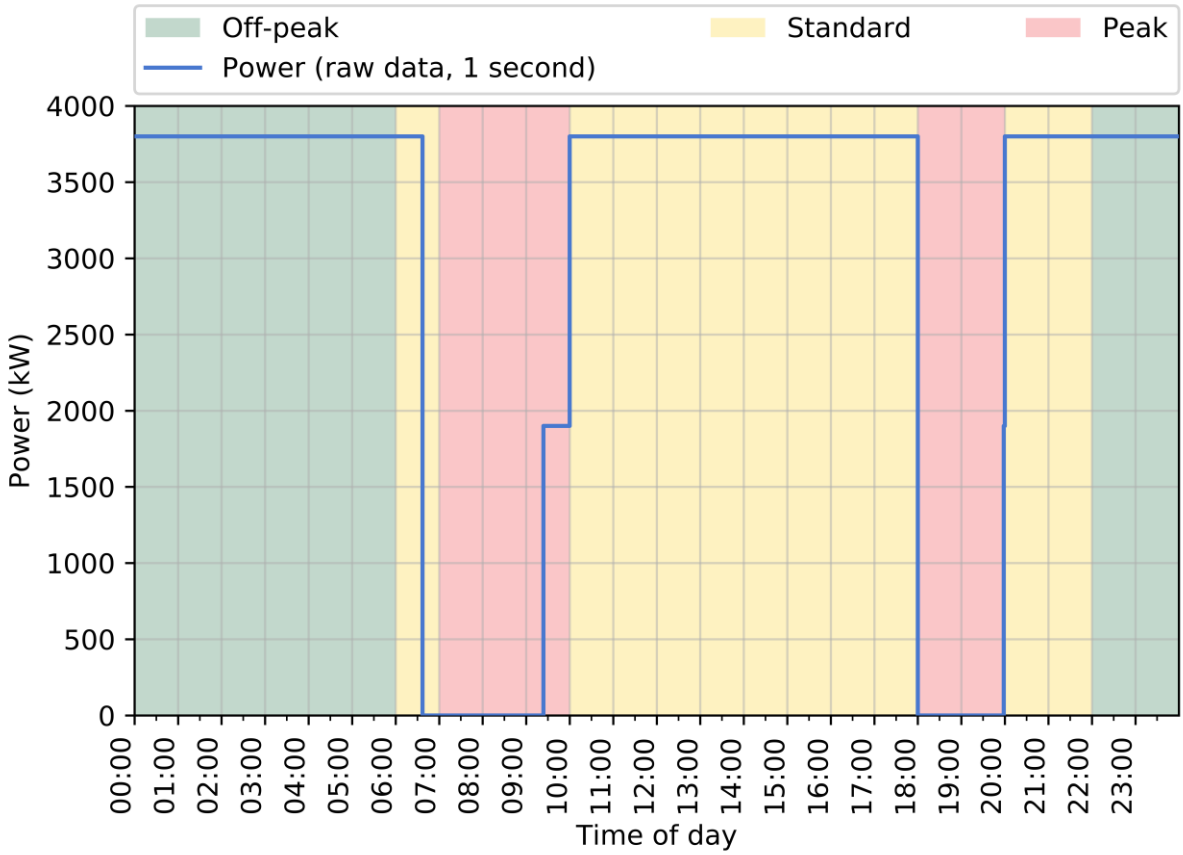


FIGURE 27 – CS1 1-FACTOR SIMULATION RESULTS: POWER CONSUMPTION (RAW DATA)

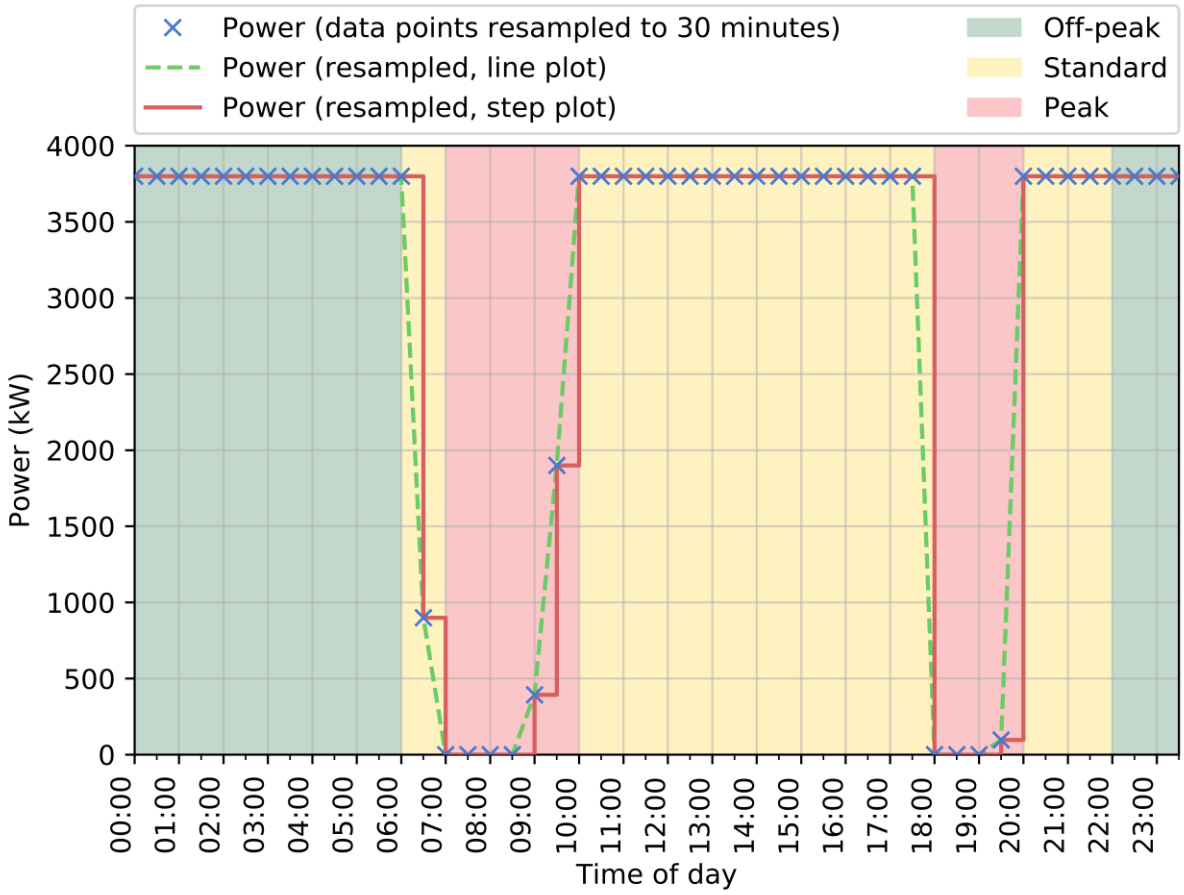


FIGURE 28 – CS1 1-FACTOR SIMULATION RESULTS: POWER CONSUMPTION (RESAMPLED DATA)

2-FACTOR CONTROL SIMULATION

The results of the 2-factor simulation for the CS1 system is shown in Figures 29 and 30.

Figure 29 indicates that performing 2-factor control on the CS1 system results in dam levels staying within control limits. No excessive pump cycling occurred. The control system was able to perform load-shift on the dewatering system. Figure 30 displays the power profile accompanying the pump statuses as in Figure 29.

The 1-factor and 2-factor results on the CS1 system are exactly the same (comparing Figures 26 and 29, and Figures 28 and 30). This is because CS1 only has one pumping level and the only difference between the 1-factor and 2-factor control systems is the logic related to the handling of downstream dams (as seen in the decision-making algorithm, Figure 13, page 36). The simulation did not require the inclusion of dams downstream from 44L.

Using the Eskom Megaflex ≤300 km, 500 V – 66 kV tariff, the BPC for the operation of this day is R30 579.18, whereas the SPC is R37 748.91.

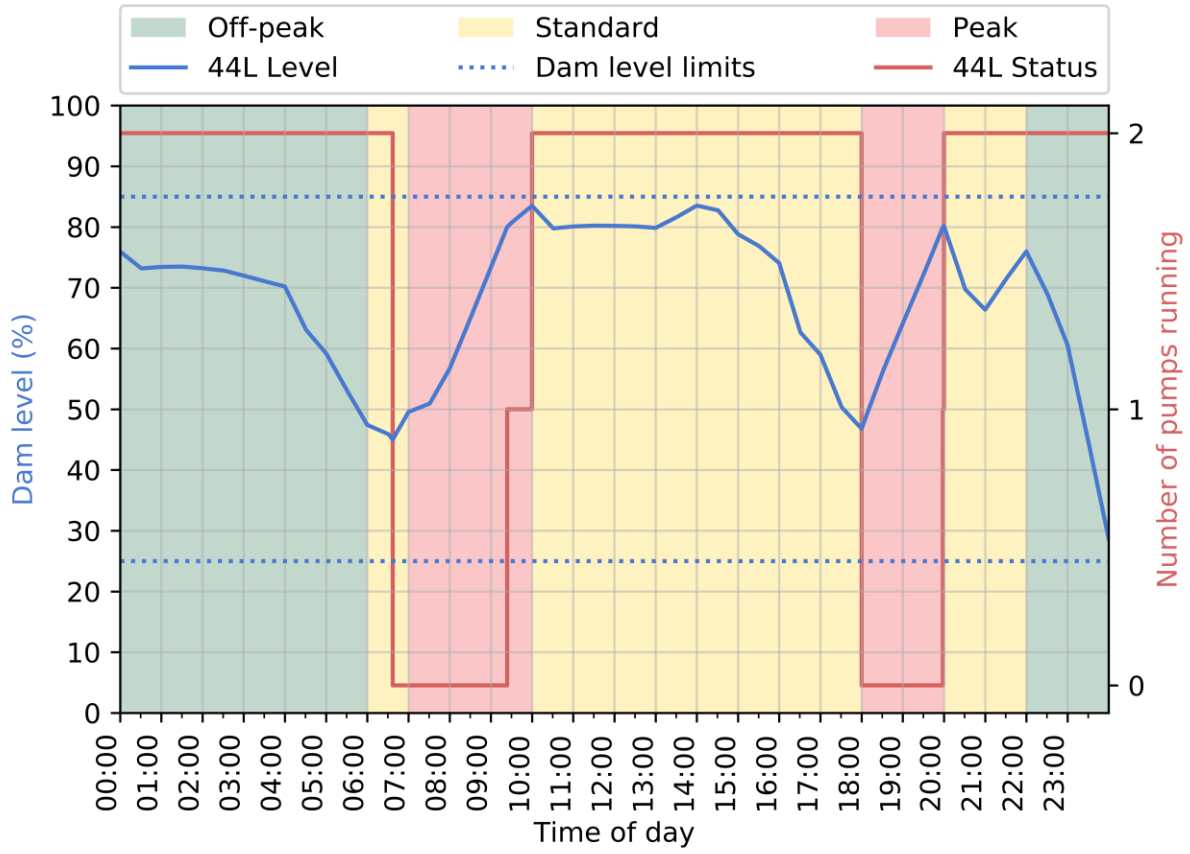


FIGURE 29 – CS1 2-FACTOR SIMULATION RESULTS: DAM LEVELS AND PUMP STATUSES

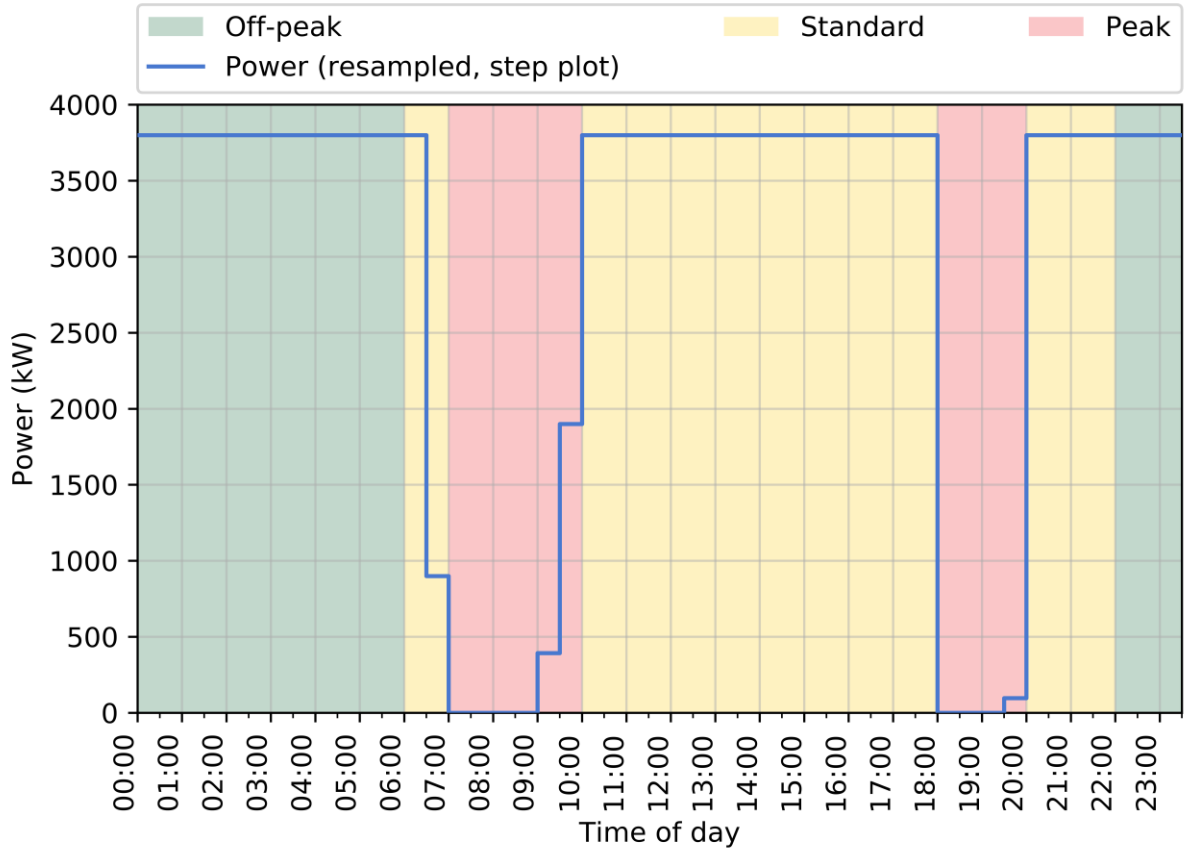


FIGURE 30 – CS1 2-FACTOR SIMULATION RESULTS: POWER CONSUMPTION

Even though the CS1 system is uncomplicated, an n-factor control approach can still be applied to this system. An n-factor simulation was performed on the CS1 dewatering system and it was found that this approach was able to perform a marginally better load-shift than the other control systems, while keeping dam levels closer to the control range (Figure 31).

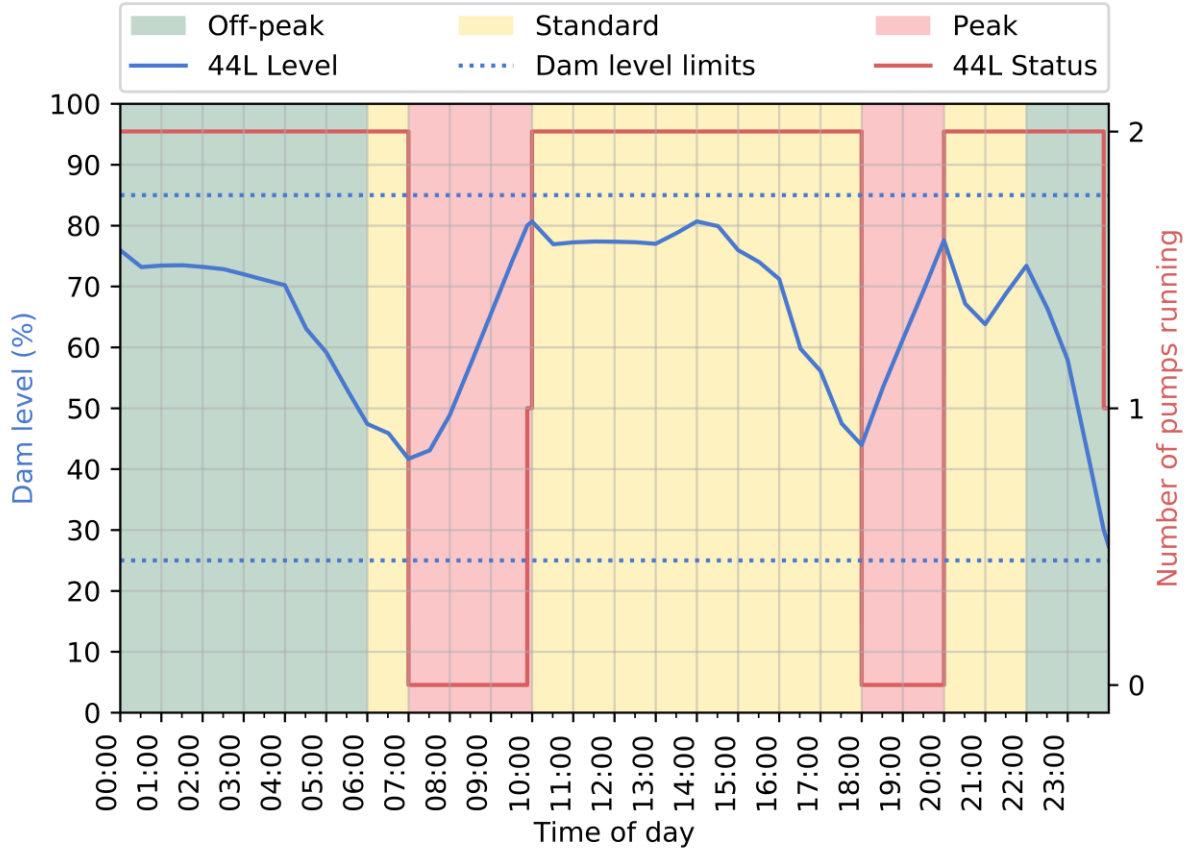


FIGURE 31 – CS1 n-FACTOR SIMULATION RESULTS: DAM LEVELS AND PUMP STATUSES

Running the pumps as shown in Figure 31 results in a total power consumption for CS1 as indicated in Figure 32. A perfect evening load-shift was performed and a morning load-shift was near-perfect. This is because of the dam level reaching maximum capacity, triggering the need to switch on a pump.

Using the Eskom Megaflex ≤300 km, 500 V – 66 kV tariff, the BPC is R30 730.13 and the SPC is R37 681.08.

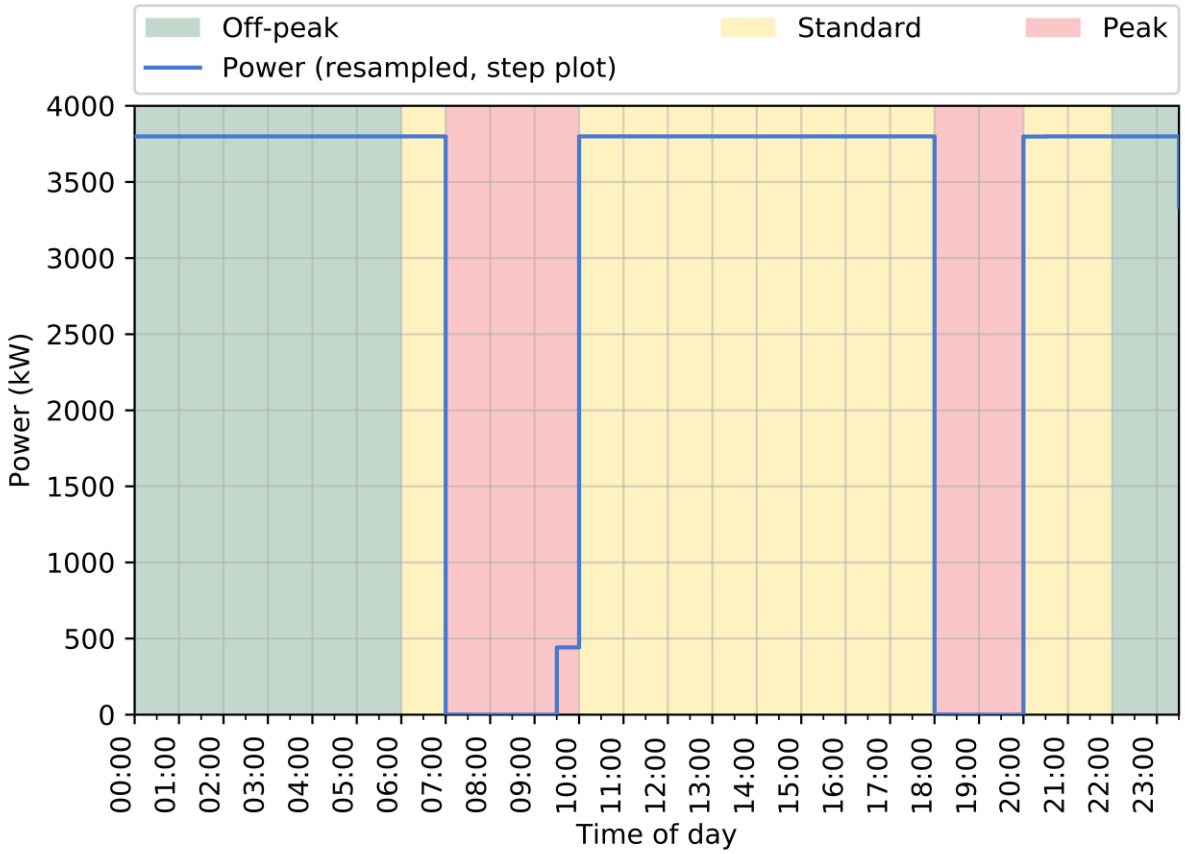


FIGURE 32 – CS1 n-FACTOR SIMULATION RESULTS: POWER CONSUMPTION

4.3.3. SCORING

As explained in Section 3.7, a score is calculated per case study, per control system. This collection of scores will facilitate comparison between the control systems. A full explanation is only given for CS1's 1-factor scoring, since all subsequent scoring works the same, even at other case studies.

1-FACTOR

PERFORMANCE SCORE

Cost score:

The cost score contribution is R30 579.18 ÷ R37 748.91 ≈ 0.8101.

Level score:

The 44L dam level stayed within penalisation limits. Thus, a score of 1 is awarded.

Pump cycling score:

Controlling the pumps according to the 1-factor philosophy did not lead to the cycling of pumps. A score of 1 is awarded.

Total performance score:

The three scores calculated above are now combined into the total performance score:

$$\text{Performance score} = \frac{\text{Cost score} \times w_1 + \text{Level score} \times w_2 + \text{Pump cycling score} \times w_3}{w_1 + w_2 + w_3} \quad (19)$$

Dams reaching critical levels or control systems causing unnecessary wear on pumps (in the form of unpreventable pump cycling) should be heavily penalised. These factors are more important than whether or not the control system is able to shift load from peak periods. For this reason, the weights are chosen as $w_1 = 1$, $w_2 = 2$, and $w_3 = 2$. This means that the level score and pump cycling score carry more weight, leading to more pronounced effects on the total performance score.

The total performance score for CS1 using the 1-factor control system is thus 0.9620.

Feature score

As calculated in Section 4.2.1 (Table 6, page 61), the feature score for CS1 is 0.5.

Combined total score

The total score for this case study is calculated by combining the simulation and feature scores using Equation (21):

$$\text{Total score} = 100 \times \frac{\text{Performance score} \times w_4 + \text{Feature score} \times w_5}{w_4 + w_5} \quad (21)$$

Weights are chosen as $w_4 = 3$ and $w_5 = 6$. Whether the control system actually works is deemed more important than the additional features it offers. For this reason, it is preferred that the performance score carry more weight. Using these weights, the total score for the 1-factor control system for CS1 works out as 80.80. Refer to Figure 33 (page 73) for a comparison with the other control systems.

2-FACTOR AND n-FACTOR

Following the same process as described from page 71, the following scores are calculated for the 2-factor and n-factor control systems for CS1 (Table 8). The previously-calculated scores for the 1-factor control system are also listed for ease of access.

TABLE 8 – CS1 SCORING RESULTS

	1-factor	2-factor	n-factor
Performance score	0.9620	0.9620	0.9631
Feature score	0.5000	0.61̇	1.0000
Total score	80.80	84.50	97.54

A graphical representation of the scores as in Table 8 allows for easy interpretation of the results. Figure 33 puts the simulation and feature scores' separate contribution towards the total score into perspective.

FIGURE 33 – CS1 TOTAL SCORES

From Table 8 and Figure 33, it is clear that the three control systems fared virtually the same in terms of their performance or capability to control the CS1 dewatering system. There is a more pronounced difference in the control systems' feature scores, which result in causing n-factor control to be the best-faring control option at CS1.

4.4. Case study 2: Mine with two dewatering levels

The main difference between Case study 2 and Case study 1, is the amount of dewatering levels present. Therefore, the description and simulation results are only summarised for Case study 2 in this section. Detail is available in Appendix C (page 99).

4.4.1. Description (summary)

Case study 2 (CS2) has two pumping levels, each with two 3 ML dams. Each level has fissure water inflow that varies in flow rate throughout the day. In total, the dewatering system consists of nine pumps, of which a maximum of three can run simultaneously per level. Water pumped to the surface accumulates in dams which are designed to overflow, and these dams need not be included in the simulation.

4.4.2. Simulation results (summary)

Validation of base case simulation

The base case simulation followed the actual pump running schedules and dam level with great accuracy. The average RMSD for the dam level comparison of 0.700% supports this. The base case simulation is deemed valid and further inferences can be made.

1-factor control simulation

The control system was able to control the dam level between its minimum and maximum allowable values, but with some amount of unnecessary pump cycling occurring. The BPC for the operation of this simulated day is R91 878.95. The SPC for this day is R121 942.09.

2-factor control simulation

Results are identical to the 1-factor simulation.

n-factor control simulation

The control system was able to control the dam level between its minimum and maximum allowable values, without preventable pump cycling occurring. The BPC for the operation of this day is R91 171.81, whereas the SPC is R121 379.67.

4.4.3. SCORING

Following the same process as described in Section 4.3.3, the scoring results for CS2 is as follows:

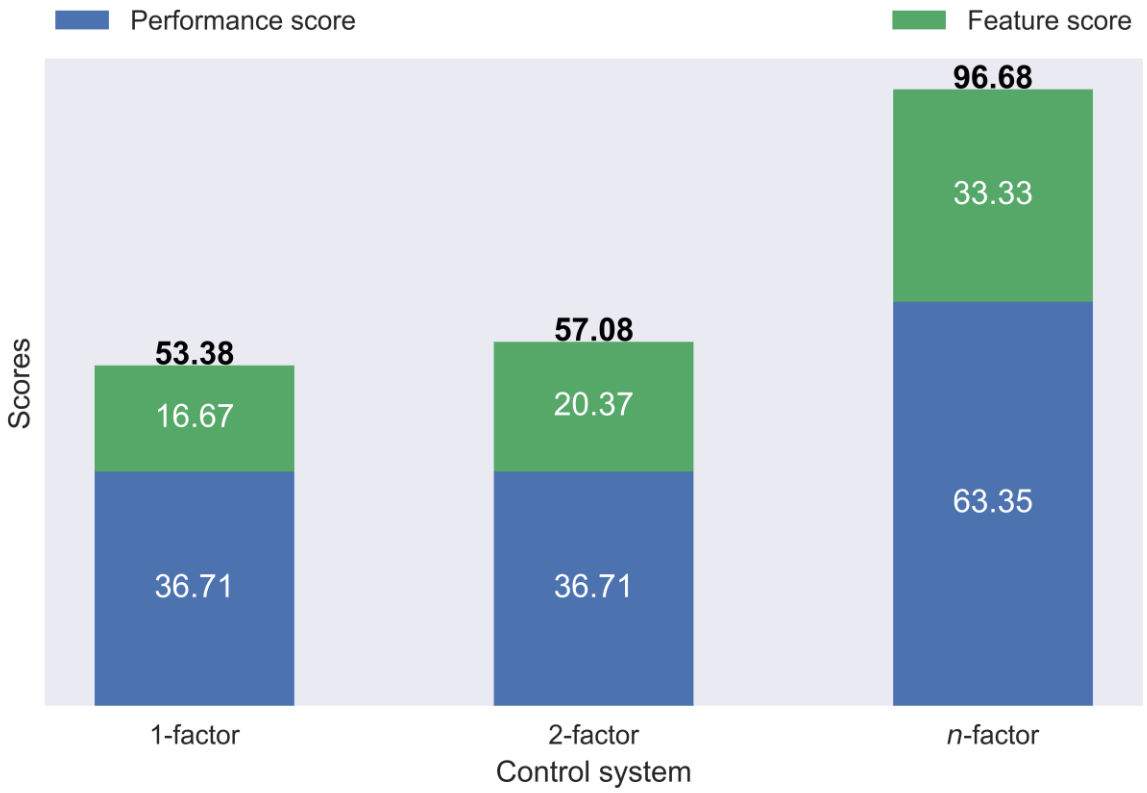


FIGURE 34 – CS2 TOTAL SCORES

It is seen from Figure 34 that performing 1-factor and 2-factor control for the CS2 dewatering system results in identical performance. n-factor control has a much higher performance score.

Including the contributions from the feature scores, 2-factor control fares better than 1-factor control for Case study 2. n-factor control fares better than 2-factor control.

4.5. Case study 3: Mine with four dewatering levels

The main difference between Case study 3 and the previous case studies, is the amount of dewatering levels present. Therefore, the description and simulation results are only summarised for Case study 3 in this section. Detail is available in Appendix D (page 109).

4.5.1. Description (summary)

Case study 3 (CS3) has four pumping levels, each with one 5 ML dam usable for water storage. 18 pumps work together to pump fissure water out to surface, where overflowing of the surface dam should be avoided. Fissure water flow rates vary throughout the day.

4.5.2. Simulation results (summary)

Validation of base case simulation

The base case simulation followed the actual pump running schedules. For some of the dewatering levels, the simulated dam level was higher than the actual dam level. This is due to the erratic inflow of water into the dams, which is not perfectly captured by the 30-minute resolution used for calculations. With the simulated dam levels being higher than the actual dam levels, this means that the assumptions create a model which errs on the conservative side. Nevertheless, CS3 dewatering system was simulated accurately and is deemed valid, with an average RMSD value for dam levels of 2.670%.

1-factor control simulation

Pump cycling occurred on all dewatering levels. One dewatering level's dam as well as the surface dam overflowed. The remaining three dewatering levels' dams was controlled within their control ranges. The BPC for the operation of this day is R185 691.93. The SPC for this day is R258 259.71.

2-factor control simulation

Control-system induced pump cycling occurred at one of the dewatering levels. All dams were controlled within their control limits. The BPC for the operation of this day is R185 691.93, whereas the SPC is R257 700.40.

No pump cycling occurred. The surface dam overflowed as a strategic choice to remove water from underground and reduce the risk at the more sensitive underground levels. The BPC for the operation of this day is R192 497.32, whereas the SPC is R270 042.86.

4.5.3. SCORING

Scoring of the CS3 simulation results follows the same process as described in Section 4.3.3. Because dams overflowed at CS3, there are some slight differences in the performance score:

Cost score:

1-factor: The cost score contribution is R185 691.93 ÷ R258 259.71 ≈ 0.7190.

2-factor: The cost score contribution is R185 691.93 ÷ R257 700.40 ≈ 0.7206.

n-factor: The cost score contribution is R192 497.32 ÷ R270 042.86 ≈ 0.7128.

Level score:

1-factor: 2 out of 5 dams exceeded penalisation limits: $1 - 2/5 = 0.6$.

2-factor: 1 out of 5 dams exceeded penalisation limits: $1 - 1/5 = 0.8$.

n-factor: 2 out of 5 dams exceeded penalisation limits: $1 - 2/5 = 0.6$.

Pump cycling score:

1-factor: Pump cycling occurred; a score of 0 is awarded.

2-factor: Pump cycling occurred; a score of 0 is awarded.

n-factor: No pump cycling occurred; a score of 1 is awarded.

Total performance score:

1-factor: 0.3838

2-factor: 0.4641

n-factor: 0.7826

* * *

The combined total scores for the CS3 dewatering systems are as follows:

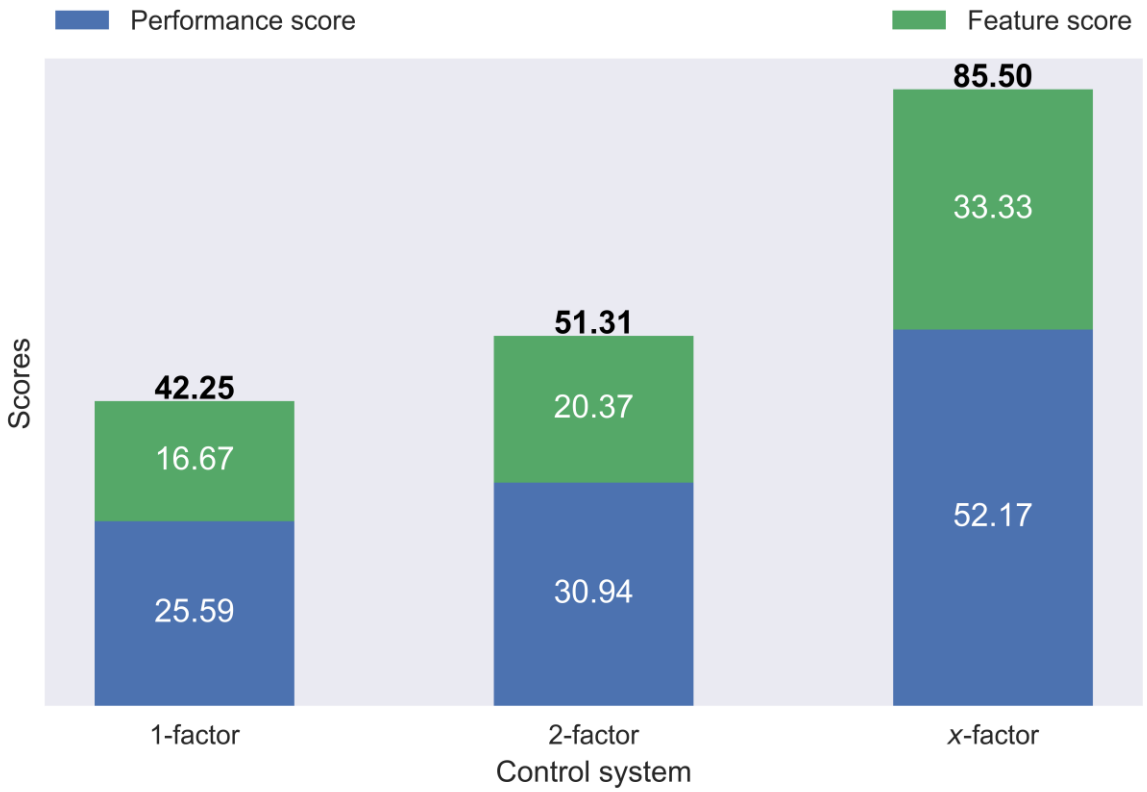


FIGURE 35 – CS3 TOTAL SCORES

It is clear from Figure 35 that control systems applied to the CS3 dewatering system rank as follows (in terms of performance, feature and combined scores): 1-factor $<$ 2-factor $<$ n-factor.

4.6. Discussion of results

The control system scores as obtained from evaluation for the three case studies (Sections 4.3.3, 4.4.3 and 4.5.3) are combined and repeated in Figure 36:

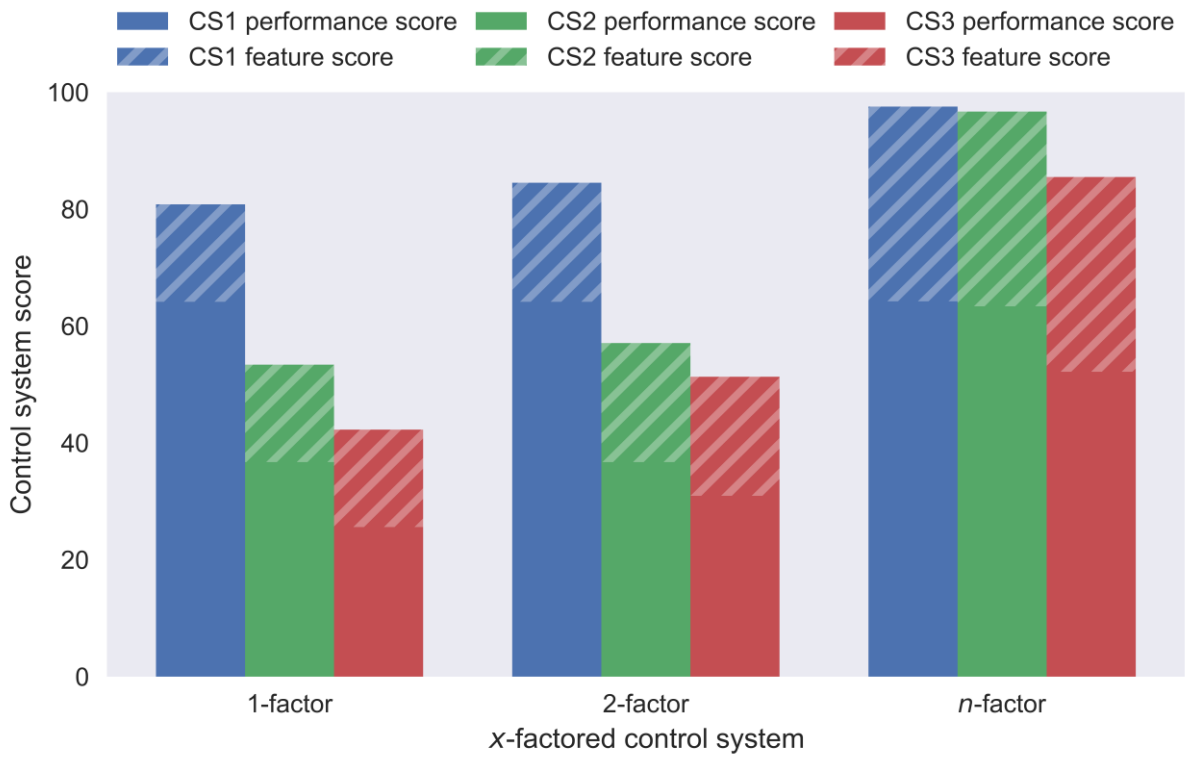


Figure 36 – Summary of control system scores

Case study 1 (CS1) has only one pumping level and the downstream dam is not of concern. This dewatering system is uncomplicated. Referring to Figure 33 (or Figure 36), it was found that the three considered control systems performed virtually identically. Eventually, what sets the control systems apart for a simple dewatering system such as CS1, are the features of the control system. 2-factor control (i.e. the mine SCADA control with PLC implementation) proved to offer more in terms of features compared to 1-factor control (PLC-only implementation). n-factor control (i.e. third-party control software in the form of REMS) offered better yet. All considered, n-factor control offers a better choice than 2-factor control, and 2-factor better than 1-factor. For a simple dewatering system such as CS1, the requirements of the instrumentation/control engineer will eventually be the deciding factor in the choice of control system to implement. The implementation and upkeep costs for the control system will also play a more important role (though this falls out of scope for the current study).

Case study 2 (CS2) has two pumping levels of concern. This dewatering system is more complex than the dewatering system at CS1. Referring to Figure 34 (or Figure 36), the

difference between the performance of the control systems is more pronounced. In terms of performance, 1-factor and 2-factor control are identical, because the level of the dam downstream from the bottom pumping level did not reach near-full values. n-factor control offers superior performance. The 1-factor and 2-factor control suffered in their performance scores, because of unnecessary pump cycling induced by the control algorithm. n-factor control proves to be the best choice.

Case study 3 (CS3) has five levels of concern, four of which has pumps forming part of this complex mine dewatering system. Referring to Figure 35 (or Figure 36), one notes that using 2-factor control instead of 1-factor control should offer better control of the dewatering system. Using n-factor, however, control offers a more significant advantage. Even though the surface dam overflowed (Figure 71), n-factor control performed the best in terms of removing amounts of water from the mine and returning operation of the dewatering system back to normal. Risk of underground flooding is lowered by using n-factor control. The 1-factor and 2-factor control again suffered in their performance scores, because of unnecessary pump cycling induced by the control algorithm. n-factor control proves to be the best choice considering actual performance, features, and all combined.

It is noted that for dewatering systems of lower complexity, the choice of the control system to use depends mainly on the features required. Differences in control system performance are negligible. However, as the complexity of the dewatering system increases, so does the need for a control system offering more complexity in the control algorithm. Differently put, control of complex dewatering systems can be better optimised using control systems offering the capability of incorporating more complex/site-specific logic in the control algorithm. Always using a generic approach, as with the mine SCADA algorithm (Figure 13, page 36), is not necessarily the optimal choice.

For the mining group providing the case studies, n-factor control (in the form of REMS) showed to be the optimal choice of control system to use. This is followed by 2-factor control (in the form of mine SCADA) and then 1-factor (mine SCADA algorithm implemented in PLCs only).

It can be extrapolated that, for mine dewatering systems more complex than Case study 3, the difference in control system performance would be even more pronounced. This will mean that the use of n-factor control (REMS) will be even more essential. More complex dewatering systems can, for example, consist of more dewatering levels or contain energy recovery devices, such as three-chamber pump systems.

The 1-factor and 2-factor control algorithm is likely to cause unnecessary cycling of pumps. This is due to its handling of the case when “two too many pumps are running”. Another

contributing factor is the behaviour when dam levels reach the lower control limit: dam levels will then be controlled between this dam level (%) and 5% below that. This default value of 5% is likely to cause pump cycling because of this narrow control range of 5%.

4.7. FURTHER VALIDATION OF THE STUDY

4.7.1. INTERNAL VALIDATION

The aim of the study is to develop and use a method by means of which mine dewatering control systems can be compared. A method was developed and it revolves around a comparison of a control system score. This score consists of the following contributors and sub-contributors:

- **Control system performance score**
  - Load-shift performance score, determined using electrical energy cost for the day's operation.
  - Pump cycling score, determined by the frequency of pump starts.
  - Dam level score, determined by the Boolean condition "if the dam level is within control limits".
- **Control system feature score**, consisting of a list of features advantageous to the control of mine dewatering systems.

The study is considered (internally) valid, because:

1. **The study is content and construct valid** (i.e. the control system performance, or total score, (the "construct") is measured by the elements as described above, and the measurement procedure is valid):
   - Electrical energy cost accurately describes the load-shift performance of the control system. ToU tariffs are what drive load-shifting. Using energy in non-peak instead of peak periods (i.e. load-shifting) means that energy is used in less expensive time slots. This automatically leads to a reduced energy cost. Energy cost thus describes load-shift performance.
   - Too frequent pump starts are referred to as pump cycling (Section 2.3.2, page 17).
   - The dam level score is determined as the amount of dams that was controlled within limits divided by the total number of dams.
   - The control system features were identified, and identified as relevant, from personal industry experience and literature (Section 2.4.4, page 20).

2. **The aim was achieved**:

   This method was followed and applied to three case studies. These case studies' dewatering systems differ in their level of complexity. For each case study, a performance score and feature score was obtained and combined into a total score. These scores facilitated a comparison between the different control systems.

4.7.2. EXTERNAL VALIDATION

The **results** from this study (Section 4.6) are only directly applicable or relatable to dewatering systems using the same control algorithms as considered in this study. This is because the 1-factor and 2-factor pump scheduling/control algorithm used in this study is from a specific gold mining group in South Africa, and n-factor using Real-Time Energy Management System (REMS).

The **methodology** developed and used in this study (combined performance and feature scores) can be applied for comparison of mine dewatering control systems making use of any pump scheduling algorithm.

The methodology can also be applied to processes that are *not* mine dewatering systems – **any process with at least one dam and pump** can be analysed.

4.8. CONCLUSION

In this chapter, the models for simulating pump dewatering control systems as x-factored control systems were implemented on three case studies, after successful accuracy validation. These case studies differ in their level of complexity.

It was found that the lower the level of complexity of the dewatering system, the lower the level of control system complexity required to optimally control the dewatering system (in terms of load shifted and control parameters staying within control ranges). For dewatering systems of low complexity, higher complexity control systems are not strictly required, but offer the advantage of more advanced features which aid in the implementation and use of the control system. As the complexity of the dewatering system increases, so does the need for a more complex control system (or a control system offering the opportunity to incorporate more complex logic in the control algorithm).

It was also found that the current implementation of the mine SCADA (2-factor) control algorithm is flawed in its handling of switching pumps off when "two too many pumps" are

running and that the default control range at the lower control limit can potentially lead to the unnecessary cycling of pumps.

In Chapter 4, further validation was performed to assess the validity of the study itself. Validation was performed by evaluating the methodology and whether the study's aim was achieved.

COMPARATIVE STUDY OF MINE
DEWATERING CONTROL SYSTEMS

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS FOR FURTHER STUDY

5. CONCLUSION AND RECOMMENDATIONS FOR FURTHER STUDY

5.1. RESEARCH CONCLUSION

In Chapter 1, the dewatering system of deep-level mines was briefly discussed. It was explained how water is used and that it needs to be removed from underground. This is done by using a series of pumps and dams (the dewatering system of a mine). It is advantageous to automatically control the dewatering system of a mine. This is done by making use of the following control systems (listed in order of increasing capability of offering more complex control logic):

- programmable logic controllers (PLCs),
- supervisory control and data acquisition (SCADA) systems, or
- third-party control software interfacing with the SCADA.

Based on personal industry experience, automation engineers are often biased or have preferences towards the use of certain control systems, but these preferences are not always based on comparative knowledge of all three control systems. A perusal of literature revealed no literature wherein the above-mentioned control systems applied to mine dewatering process control have been compared experimentally. This study, however, compared the control systems experimentally.

The aim and objectives of this study were stated in Chapter 1, and can be summarised as: **Aim**: develop and use a method by means of which mine dewatering control systems can be compared. The aim is achieved by means of the following **objectives**:

1. Compare the three control systems by means of:
    a. Evaluation and comparison of control system features.
    b. Comparison of the performance of the control systems.
2. Draw conclusions as to which control system is best-suited for which level of dewatering system complexity.

Theoretical information adding to the background of the study was discussed in Chapter 2. Previous studies were also analysed to determine their shortfalls and applicability to this study.

Chapter 3 was dedicated towards development of a methodology by means of which the control systems can be compared. The methodology consists of two main contributors: a feature score and a performance score. The feature score represents the control system's

accompanying features that are beneficial towards the automated control of a dewatering system. These features were identified and described in Section 2.4.4. The performance score represents the control system's ability to control the dewatering system within parameter control ranges and in a manner leading to reduced electrical running costs.

The developed methodology was applied to three case studies (Chapter 4). The three case studies are provided by a group of deep-level gold mines in South Africa. Case study 1, 2, and 3's dewatering systems differ in terms of complexity. Case study 1's dewatering system is the simplest of the three case studies. Case study 3's dewatering system is the most complex.

From the case study results, it was found that a higher level of dewatering system complexity requires a control system capable of incorporating a more complex control algorithm to be able to optimally control the dewatering system. For dewatering systems of low complexity, higher complexity control systems are not strictly required, but offer the advantage of more advanced features which aid in the implementation and use of the control system. As the complexity of the dewatering system increases, so does the need for a more complex control system. Stated otherwise, it was found that control of complex dewatering systems can be better optimised using control systems offering the capability of incorporating more complex/site-specific logic in the control algorithm.

It can be extrapolated that for mine dewatering systems more complex than Case study 3, the difference in control system performance would be even more pronounced. This will mean that the use of n-factor control will be even more essential. More complex dewatering systems can, for example, consist of more dewatering levels or contain energy recovery devices, such as three-chamber pump systems.

Specific to the mining group providing the case studies, n-factor control (in the form of Real-Time Energy Management System, REMS) showed to be the optimal choice of control system to use. This is followed by 2-factor control (in the form of mine SCADA) and then 1-factor (mine SCADA algorithm implemented in PLCs only).

Furthermore, it was found that the 1-factor and 2-factor control algorithm as developed by the case studies' mine is likely to cause unnecessary cycling of pumps. This leads to increased frequency of pump maintenance (Zhuan & Xia, 2013) and preventable expenditure.

Evaluation of the study revealed that:

- The study objectives and aim have been met.
- It was determined that the results of the study are only directly applicable or relatable to dewatering systems making use of the same control algorithms.
- The methodology developed is applicable to not only other mine dewatering systems, but also any process consisting of at least one dam and at least one pump pumping a liquid from this dam.

5.2. RECOMMENDATIONS FOR FURTHER STUDY

The assumptions made in the study placed certain limitations on the investigation. These limitations provide the opportunity for research further investigating solution to the problem statement of this or related studies:

- The data calculation/polling rate was assumed as being one second. Further studies can investigate the effect of adjusting this value.
- Only one day was considered per case study for evaluation of the control systems. Further studies can investigate considering more days for comparison. Care should be given to not make use of average values as inputs to the simulations, except if it is determined that these average values are representative of the normal daily operation of the system.

Furthermore, future studies can address items listed in the scope of the investigation (Section 1.3, page 10):

- Variable speed pumps (VSD-driven) can be investigated.
- Additional dewatering system components, such as three-chamber pump systems and U-tubes, can be investigated.
- The effect of pump non-availability can be investigated.
- Implementation and upkeep costs of the control systems can be included in the analysis and comparison.

COMPARATIVE STUDY OF MINE
DEWATERING CONTROL SYSTEMS

REFERENCES

6. REFERENCES

Aydogmus, Z. 2009. Implementation of a fuzzy-based level control using SCADA. *Expert systems with applications*, 36(3 part 2):6593–6597. DOI: 10.1016/j.eswa.2008.07.055.

Behandish, M. & Wu, Z. Y. 2014. Concurrent pump scheduling and storage level optimization using meta-models and evolutionary algorithms. *Procedia engineering*, 70:103–112. DOI: 10.1016/j.proeng.2014.02.013.

Bene, J. G. 2013. Pump schedule optimisation techniques for water distribution systems. Finland: University of Oulu. (Dissertation — PhD).

Botha, A. 2010. Optimising the demand of a mine water reticulation system to reduce electricity consumption. Potchefstroom: NWU. (Dissertation — MEng).

Boyer, S. A. 2002. SCADA — Supervisory control and data acquisition. (*In* Lipták, B. G., *ed.* Instrument engineers' handbook: process software and digital networks. 3rd ed. Boca Raton, FL: CRC Press p. 357–367).

Boyer, S. A. 2004. SCADA: supervisory control and data acquisition. 3rd ed. Durham, NC: The Instrumentation, Systems, and Automation Society.

Breytenbach, W. J. J. 2014. Integration of electricity cost saving interventions on a water distribution utility. Potchefstroom: NWU. (Dissertation — MEng).

Brion, L. M. & Mays, L. W. 1991. Methodology for optimal operation of pumping stations in water distribution systems. *Journal of hydraulic engineering*, 117(11):1551–1569. DOI: 10.1061/(asce)0733-9429(1991)117:11(1551).

Brito, E. 2011. Making pump maintenance mandatory. *Chemical engineering*, 118(10):48. Available at: http://www.chemengonline.com/making-pump-maintenance-mandatory/.

Cembrano, G., Quevedo, J., Salamero, M., Puig, V., Figueras, J. & Martí, J. 2004. Optimal control of urban drainage systems: a case study. *Control engineering practice*, 12(1):1–9. DOI: 10.1016/S0967-0661(02)00280-0.

Cembrano, G., Wells, G., Quevedo, J., Peh Rez, R. & Argelaguet, R. 2000. Optimal control of a water distribution network in a supervisory control system. *Control engineering practice*, 8:1177–1188. DOI: 10.1016/S0967-0661(00)00058-7.

Cilliers, C. 2014. Cost savings on mine dewatering pumps by reducing preparation-and comeback loads. Potchefstroom: NWU. (Dissertation — MEng).

De Jager, J. P. 2015. Investigating the effect of pump availability on load shift performance. Potchefstroom: NWU. (Dissertation — MEng).

Dieu, B. 2001. Application of the SCADA system in wastewater treatment plants. *ISA transactions*, 40(3):267–281. DOI: 10.1016/S0019-0578(00)00053-7.

DWAF (Department of Water Affairs and Forestry). 2008. Best practice guideline A6: water management for underground mines. Available at: http://www.chamberofmines.org.za/work/environment/environmental-resources/send/26-environmental-resources/362-a6-water-management-for-underground-mines.

Els, L. A. 2015. Load management on a municipal water treatment plant. Potchefstroom: NWU. (Dissertation — MEng).

Eskom. 2017. Schedule of standard prices for Eskom tariffs (2017/2018). Available at: http://www.eskom.co.za/CustomerCare/TariffsAndCharges/Documents/Eskom schedule of standard prices 2017_18 15_02_2017 (00).pdf.

Felder, R. M., Rousseau, R. W. & Bullard, L. G. 2015. Elementary principles of chemical processes. 4th ed. Hoboken, NJ: John Wiley & Sons.

Gauch, H. G., Hwang, J. T. G. & Fick, G. W. 2003. Model evaluation by comparison of model-based predictions and measured values. *Agronomy journal*, 95(6):1442–1446. DOI: 10.2134/agronj2003.1442.

Grobbelaar, H. L. 2014. Maintenance procedures on DSM pumping projects to improve sustainability. Potchefstroom: NWU. (Dissertation — MEng).

Grundfos 2004. Pump handbook. Available at: http://net.grundfos.com/doc/webnet/mining/_downloads/pump-handbook.pdf.

Hasan, A. N., Twala, B. & Marwala, T. 2013. Predicting mine dam levels and energy consumption using artificial intelligence methods (*In* 2013 IEEE Symposium on computational intelligence for engineering solutions (CIES). p. 171–175). DOI: 10.1109/CIES.2013.6611745.

Horowitz, F. B., Lipták, B. G. & Bain, S. 2005. Pump controls. (*In* Lipták, B. G., *ed.* Instrument engineers' handbook, volume two: process control and optimization. 4th ed. Boca Raton, FL: CRC Press. p. 2084–2109).

Kobayashi, K. & Salam, M. U. 2000. Comparing simulated and measured values using mean squared deviation and its components. *Agronomy journal*, 92(2):345–352. DOI: 10.1007/s100870050043.

Lansey, K. E. & Awumah, K. 1994. Optimal pump operations considering pump switches. *Journal of water resources planning and management*, 120(1):17–35. DOI: 10.1061/(asce)0733-9496(1994)120:1(17).

Mehta, B. R. & Reddy, Y. J. 2014. Industrial process automation systems: design and implementation. Oxford: Butterworth-Heinemann.

Nortjé, A. 2012. DSM strategy for national water pumping systems. Potchefstroom: NWU. (Dissertation — MEng).

Oberholzer, P. J. 2014. Best practices for automation and control of mine dewatering systems. Potchefstroom: NWU. (Dissertation — MEng).

Oosthuizen, N. L. 2012. Optimum water distribution between pumping stations of multiple mine shafts. Potchefstroom: NWU. (Dissertation — MEng).

Ormsbee, L., Lingireddy, S. & Chase, D. 2009. Optimal pump scheduling for water distribution systems. (*In* Multidisciplinary international conference on scheduling: theory and applications (MISTA 2009)).

Pasha, M. F. K. & Lansey, K. 2009. Optimal pump scheduling by linear programming. (*In* World environmental and water resources congress 2009. p. 38–38). DOI: 10.1061/41036(342)38.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. 2011. Scikit-learn: machine learning in Python. *Journal of machine learning research*, 12:2825–2830. Available at: http://scikit-learn.org/0.19/modules/model_evaluation.html#regression-metrics.

Pezeshk, S. & Helweg, O. J. 1996. Adaptive search optimization in reducing pump operating costs. *Journal of water resources planning and management*, 122(1):57–63. DOI: 10.1061/(asce)0733-9496(1996)122:1(57).

Piñeiro, G., Perelman, S., Guerschman, J. P. & Paruelo, J. M. 2008. How to evaluate models: Observed vs. predicted or predicted vs. observed? *Ecological modelling*, 216(3–4):316–322. DOI: 10.1016/j.ecolmodel.2008.05.006.

Puleo, V., Morley, M., Freni, G. & Savić, D. 2014. Multi-stage linear programming optimization for pump scheduling. *Procedia engineering*, 70:1378–1385. DOI: 10.1016/j.proeng.2014.02.152.

Rautenbach, J. W. 2007. Engineering a novel automated pump control system for the mining environment. Potchefstroom: NWU. (Thesis — PhD).

Richter, R. P. 2008. Comparison between automated and manual DSM pumping projects. Potchefstroom: NWU. (Dissertation — MEng).

Schaschke, C. 2014. A dictionary of chemical engineering. Oxford: Oxford University Press.

Shankar, K. G. 2008. Control of boiler operation using PLC-SCADA. (*In* IMECS 2008: International multiconference of engineers and computer scientists. p. 1281–1286).

Smith, T. 2014. Automated control of mine dewatering pumps. Potchefstroom: NWU. (Dissertation — MEng).

TEMM International (Pty) Ltd 2017. Real-Time Energy Management System v8.2.14.

Van der Merwe, M. 2016. Strategies to revive DSM mine pumping projects under the new ESCo model. Potchefstroom: NWU. (Dissertation — MEng).

Van Niekerk, A. P. 2014. Implementing DSM interventions on water reticulation systems of marginal deep level mines. Potchefstroom: NWU. (Dissertation — MEng).

Van Niekerk, W. F. 2013. The value of simulation models for mine DSM projects. Potchefstroom: NWU. (Dissertation — MEng).

Vosloo, J. C. 2008. A new minimum cost model for water reticulation systems on deep mines. Potchefstroom: NWU. (Thesis — PhD).

Vosloo, J. C., Liebenberg, L. & Velleman, D. 2012. Case study: energy savings for a deep-mine water reticulation system. *Applied energy*, 92:328–335. DOI: 10.1016/j.apenergy.2011.10.024.

Wonderware 2017. Wonderware InTouch 2017. Available at: https://www.wonderware.com/hmi-scada/intouch/.

Zhuan, X. & Xia, X. 2013. Optimal operation scheduling of a pumping station with multiple pumps. *Applied energy*, 104:250–257. DOI: 10.1016/j.apenergy.2012.10.028.

COMPARATIVE STUDY OF MINE
DEWATERING CONTROL SYSTEMS

APPENDIX A

ESKOM MEGAFLEX TARIFFS 2017/2018

APPENDIX A: ESKOM MEGAFLEX TARIFFS 2017/2018

Table 9, below, lists the Eskom Megaflex active energy charges (c/kWh, VAT exclusive) for non-local authority supplies applicable from 1 April 2017 to 31 March 2018 (Eskom, 2017:21).

TABLE 9 – ESKOM MEGAFLEX TARIFFS 2017/2018

Transmission zone	Voltage	Low demand season Peak	Low demand season Standard	Low demand season Off-peak	High demand season Peak	High demand season Standard	High demand season Off-peak
≤ 300 km	< 500 V	91.14	62.89	40.09	278.33	84.68	46.24
	≥ 500 V & < 66 kV	89.36	61.51	39.02	273.96	83.00	45.07
	≥ 66 kV & ≤ 132 kV	86.55	59.56	37.79	265.29	80.36	43.65
	> 132 kV	81.58	56.13	35.62	250.03	75.74	41.14
> 300 km & ≤ 600 km	< 500 V	91.54	63.02	39.98	280.60	85.02	46.16
	≥ 500 V & < 66 kV	90.27	62.12	39.41	276.70	83.82	45.52
	≥ 66 kV & ≤ 132 kV	87.39	60.14	38.15	267.90	81.15	44.06
	> 132 kV	82.36	56.69	35.95	252.53	76.51	41.52
> 600 km & ≤ 900 km	< 500 V	92.45	63.63	40.35	283.4	85.85	46.60
	≥ 500 V & < 66 kV	91.16	62.75	39.81	279.48	84.67	45.98
	≥ 66 kV & ≤ 132 kV	88.27	60.76	38.54	270.63	81.98	44.51
	> 132 kV	83.21	57.26	36.34	255.08	77.26	41.97
> 900 km	< 500 V	93.39	64.26	40.79	286.25	86.74	47.09
	≥ 500 V & < 66 kV	92.06	63.35	40.20	282.26	85.50	46.41
	≥ 66 kV & ≤ 132 kV	89.16	61.37	38.93	273.34	82.80	44.96
	> 132 kV	84.07	57.88	36.74	257.56	78.06	42.41

COMPARATIVE STUDY OF MINE
DEWATERING CONTROL SYSTEMS

APPENDIX B

ROOT MEAN SQUARE DEVIATION FOR COMPARISON OF SIMULATION AND ACTUAL VALUES

APPENDIX B: ROOT MEAN SQUARE DEVIATION FOR COMPARISON OF SIMULATION AND ACTUAL VALUES

Model or simulation accuracy is tested by comparing its outputs to observed (actual) values. The accuracy is commonly evaluated by using the correlation coefficient (r), coefficient of determination (r^2), or evaluation of linearly-regressed trend line equation parameters. However, Kobayashi & Salam (2000) and Gauch *et al.* (2003) showed that mean square deviation (MSD) is more satisfactory and informative for model evaluation. If $\hat{y}_i$ is i^{th} simulated value and y_i the corresponding actual value, the estimated MSD over n values are

$$\text{MSD} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{22}$$

The square root of MSD, the root mean square error (RMSD), relates the MSD to the variable under evaluation by giving it the same order of magnitude and units of measurement:

$$\text{RMSD} = \sqrt{\text{MSD}} \tag{23}$$

The RMSD represents the mean deviation of predicted/simulated/calculated values with respect to the observed/actual values. The lower this value, the closer the simulated and actual values are to each other, or stated otherwise, the more accurate the simulation.

COMPARATIVE STUDY OF MINE
DEWATERING CONTROL SYSTEMS

APPENDIX C

CASE STUDY 2 SIMULATION RESULTS (DETAIL)

APPENDIX C: CASE STUDY 2 SIMULATION RESULTS (DETAIL)

This section contains the detailed description and simulation results of Case study 2.

C.1. DESCRIPTION OF CASE STUDY AND SPECIFICS

A gold mine in South Africa with two pumping levels provided Case study 2 (CS2). Figure 37 illustrates the layout of Case study 2's dewatering system.

The bottom level is 1900 m below ground and is known as 27-level (27L). This level has only fissure water as an inflowing water source to its two 3 ML dams. 27-level has four installed and working dewatering pumps. These pumps have a rated power of 2 750 kW per pump. The average measured power consumption of the pumps is 2656.6 kW and the average flow is calculated to be 194.6 L/s per pump. Although it is possible to run three pumps at a time on 27L, the mine prefers to run two to lessen the strain on the electrical infrastructure.

From 27-level, water is pumped to 12-level (12L). 12-level has additional water inflow in the form of fissure water and water transferred from another shaft. This level has two 3 ML dams and five installed and working pumps. The pumps have a rated power of 3 300 kW per pump. The average measured power consumption of the pumps is 2 925.6 kW and the average flow is calculated to be 236.1 L/s per pump. Three pumps can run at any one time on this level.

From 12-level, water is pumped to "biological dams" located on the surface. These are "natural" dams with reeds growing in it, purifying the water before it overflows from these dams into a pipeline feeding farmlands, amongst others.

Table 10 (page 101) lists the answers to the questions posed in Section 3.3 (page 30), needed for simulation of the system. Because the surface biological dams are designed to overflow, it is not necessary to include these dams in the simulation model for CS2.

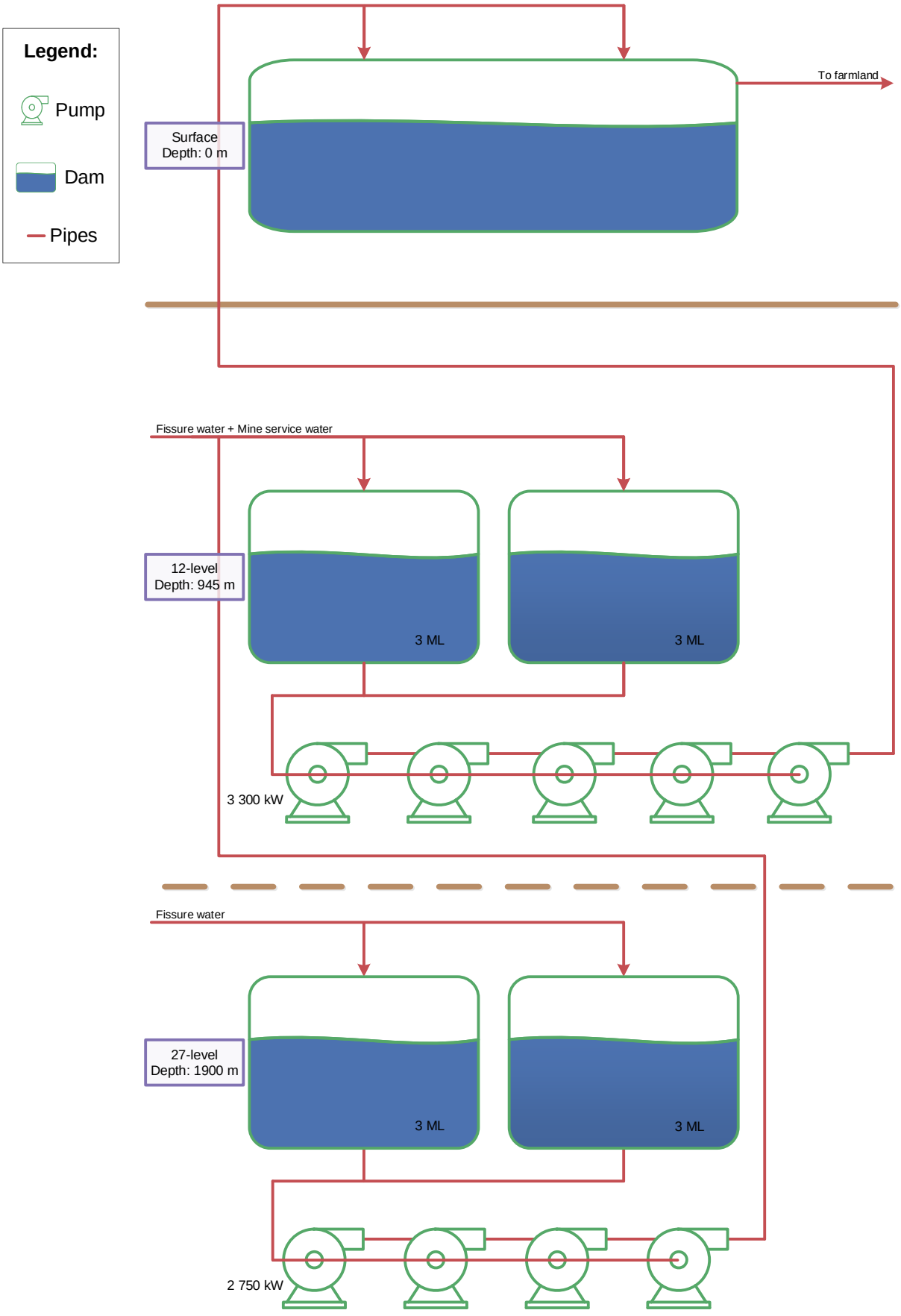


FIGURE 37 – CASE STUDY 2 (CS2) DEWATERING SYSTEM LAYOUT

TABLE 10 – CASE STUDY 2 (CS2) SIMULATION INFORMATION

	27-level	12-level	Surface
Number of dams	3	3	3
Dam capacity	3 ML	3 ML	?
Dams interconnected	Yes	Yes	Yes
Additional inflow	Varies per number of pumps running on 27L Varies throughout the day	Varies throughout the day	No
Dam level limits (preferred)	Minimum: 30% Maximum: 80%	Minimum: 30% Maximum: 80%	N/A
Penalisation if dam(s) run empty/overflow	Yes	Yes	No
Dam penalisation limits	≤25%, ≥85%	≤25%, ≥85%	N/A
Pumps installed	4	5	
Power consumption per pump	2 925.6 kW	2 656.6 kW	
Flow per pump	236.1 L/s	194.6 L/s	
Maximum pumps running simultaneously	3 (2 preferred)	3	
Eskom tariff structure	Megaflex ≤300 km, 500 V – 66 kV		

C.2. SIMULATION RESULTS

VALIDATION OF BASE CASE SIMULATION

From Figure 38, it is seen that for 27L, the validation mode of the simulation followed the exact running schedules that were followed in reality. For 27L, the simulation was able to accurately capture the behaviour of the dam level (Figure 39). 27L's dam level simulation has an RMSD value of 0.69%.

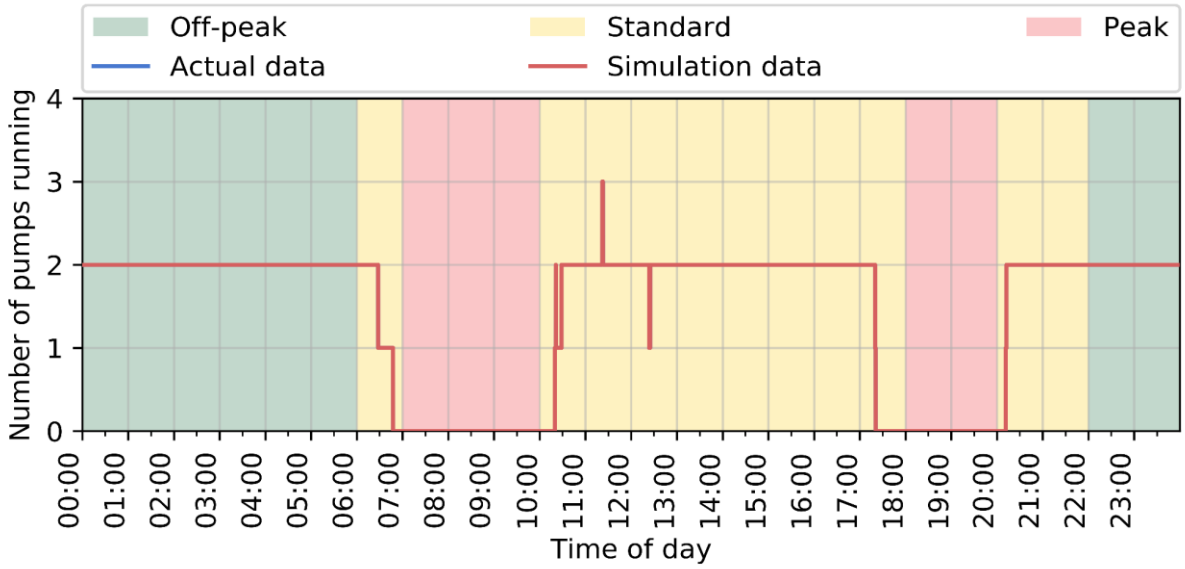


FIGURE 38 – CS2 SIMULATION VALIDATION: 27L PUMP STATUSES

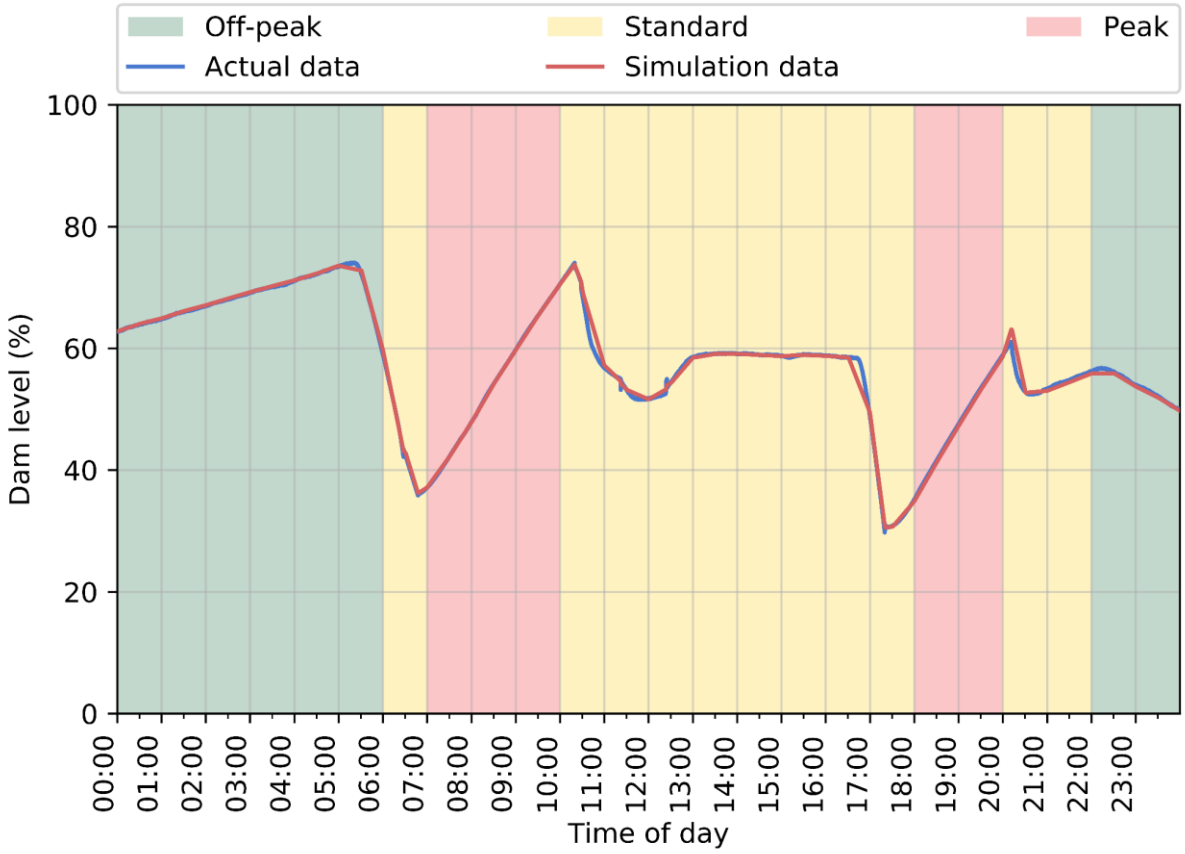


FIGURE 39 – CS2 SIMULATION VALIDATION: 27L DAM LEVEL

Figure 40 illustrates the actual and simulated pump running schedules on 12L for this same day. It is seen that the simulation followed the same pump running schedules. Figure 41 illustrates the corresponding dam level at 12L. It is seen that the simulation accurately describes the behaviour of the actual dam level, as supported by an RMSD of 0.71%.

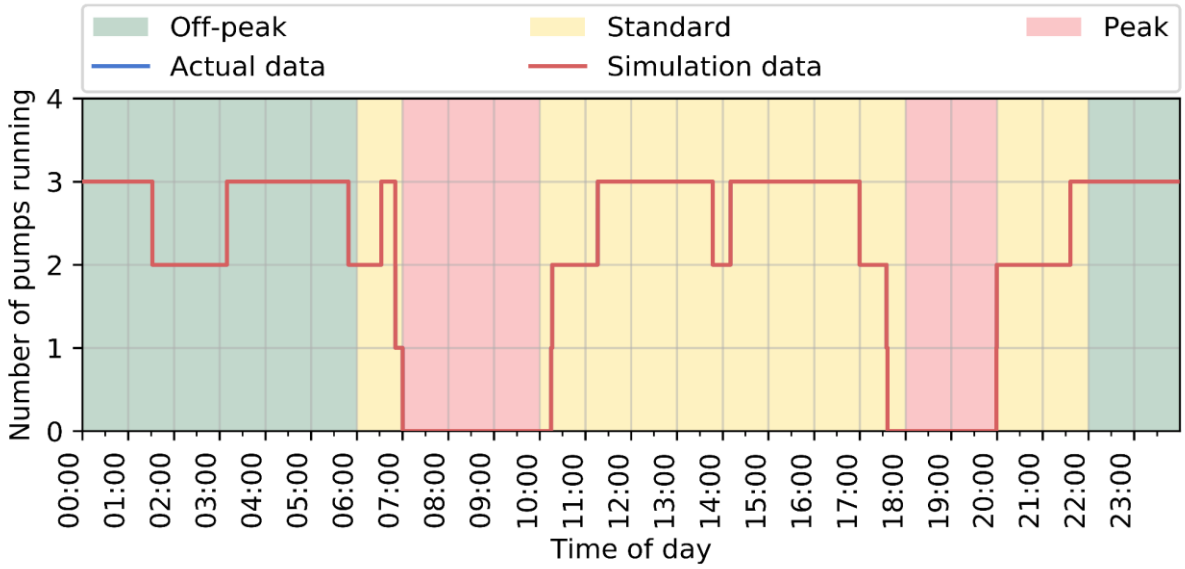


FIGURE 40 – CS2 SIMULATION VALIDATION: 12L PUMP STATUSES

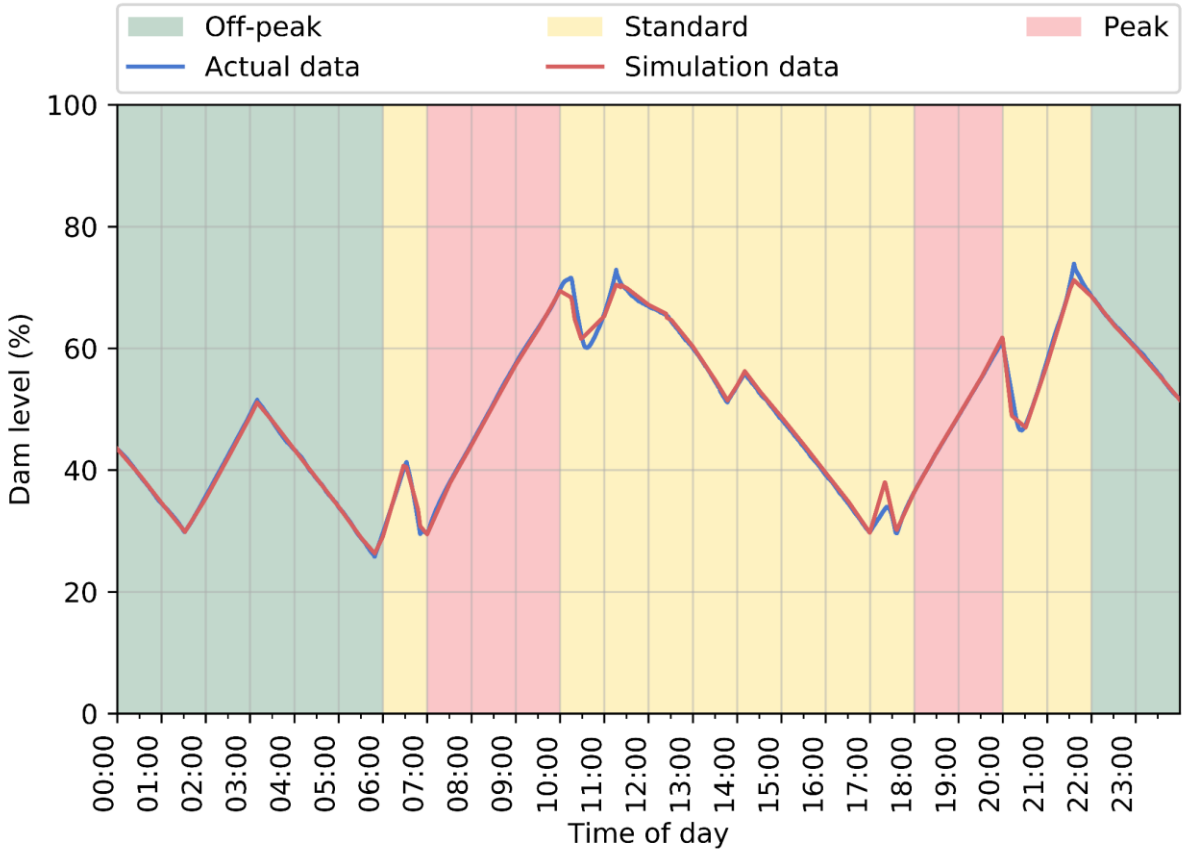


FIGURE 41 – CS2 SIMULATION VALIDATION: 12L DAM LEVEL

There is little difference between the actual and simulated values, as supported by an average RMSD value of 0.700%. It can be concluded that the simulation and assumptions made are accurate enough to continue making inferences from it.

1-FACTOR CONTROL SIMULATION

A 1-factor control simulation was run for the CS2 dewatering system. At 27L, the control system is able to control the dam level between its minimum and maximum values (Figure 42). However, there is some amount of unnecessary pump cycling occurring at 11:00.

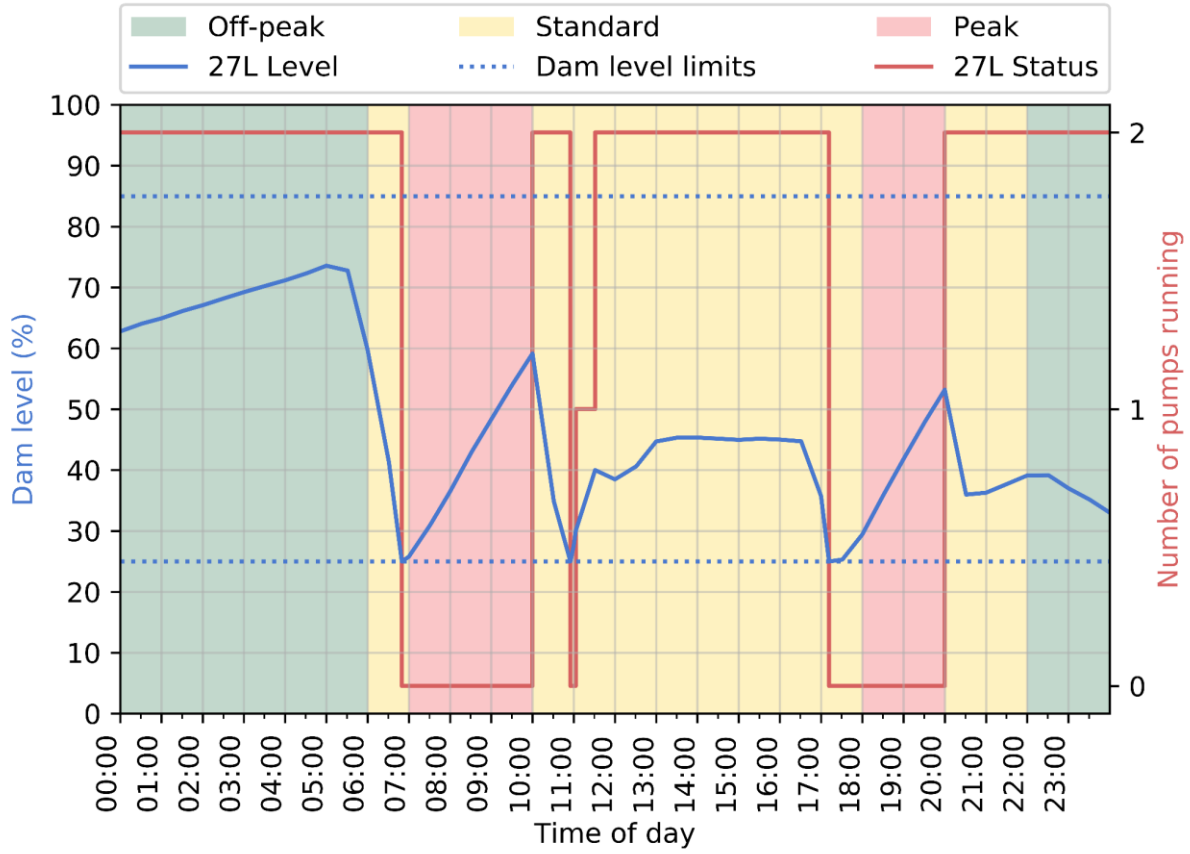


FIGURE 42 – CS2 1-FACTOR SIMULATION RESULTS: 27L DAM LEVEL AND PUMP STATUSES

Similarly, at 12L (Figure 43, on the next page), just before 18:00, the control system switches on a pump, only to be switched off when peak period starts at 18:00.

For both pumping levels, load-shift is performed well. Following the pumping schedule as in Figures 42 and 43, the total power consumption varies between 10.6 MW and 13.7 MW during non-peak periods, and 0 MW during peak periods (Figure 44, on the next page). Using the Eskom Megaflex ≤300 km, 500 V – 66 kV tariff, the BPC for the operation of this simulated day is R91 878.95. The SPC for this day is R121 942.09.

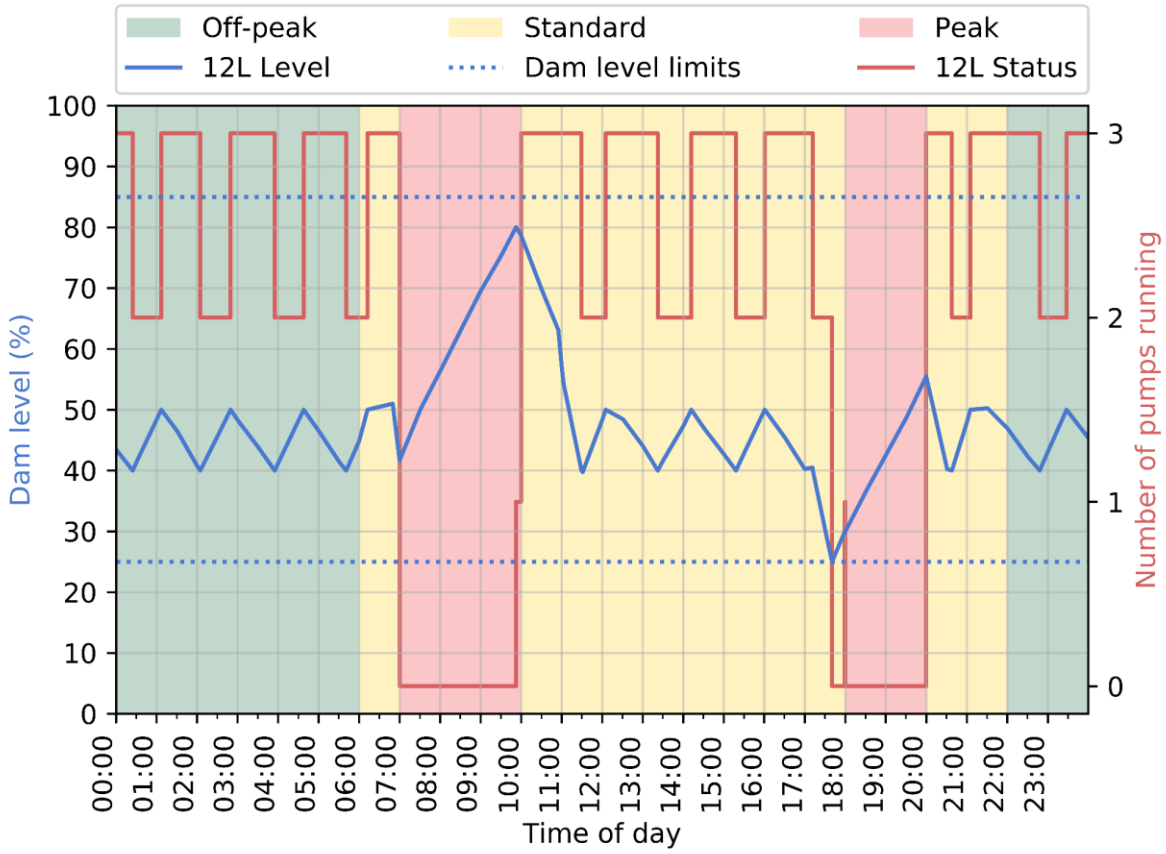


FIGURE 43 – CS2 1-FACTOR SIMULATION RESULTS: 12L DAM LEVEL AND PUMP STATUSES

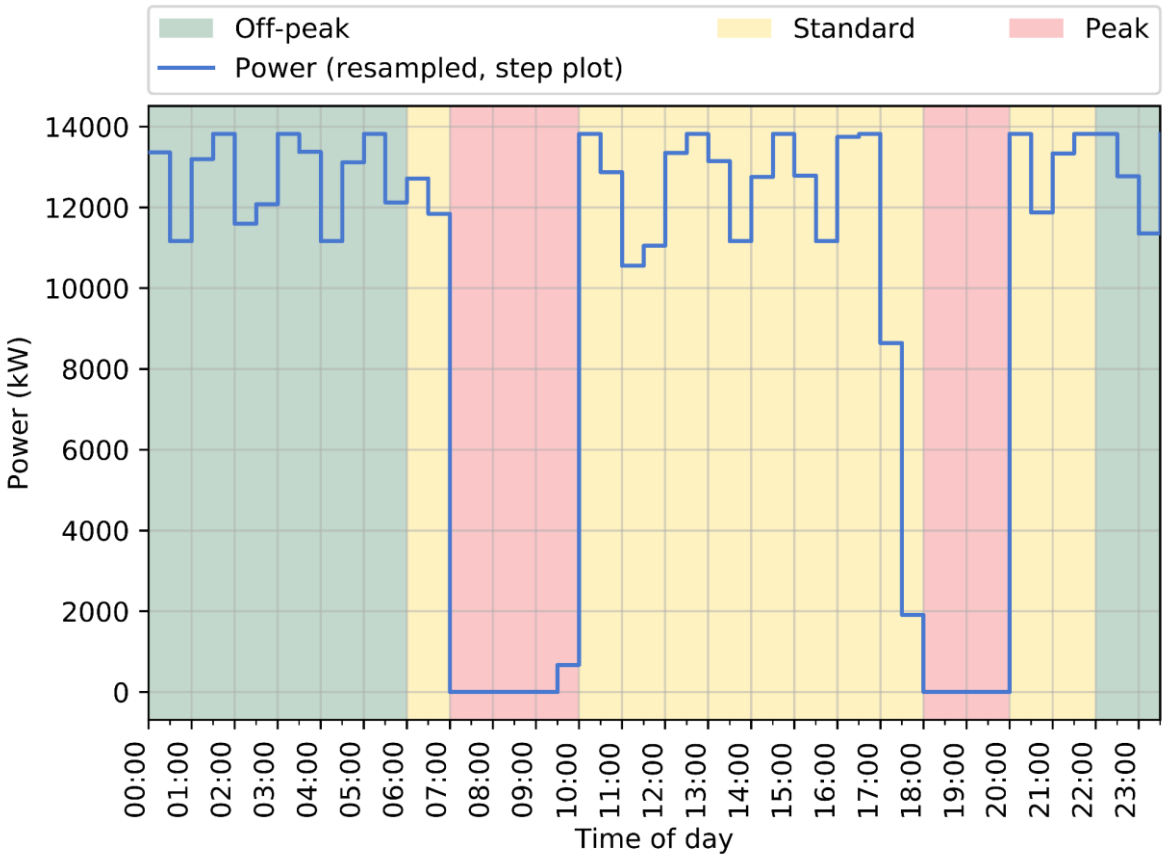


FIGURE 44 – CS2 1-FACTOR SIMULATION RESULTS: POWER CONSUMPTION (RESAMPLED DATA)

2-FACTOR CONTROL SIMULATION

As was the case with Case study 1, the outputs for Case study 2's 2-factor simulation are exactly the same as the 1-factor simulation. This is because the only difference is in the scheduling if the downstream dam reaches ≥95% (as seen in the decision-making algorithm, Figure 13, page 36). The 12L dam did not reach 95%, causing the 2-factor outputs to be identical. Results are not repeated.

Using the Eskom Megaflex ≤300 km, 500 V – 66 kV tariff, the BPC for the operation of this day is R91 878.95, whereas the SPC is R121 942.09.

n-FACTOR CONTROL SIMULATION

Performing an n-factor control simulation on the CS2 dewatering system reveals that this control system is able of controlling all dam levels between their control limits (Figures 45 and 46). This is done with no preventable cycling of the pumps on 27L and 12L. The control system is able to perform morning and evening load-shifts successfully.

Running the pumps as shown Figures 45 and 46 results in a total power consumption for CS2 as shown in Figure 47 (on the next page). Using the Eskom Megaflex ≤300 km, 500 V – 66 kV tariff, the BPC for the operation of this day is R91 171.81, whereas the SPC is R121 379.67.

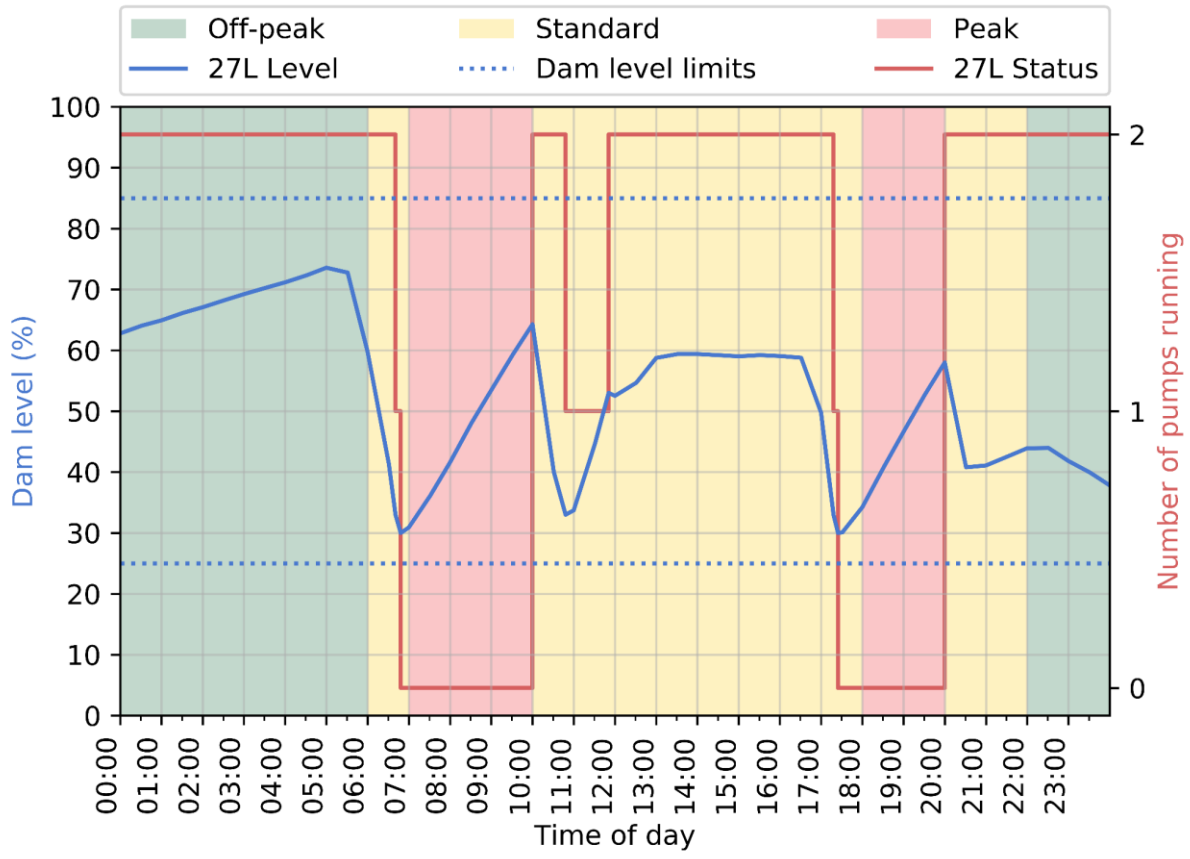


FIGURE 45 – CS2 n-FACTOR SIMULATION RESULTS: 27L DAM LEVEL AND PUMP STATUSES

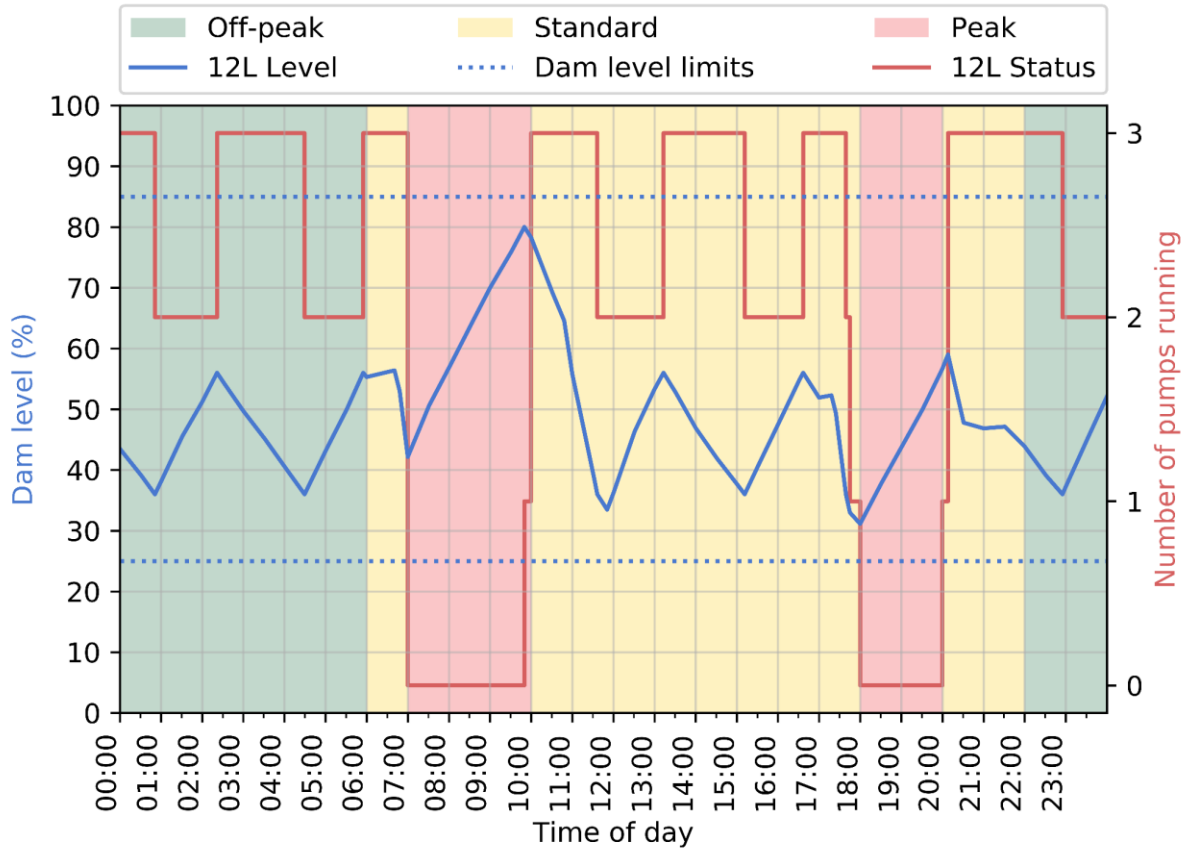


FIGURE 46 – CS2 n-FACTOR SIMULATION RESULTS: 12L DAM LEVEL AND PUMP STATUSES

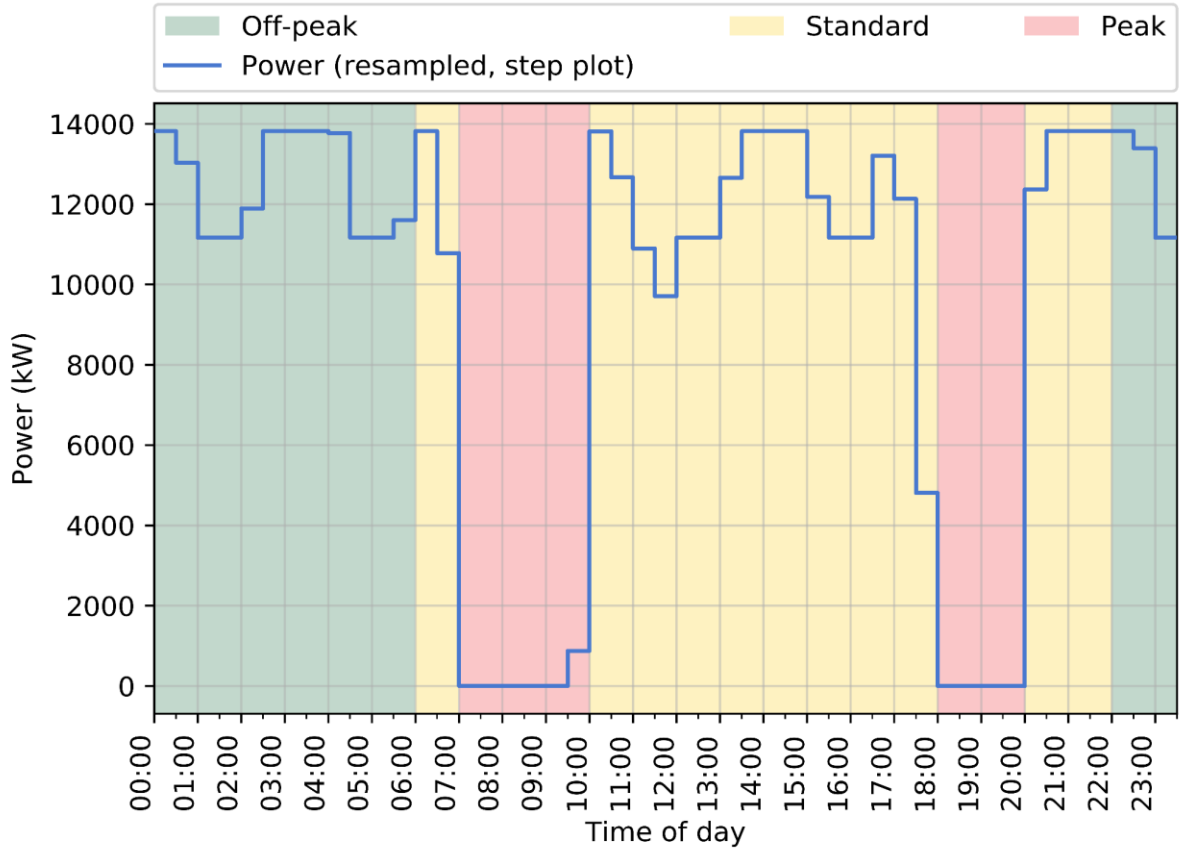


FIGURE 47 – CS2 n-FACTOR SIMULATION RESULTS: POWER CONSUMPTION

COMPARATIVE STUDY OF MINE
DEWATERING CONTROL SYSTEMS

APPENDIX D

CASE STUDY 3 SIMULATION RESULTS (DETAIL)

APPENDIX D: CASE STUDY 3 SIMULATION RESULTS (DETAIL)

This section contains the detailed description and simulation results of Case study 3.

6.1.1. DESCRIPTION OF CASE STUDY AND SPECIFICS

A South African gold mine with four dewatering levels provided Case study 3 (CS3). Figure 48 (on the next page) shows the layout of the dewatering system.

CS3 has four dewatering levels. The bottom-most level, 41-level (41L), has five dewatering pumps. These pumps pump water to 31-level (31L), which has four dewatering pumps. From 31L, water is pumped to 20-level (20L). 20L's four pumps pump water to IM-level. IM-level has five pumps that pump water to the surface dam. Some of the water from the surface dam is fed to a gold processing plant, while the rest of the water is treated and fed back to the underground circuit.

Each of the dewatering levels has two dams installed, but only one is in use per level. All dams have capacities of 5 ML. All levels have fissure water inflows present, which vary from day to day, throughout the day. 20L has the option of requesting additional water from another connected shaft, in the case that additional water is required in the surface tank.

The measured power consumptions of the pumps at CS3 vary between 3.3 MW to 3.6 MW. Average calculated flows vary from 147 to 217 L/s. Table 11 contains the information required to perform a simulation of the CS3 dewatering system. Even though it is tolerable if the surface dam overflows, this will warrant penalisation in this study because the overflow of this dam is not ideal.

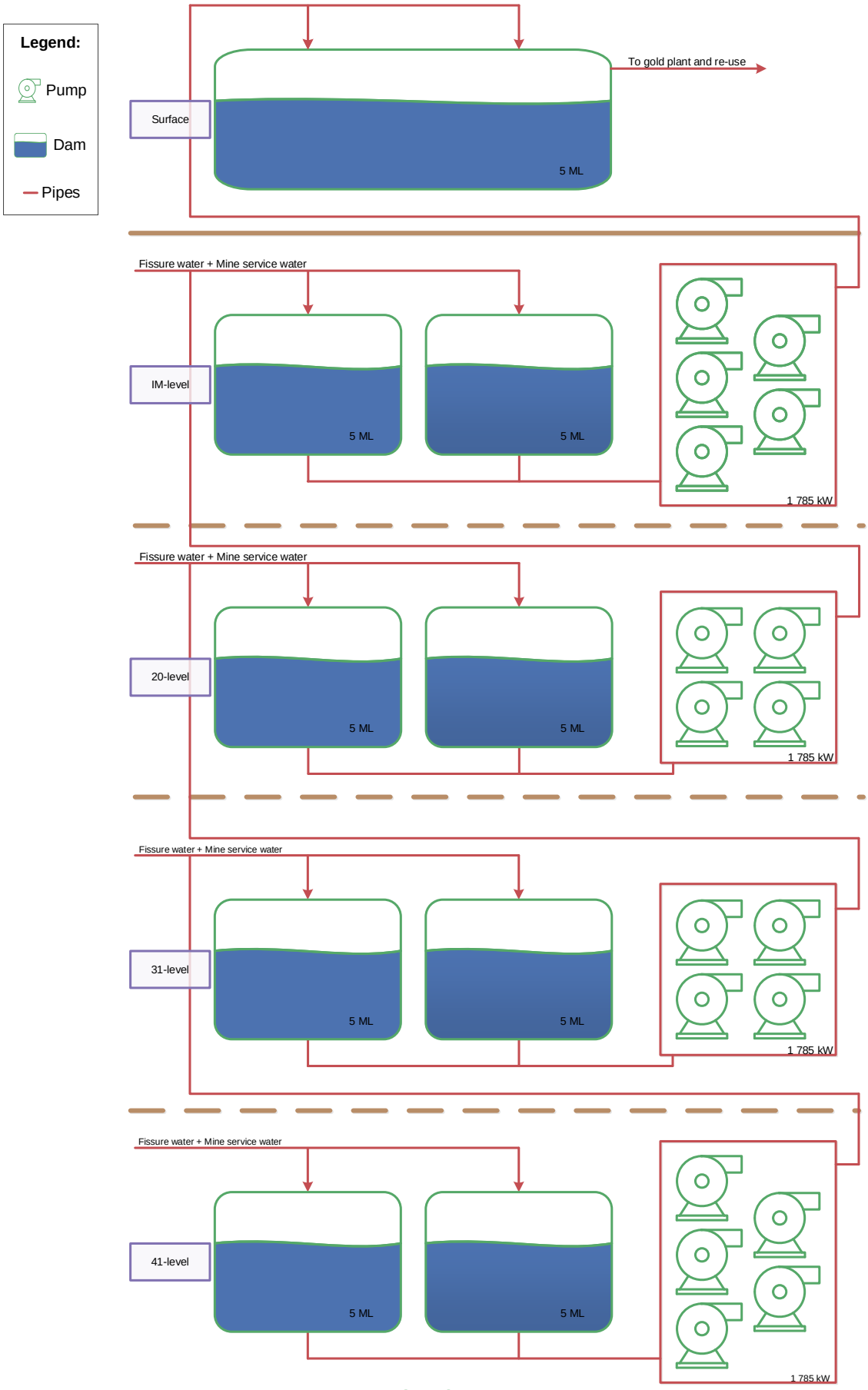


FIGURE 48 – CASE STUDY 3 (CS3) DEWATERING SYSTEM LAYOUT

Table 11 contains the information required to complete the simulation of the dewatering system:

TABLE 11 – CASE STUDY 3 (CS3) SIMULATION INFORMATION

	41L	31L	20L	IM	Surface
Number of dams	2	2	2	2	1
Dam capacity	5 ML	5 ML	5 ML	5 ML	5 ML
Dams interconnected	No. Only one dam in use.	No. Only one dam in use.	No. Only one dam in use.	No. Only one dam in use.	N/A
Additional inflow	Fissure water.	A small amount of fissure water.	A small amount of fissure water. Additional water upon request.	A small amount of fissure water.	None.
Dam level limits (preferred)	Minimum: 45% Maximum: 80%	Minimum: 45% Maximum: 80%	Minimum: 45% Maximum: 80%	Minimum: 45% Maximum: 80%	Minimum: 45% Maximum: 100%
Penalisation if dam(s) run empty/overflow	Yes	Yes	Yes	Yes	Overflow not preferable
Dam penalisation limits	≤25%, ≥95%	≤25%, ≥95%	≤25%, ≥85%	≤25%, ≥95%	≤40% before peak, >100%
Pumps installed	5	4	4	5	
Power consumption per pump	3 572.8 kW	3 821.0 kW	3 283.6 kW	3 508.4 kW	
Flow per pump	147.4 L/s	171.8 L/s	146.8 L/s	216.8 L/s	
Maximum pumps running simultaneously	3	2	2	3	
Eskom tariff structure	Megaflex ≤300 km, 500 V – 66 kV				

6.1.2. SIMULATION RESULTS

VALIDATION OF BASE CASE SIMULATION

The CS3 simulations are set up the same way as the previous simulations.

Figure 49 shows that the simulation followed the same pump schedules at 41L as was followed by the pump operators at CS3.

The dam level comparison for 41L reveals an RMSD of 2.81% (Figure 50, on the next page). The biggest discrepancies between the simulated and actual dam levels are noted during peaks in the dam level. These peaks are the result of sudden changes (inrushes) of water into the 41L dam, which are not perfectly captured by the simulation assumptions and 30-minute inflow calculations. The average difference between the actual and simulated dam level, however, is still a very respectable 2.81% and satisfactory for further simulation purposes.

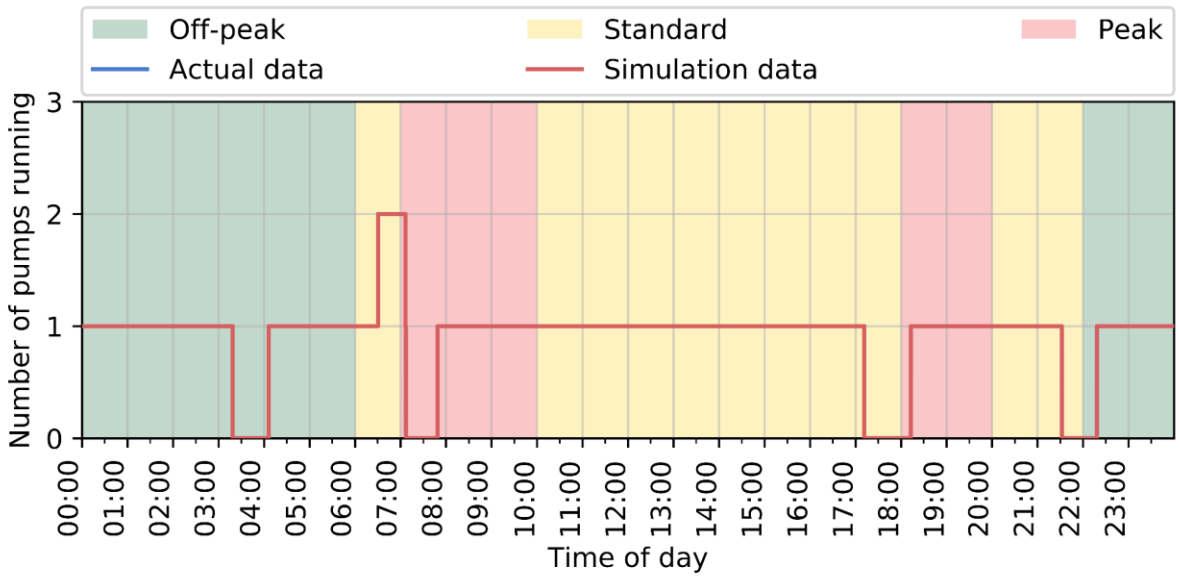


FIGURE 49 – CS3 SIMULATION VALIDATION: 41L PUMP STATUSES

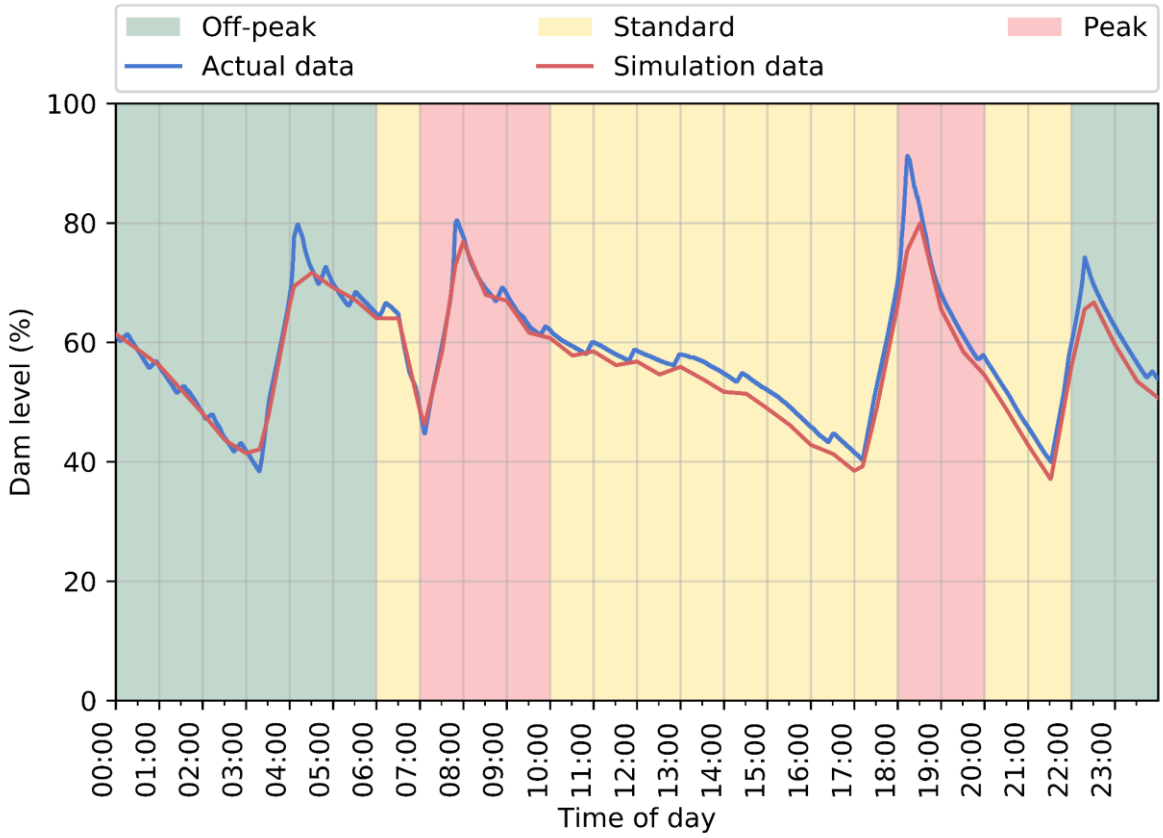


FIGURE 50 – CS3 SIMULATION VALIDATION: 41L DAM LEVEL

Figure 51 shows that the simulation of 31L followed the same pumping schedule as was followed in reality. Comparing the 31L dam level results (Figure 52), a more pronounced difference is noticed. The RMSD value of 3.80% supports this. This is because of erratic inflow of fissure water into the dam, which is not perfectly captured by the 30-minute resolution used for calculations. With the simulation dam levels being higher than the actual dam levels, this means that the assumptions create a model for 31L which errs on the conservative side.

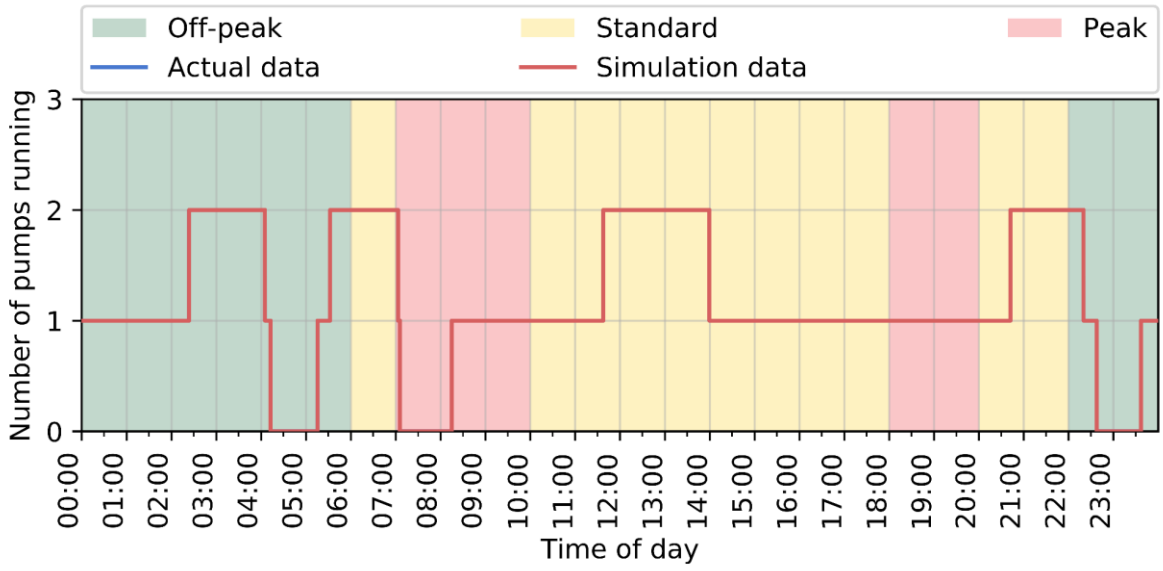


FIGURE 51 – CS3 SIMULATION VALIDATION: 31L PUMP STATUSES

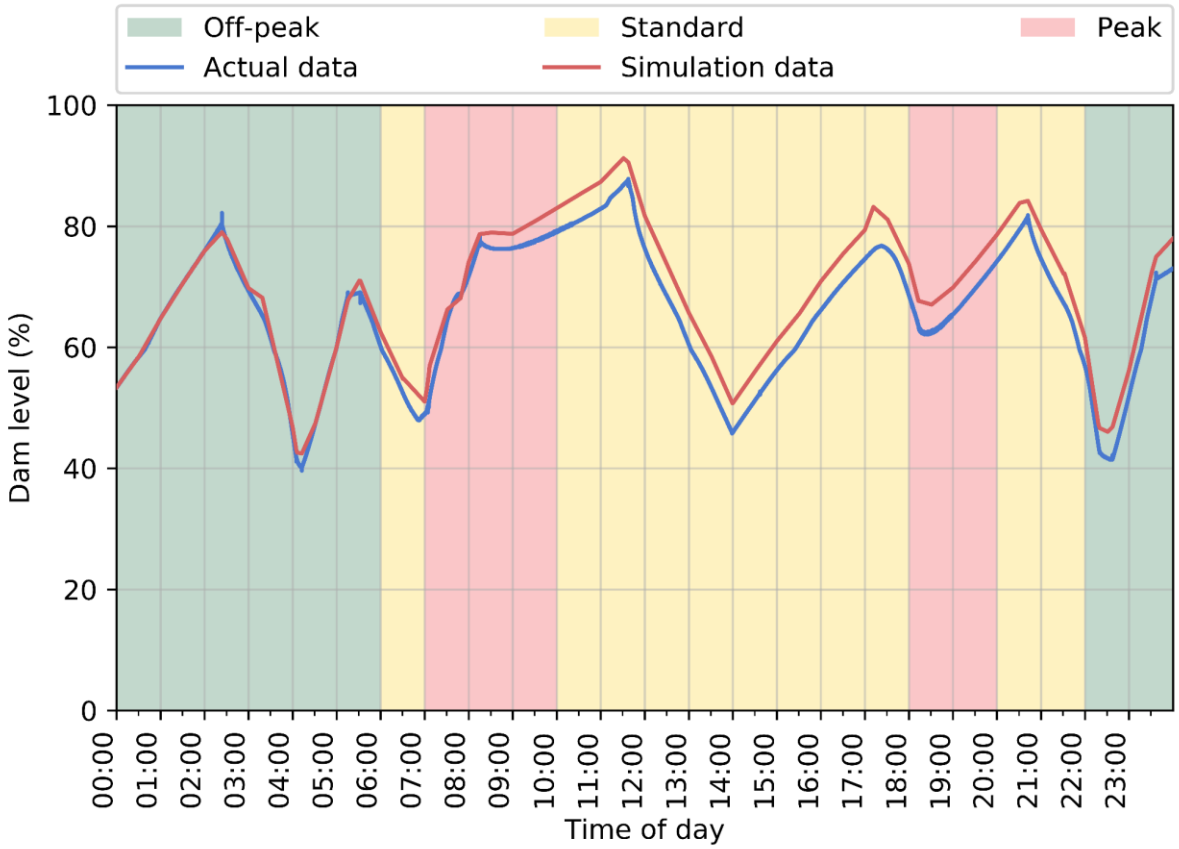


FIGURE 52 – CS3 SIMULATION VALIDATION: 31L DAM LEVEL

The validation simulation followed the same pumping schedule for 20L as was followed in reality (Figure 53). The simulated level of the 20L dam is higher than the actual data (Figure 54). The RMSD for the comparison between the simulated and actual dam levels is 4.18%. The simulation thus creates a conservative model of the dam level.

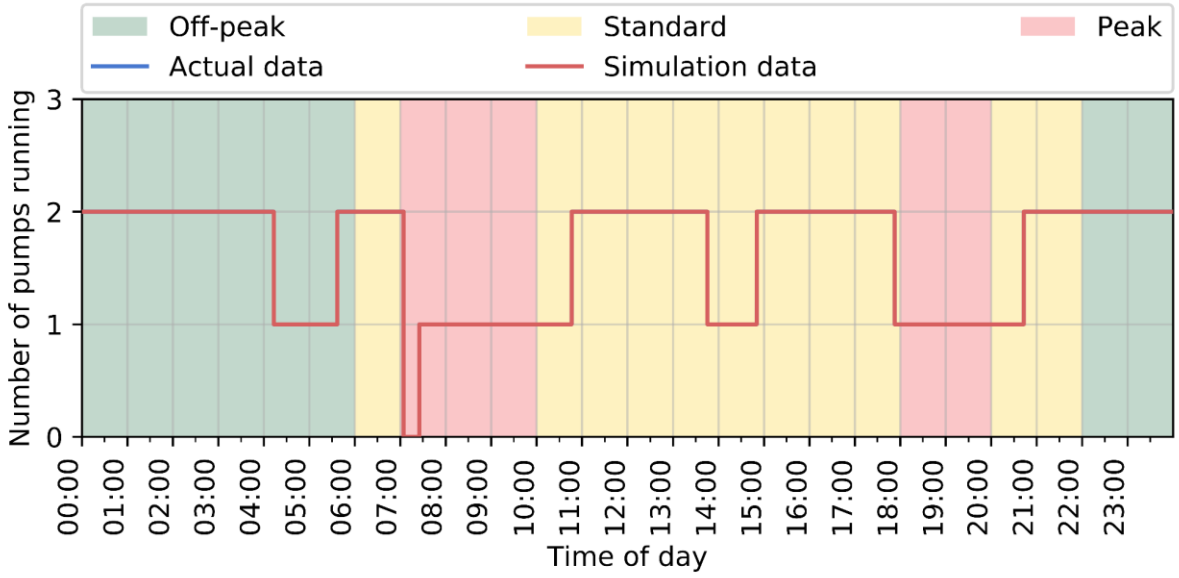


FIGURE 53 – CS3 SIMULATION VALIDATION: 20L PUMP STATUSES

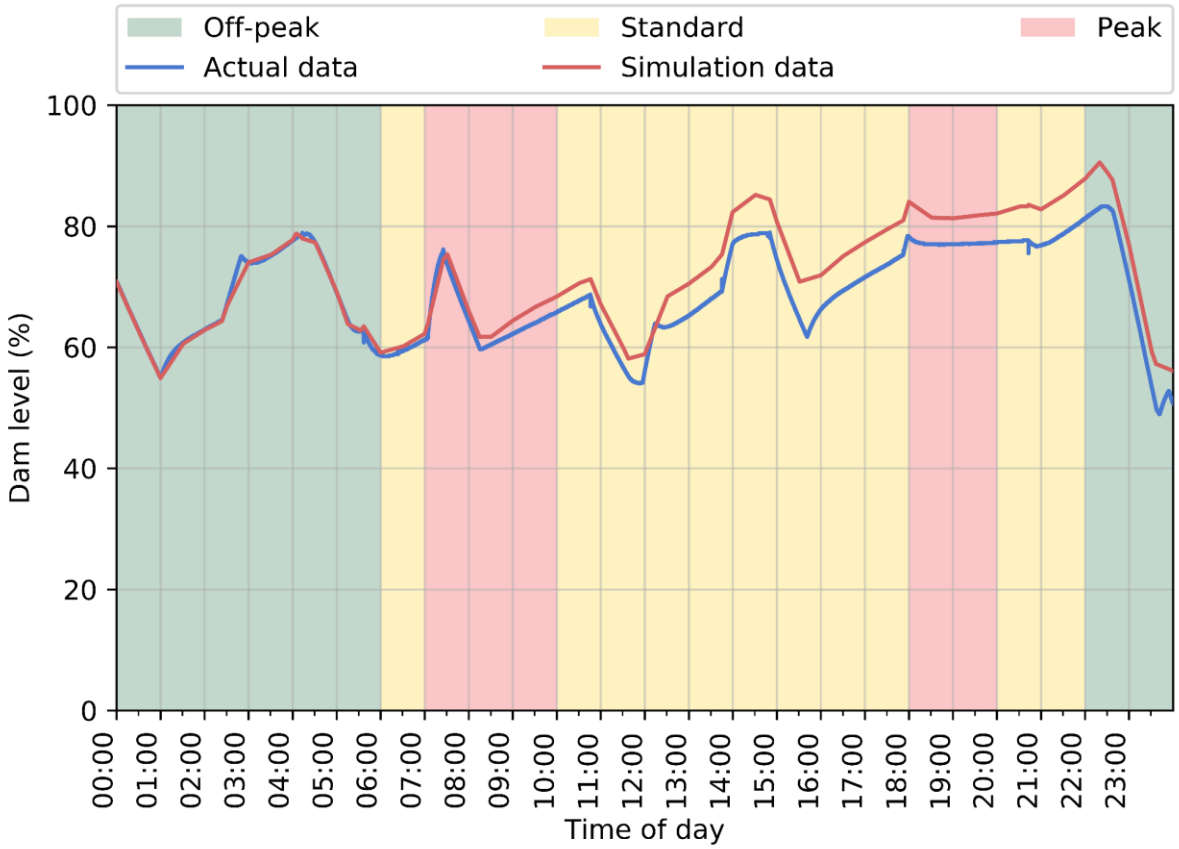


FIGURE 54 – CS3 SIMULATION VALIDATION: 20L DAM LEVEL

Figure 55 shows that the validation simulation for CS3 followed the same pumping schedules at IM-level as was followed in reality. Figure 56 shows that the simulation closely follows the same dam level profile. For IM-level, model assumptions were proven to be accurate by the RMSD value of 1.75%.

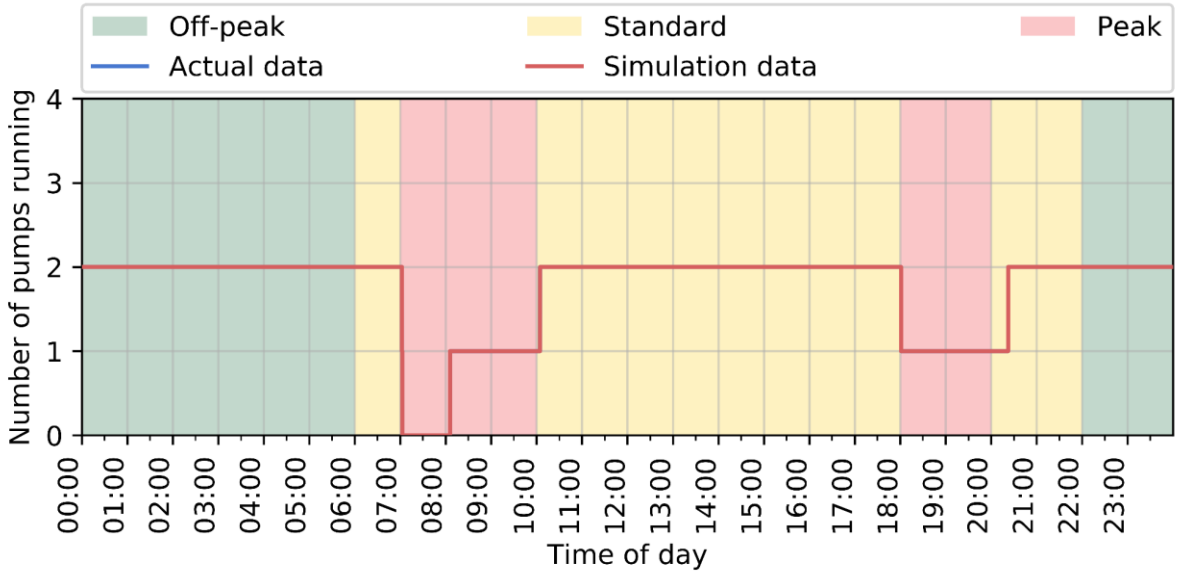


FIGURE 55 – CS3 SIMULATION VALIDATION: IM PUMP STATUSES

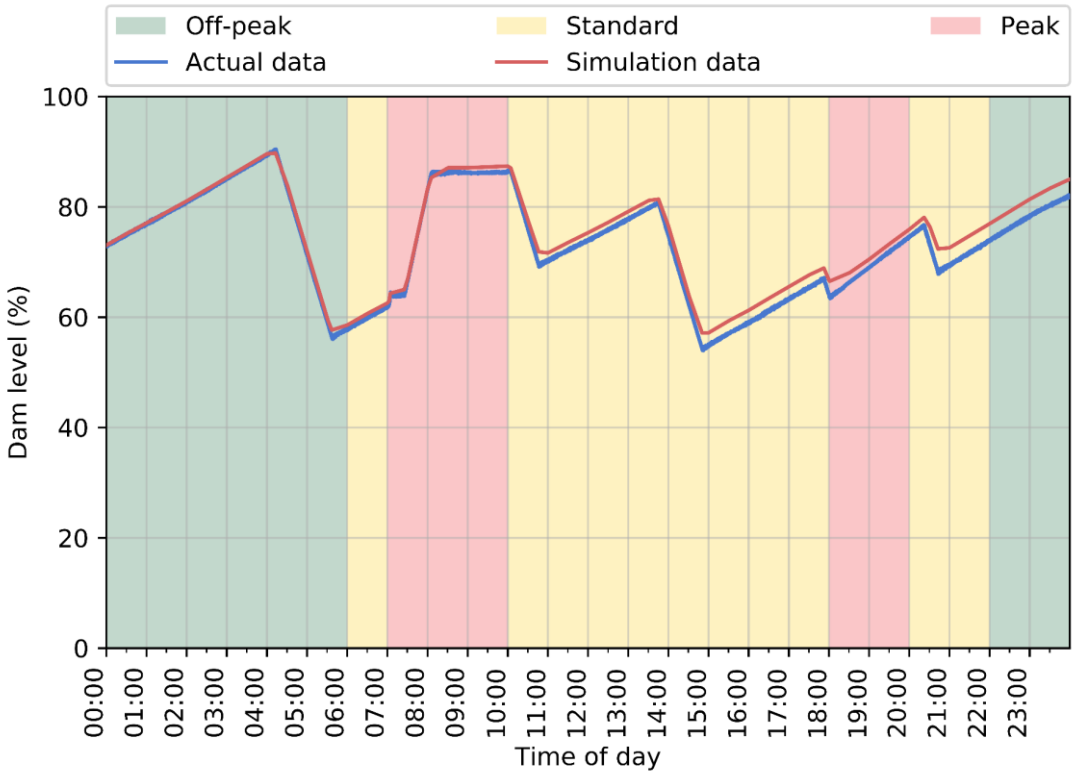


FIGURE 56 – CS3 SIMULATION VALIDATION: IM DAM LEVEL

Because the dam level of the surface dam is also of importance, this dam level was included in the simulation (Figure 57). It was found that the simulation attempt of this dam level was accurate, as supported by the RMSD value of 0.81%.

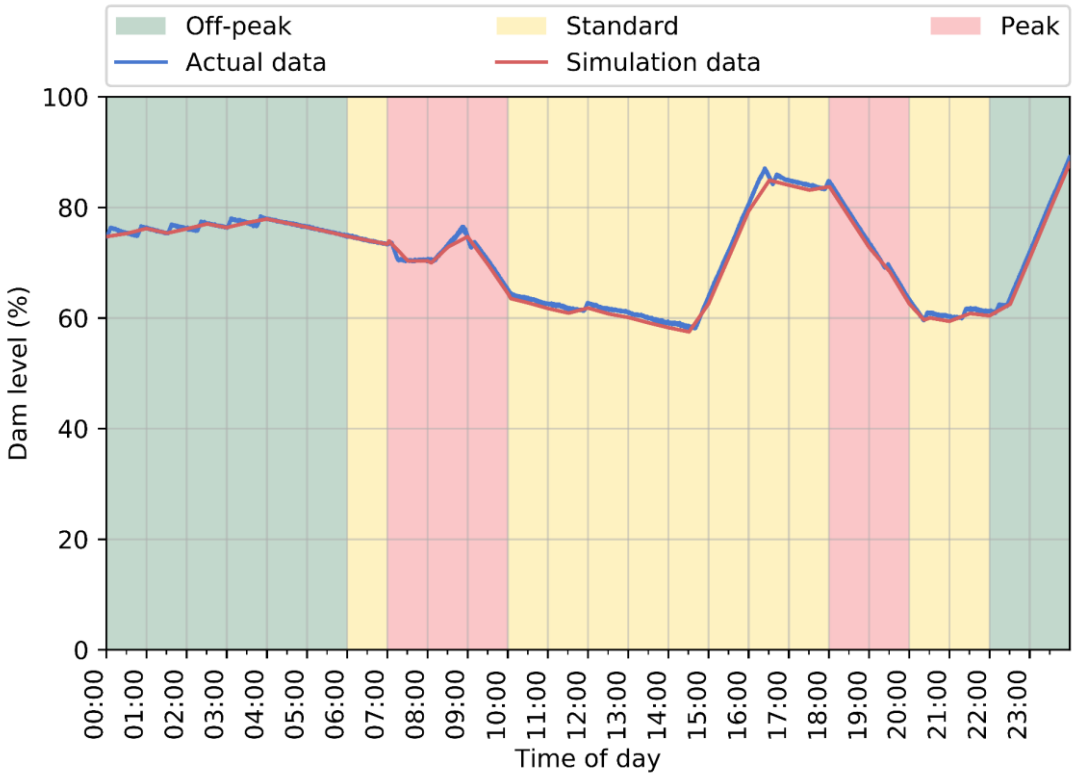


FIGURE 57 – CS3 SIMULATION VALIDATION: SURFACE DAM LEVEL

Combining all validation results listed in this section, it is found that the CS3 dewatering system was simulated accurately, with an average RMSD value of 2.670%. The simulation of the CS3 dewatering system is thus accurate enough to perform additional simulations on this dewatering system.

Inspecting the actual data for the day considered for simulation, one notes that “poor” load-shifts were performed and dam levels were considerably high throughout (Figures 49 to 56). This was a typical problematic day at a mine, where water levels underground are such that performing load-shifting is difficult.

1-FACTOR CONTROL SIMULATION

Performing a 1-factor control simulation on the CS3 dewatering system reveals that pump cycling is caused on 41L (Figure 58). This happens shortly after the running pump is switched off. The 41L dam level was kept within the control range.

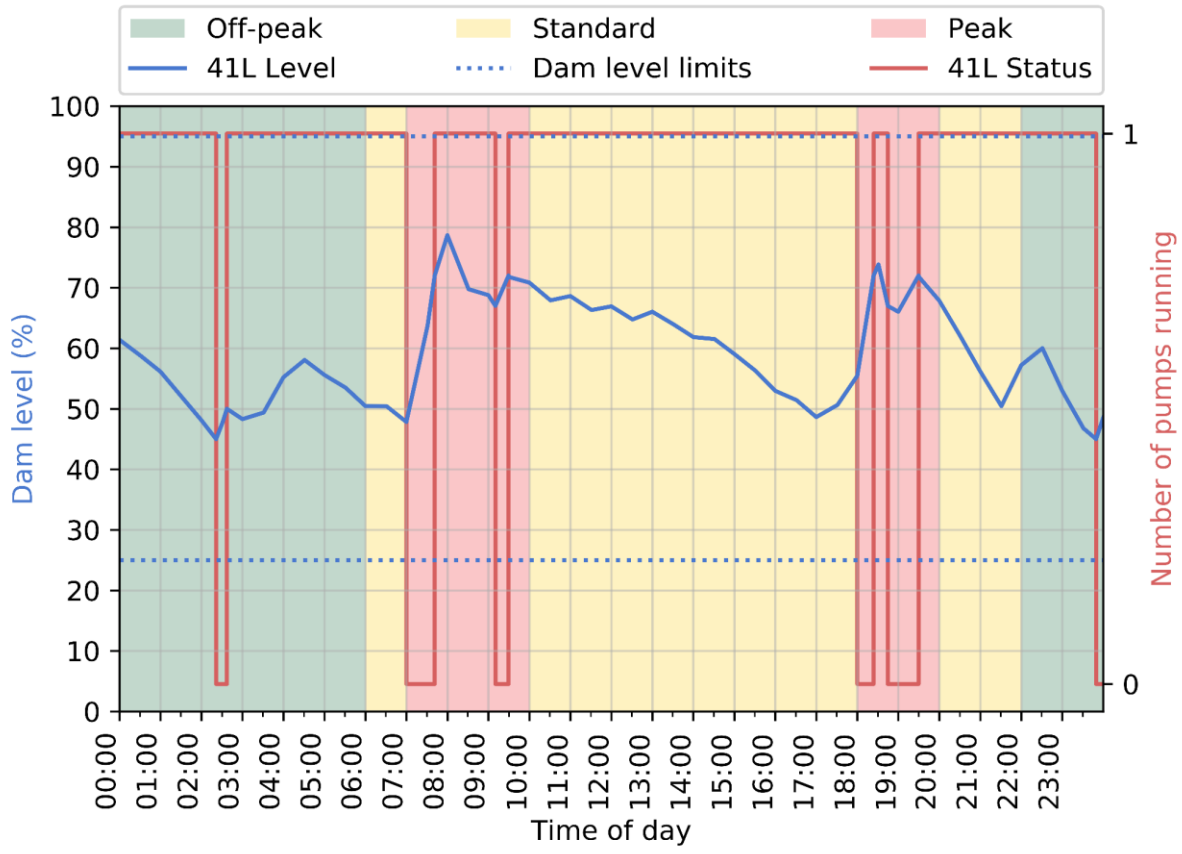


FIGURE 58 – CS3 1-FACTOR SIMULATION RESULTS: 41L DAM LEVEL AND PUMP STATUSES

Figure 59 (on the next page) shows that pump cycling occurred on 31L (similar to 41L). It was possible to switch off all pumps during the Eskom evening peak period. The 1-factor control system was able to control the 31L dam level within its control range.

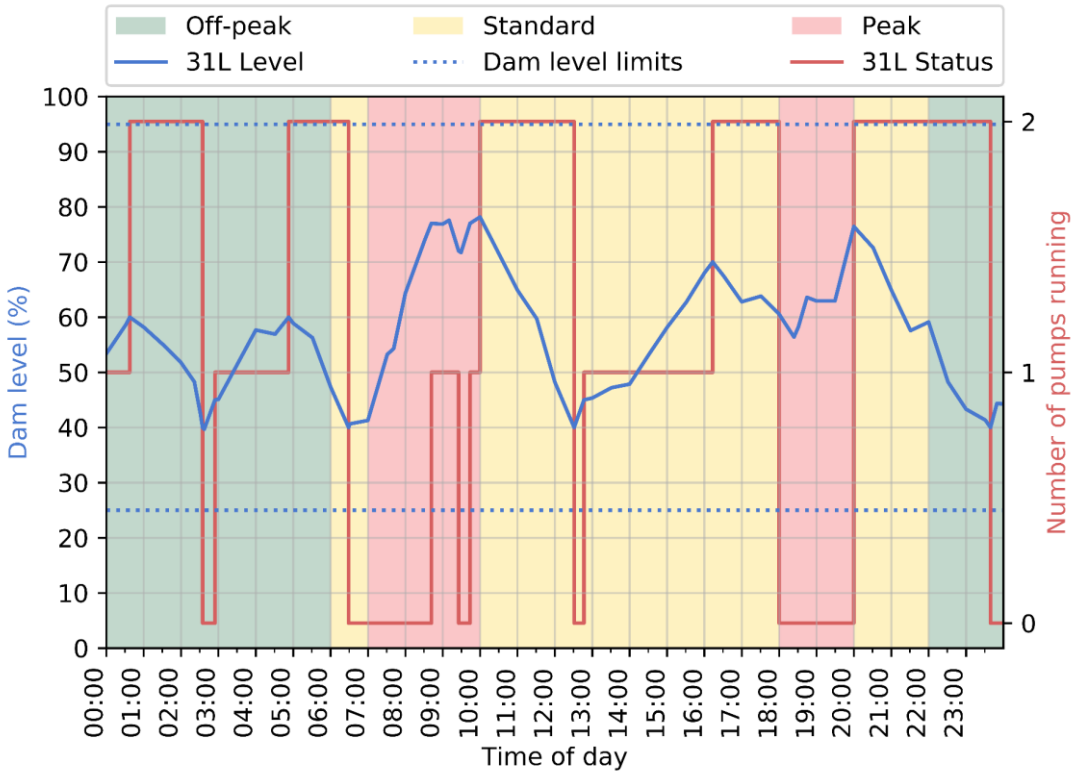


FIGURE 59 – CS3 1-FACTOR SIMULATION RESULTS: 31L DAM LEVEL AND PUMP STATUSES

At 20-level, it is noticed that pump cycling occurred shortly after all pumps were switched off (Figure 60). For most of the evening peak period, the control system was able to switch off all running pumps. The 20-level dam overflowed around 02:00 and from 17:00 to 18:00.

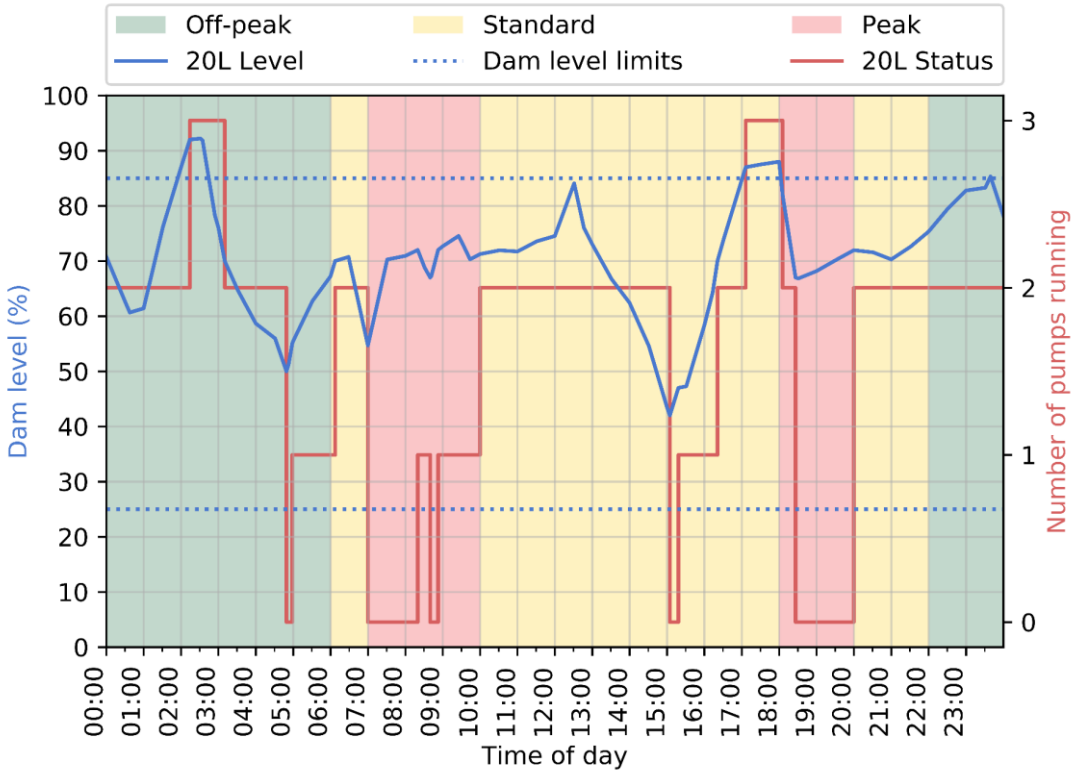


FIGURE 60 – CS3 1-FACTOR SIMULATION RESULTS: 20L DAM LEVEL AND PUMP STATUSES

At IM-level, pump cycling occurred before the morning peak period and during the evening peak period (Figure 61). The cycling occurred when all running pumps were switched off. Figure 61 reveals that the IM-level dam level was controlled within control ranges. At the start of the morning peak period, the control system switched off all pumps. Pumps were kept off for most of the morning peak period.

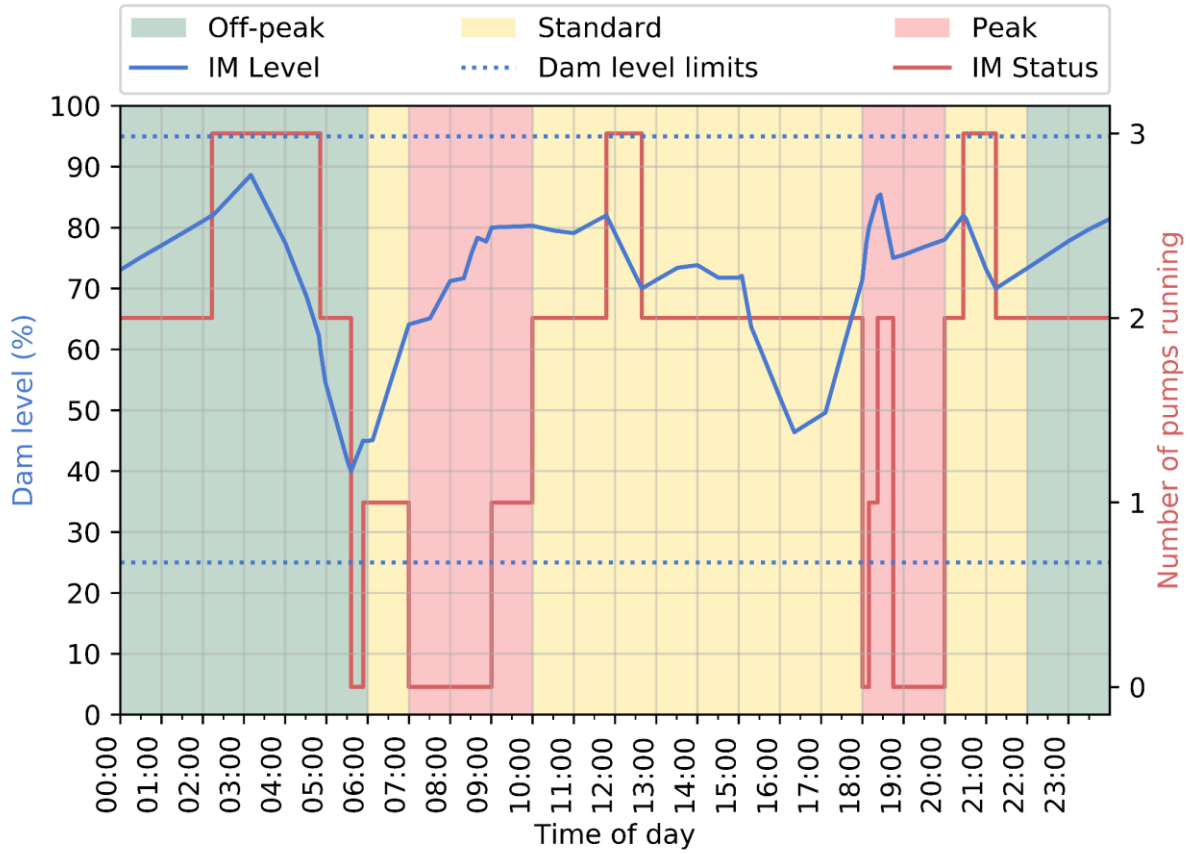


FIGURE 61 – CS3 1-FACTOR SIMULATION RESULTS: IM DAM LEVEL AND PUMP STATUSES

Figure 62 illustrates the 1-factor control system's effect on the surface dam level. It is seen that the surface dam overflowed before morning peak period.

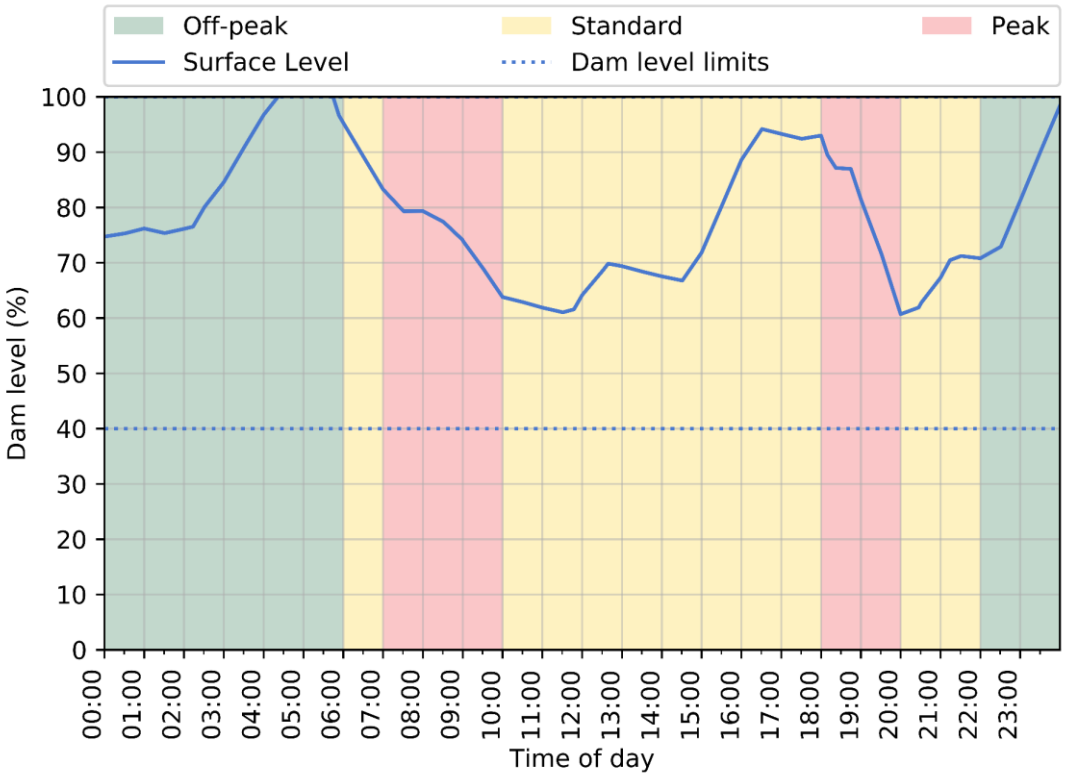


FIGURE 62 – CS3 1-FACTOR SIMULATION RESULTS: SURFACE DAM LEVEL

Although this system is heavily constrained, the control system was able to perform morning and evening load-shifts (Figure 63). Using the Eskom Megaflex ≤300 km, 500 V – 66 kV tariff, the BPC for the operation of this day is R185 691.93. The SPC for this day is R258 259.71.

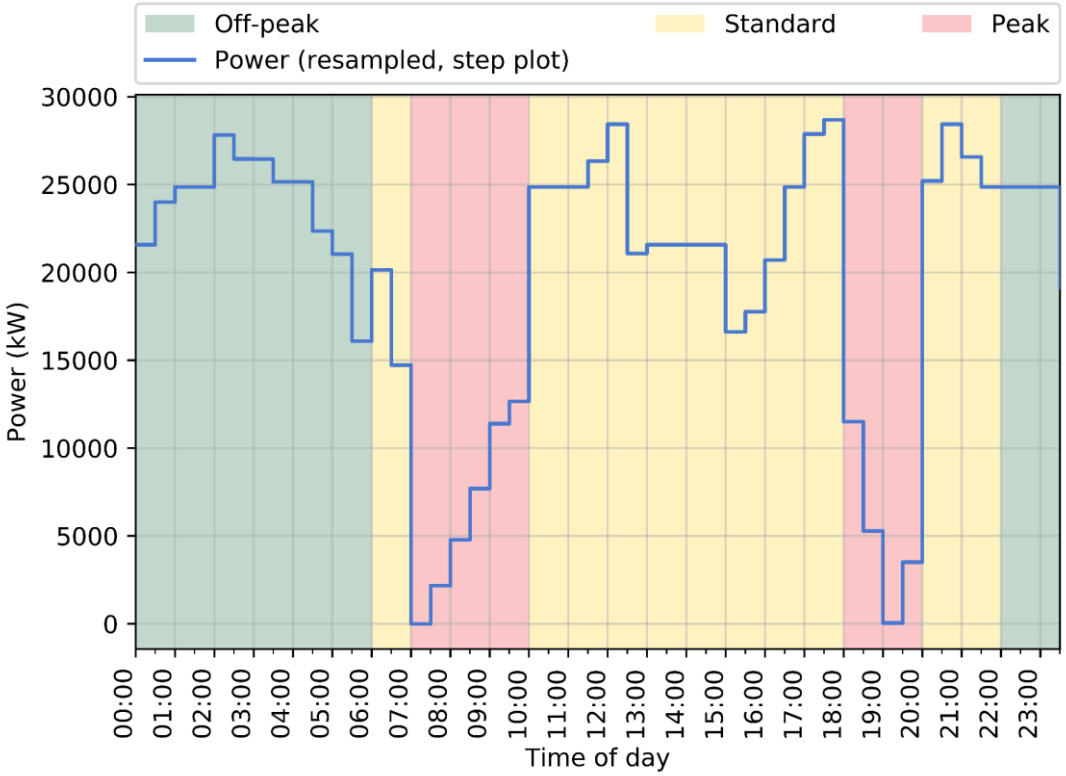


FIGURE 63 – CS3 1-FACTOR SIMULATION RESULTS: POWER CONSUMPTION (RESAMPLED DATA)

2-FACTOR CONTROL SIMULATION

As with the previous case studies, the differences between the 1-factor and 2-factor control systems become known when the downstream dam reaches ≥95% (as shown in the decision-making algorithm, Figure 13, page 36). Because of the simulated day at CS3 being highly constrained, this effect is present in CS3's 2-factor simulation.

41L, 31L and 20L all have the same results as with the 1-factor simulation (Figures 58, 59, and 60) and results are not repeated in this section.

At IM-level, the dam level is controlled within its penalisation limits, but control-system induced pump cycling occurs (Figure 64). Pump cycling occurs whenever all pumps are switched off. To prevent the surface dam from overflowing (just before 04:30) (Figure 65), the control system schedules fewer pumps at IM-level.

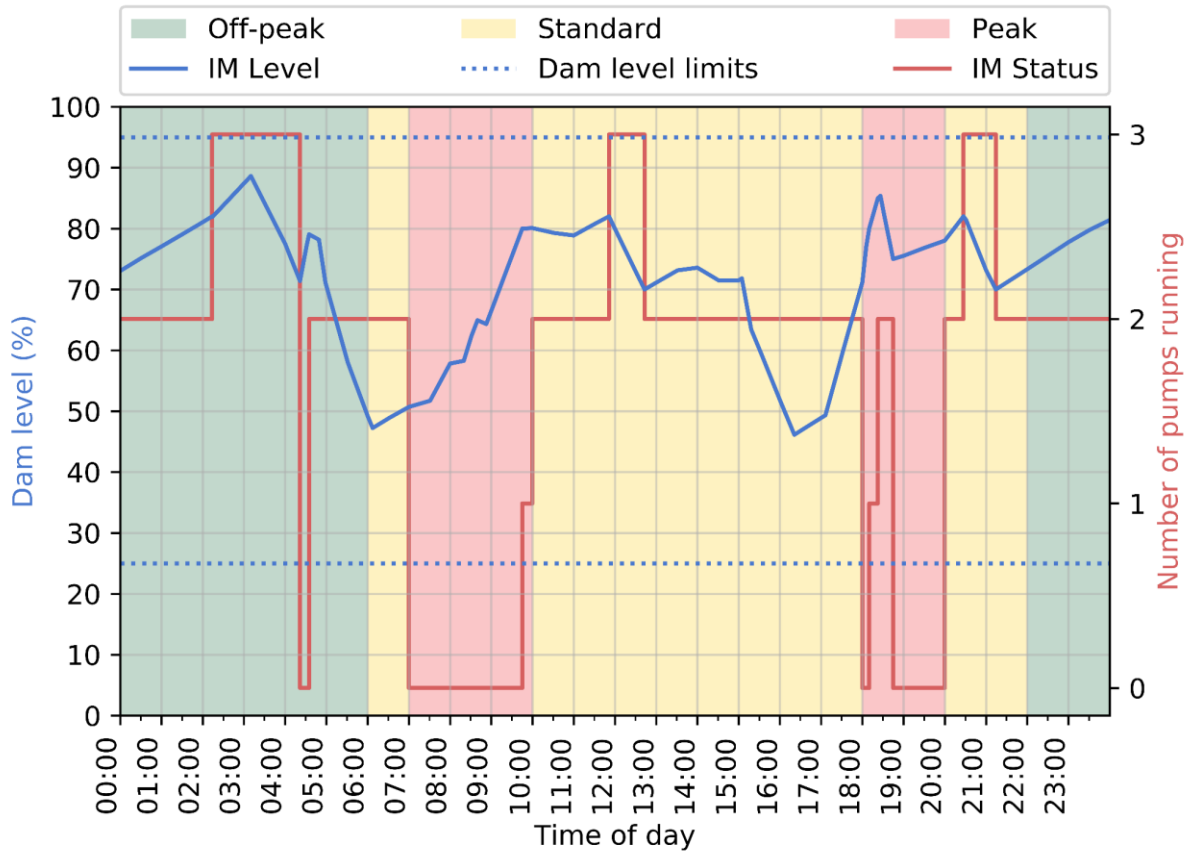


FIGURE 64 – CS3 2-FACTOR SIMULATION RESULTS: IM DAM LEVEL AND PUMP STATUSES

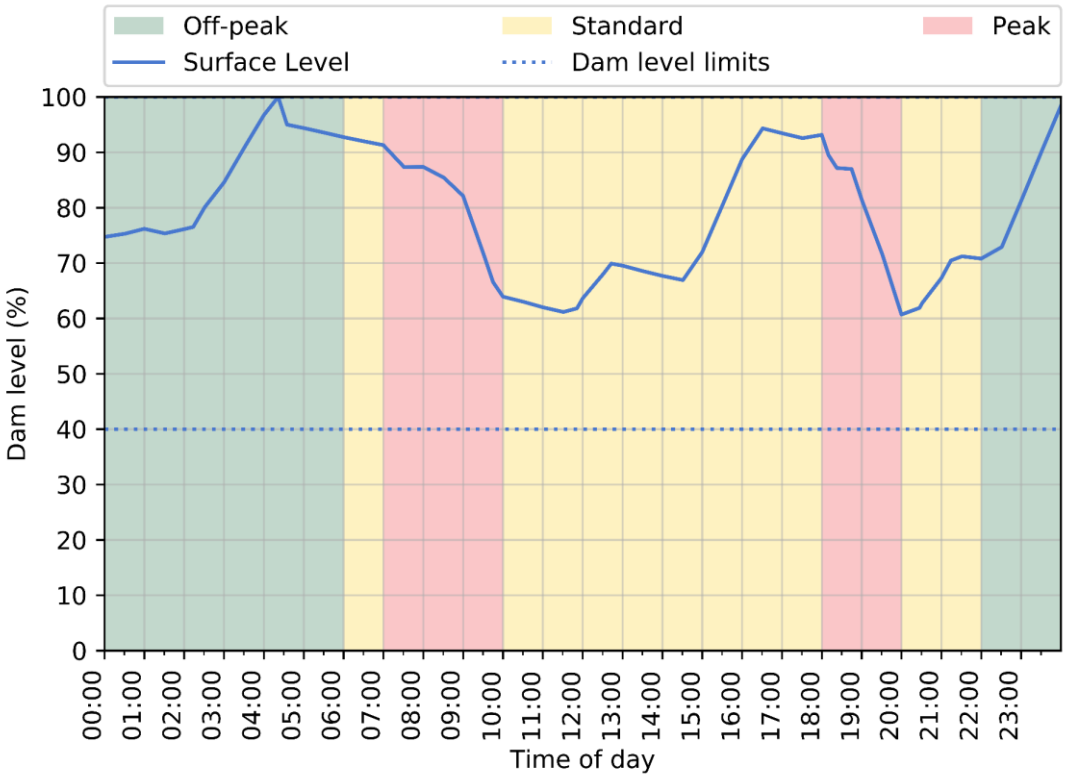


FIGURE 65 – CS3 2-FACTOR SIMULATION RESULTS: SURFACE DAM LEVEL

Although this system is heavily constrained, the control system was able to perform morning and evening load-shifts (Figure 66). Using the Eskom Megaflex ≤300 km, 500 V – 66 kV tariff, the BPC for the operation of this day is R185 691.93, whereas the SPC is R257 700.40.

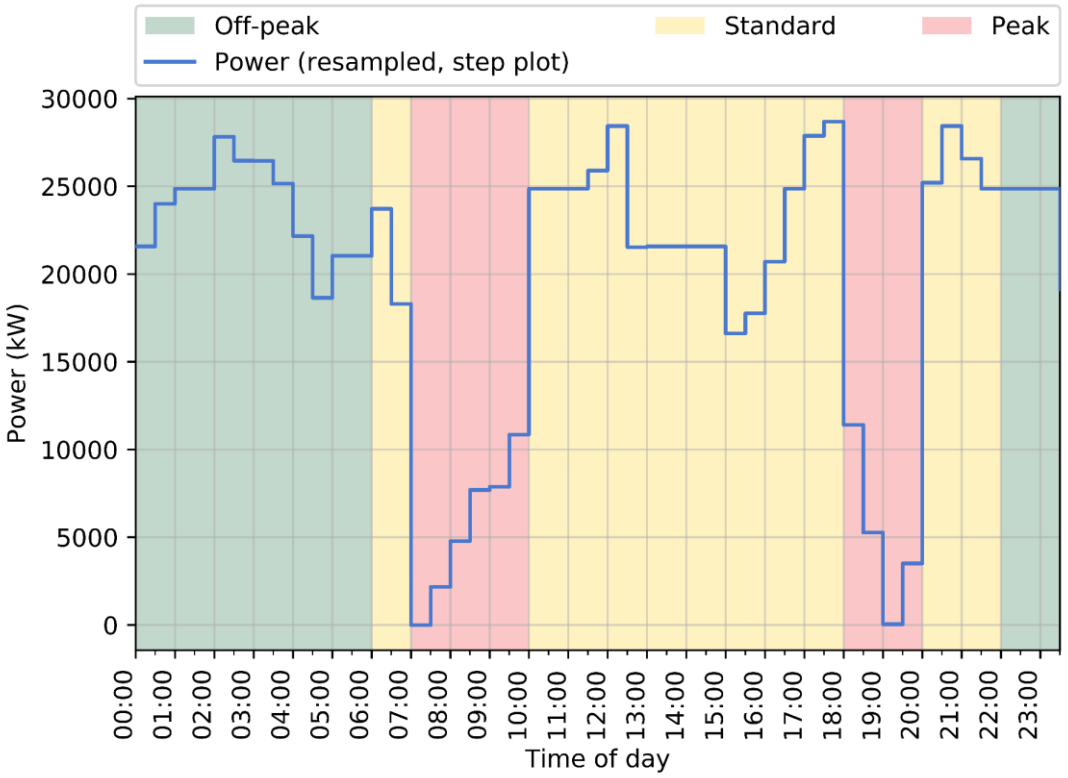


FIGURE 66 – CS3 2-FACTOR SIMULATION RESULTS: POWER CONSUMPTION (RESAMPLED DATA)

Applying the n-factor control system to the CS3 dewatering system allows for additional logic to be built into the control philosophy. 41L's dam level was controlled within range and no pump cycling occurred (pump statuses are not changed within the time span of less than an hour) (Figure 67).

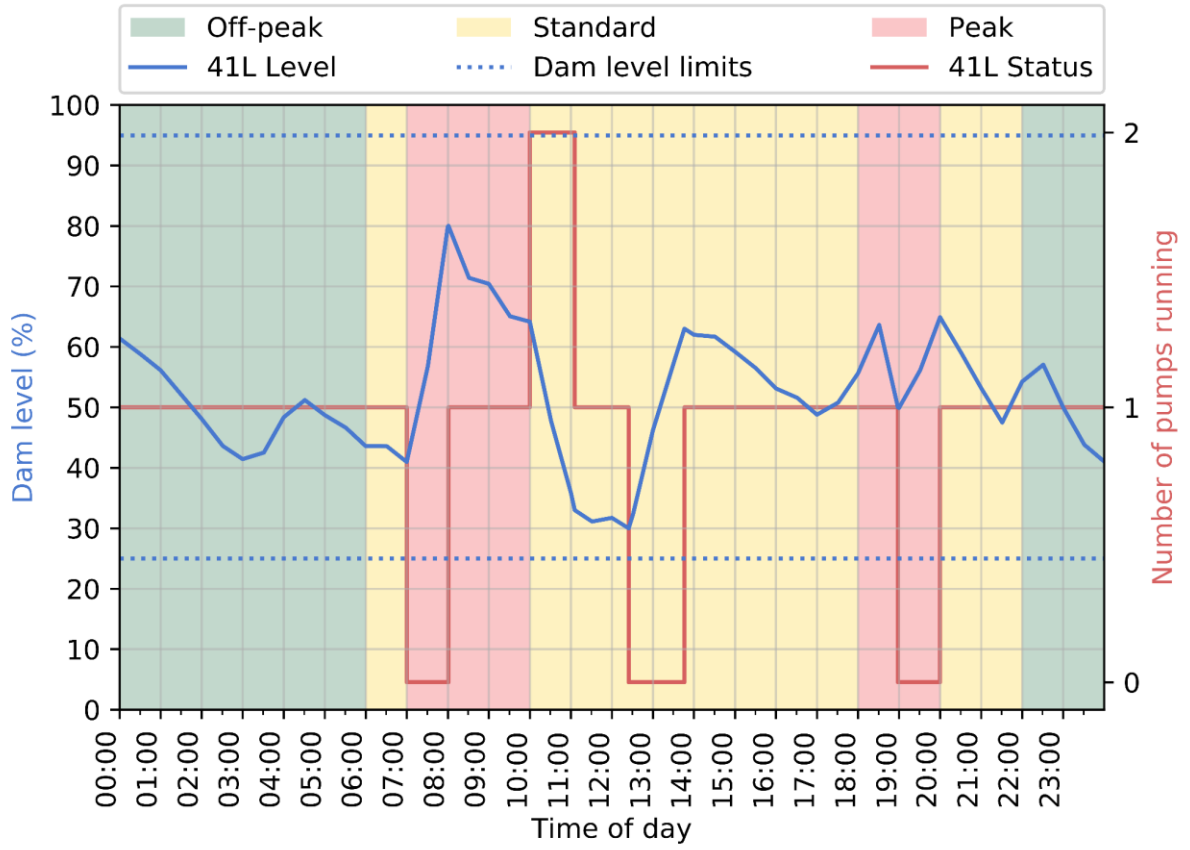


FIGURE 67 – CS3 n-FACTOR SIMULATION RESULTS: 41L DAM LEVEL AND PUMP STATUSES

Figure 68 (on the next page) shows that, the control system was able to perform a morning load-shift on the pumps at 31L. Evening load-shift was skipped and one pump was kept running. 31L's dam level only barely exceeds the higher penalisation limit (by 0.38%) for a short moment after 11:00, but otherwise the dam level is controlled well.

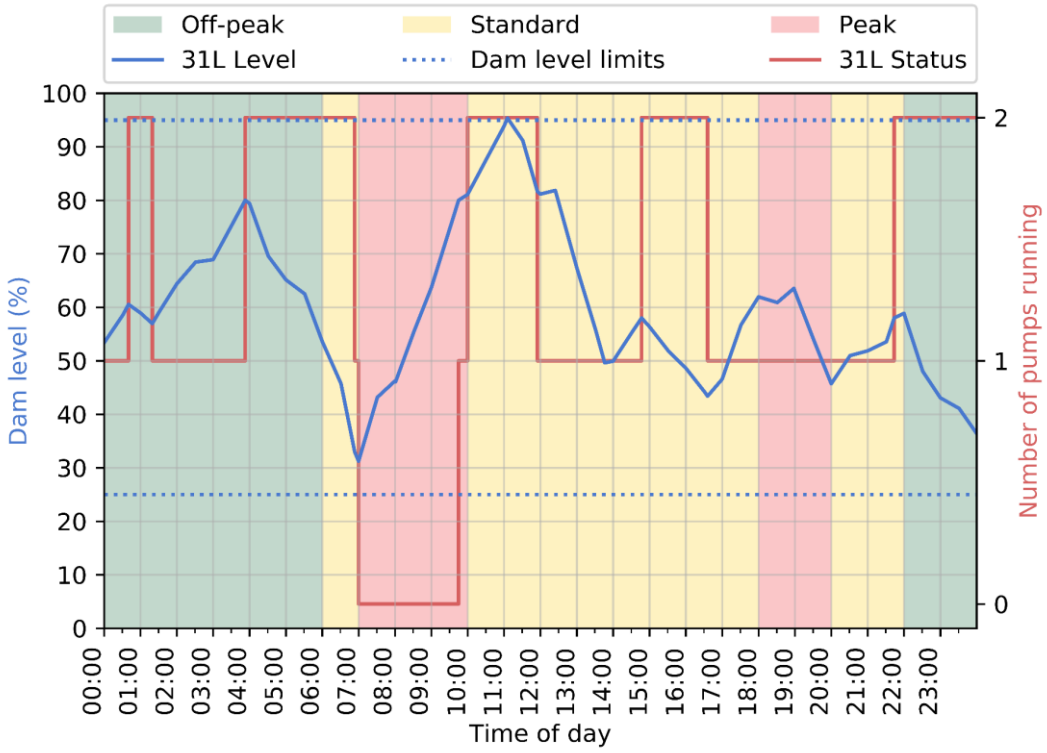


FIGURE 68 – CS3 n-FACTOR SIMULATION RESULTS: 31L DAM LEVEL AND PUMP STATUSES

Figure 69 indicates that the n-factor control system was able to control the 20L pumps without causing cycling. Pumps were kept switched off for most of the morning peak period. One pump was switched off during the evening peak period. The dam level is controlled within operating limits.

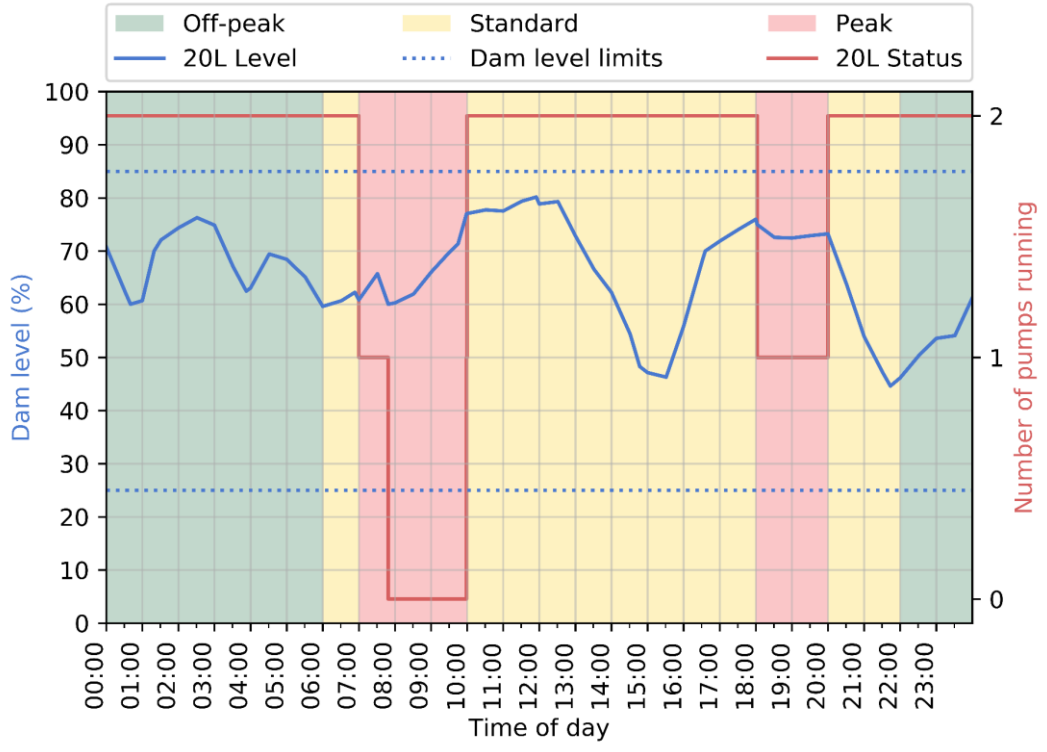


FIGURE 69 – CS3 n-FACTOR SIMULATION RESULTS: 20L DAM LEVEL AND PUMP STATUSES

Figure 70 indicates that the IM-level pumps where not stopped and started within a time span of less than an hour. The IM dam was controlled within operating range.

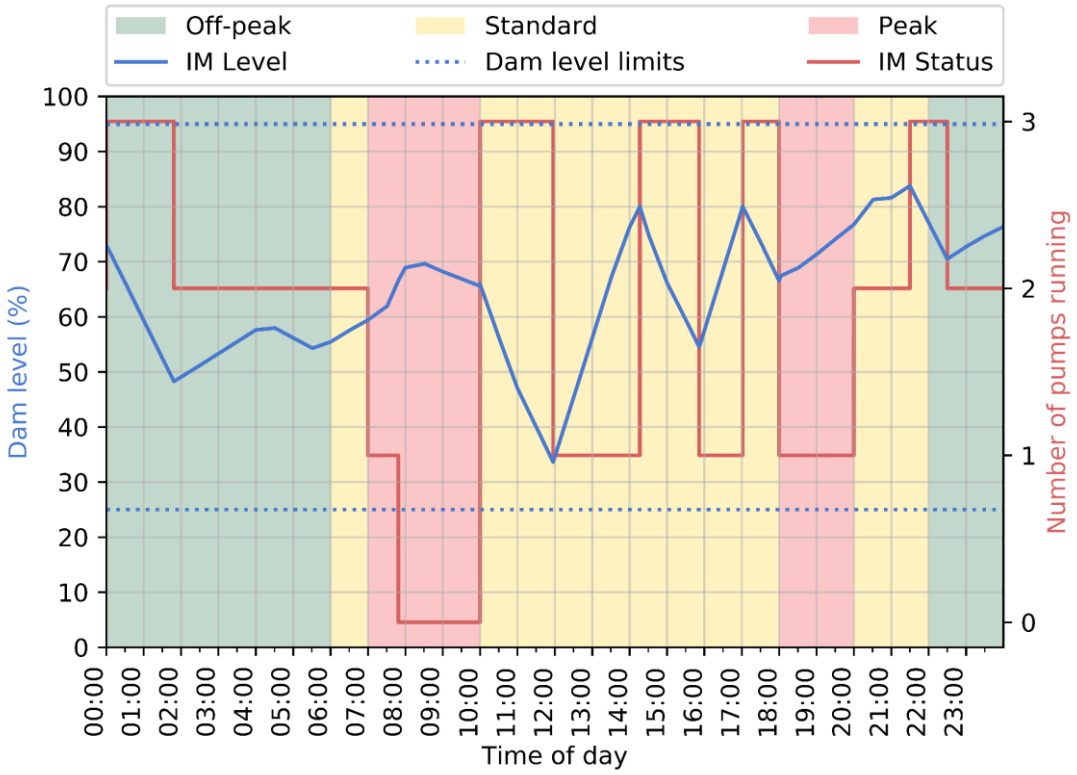


FIGURE 70 – CS3 n-FACTOR SIMULATION RESULTS: IM DAM LEVEL AND PUMP STATUSES

The surface dam overflowed near the end of the day (Figure 71).

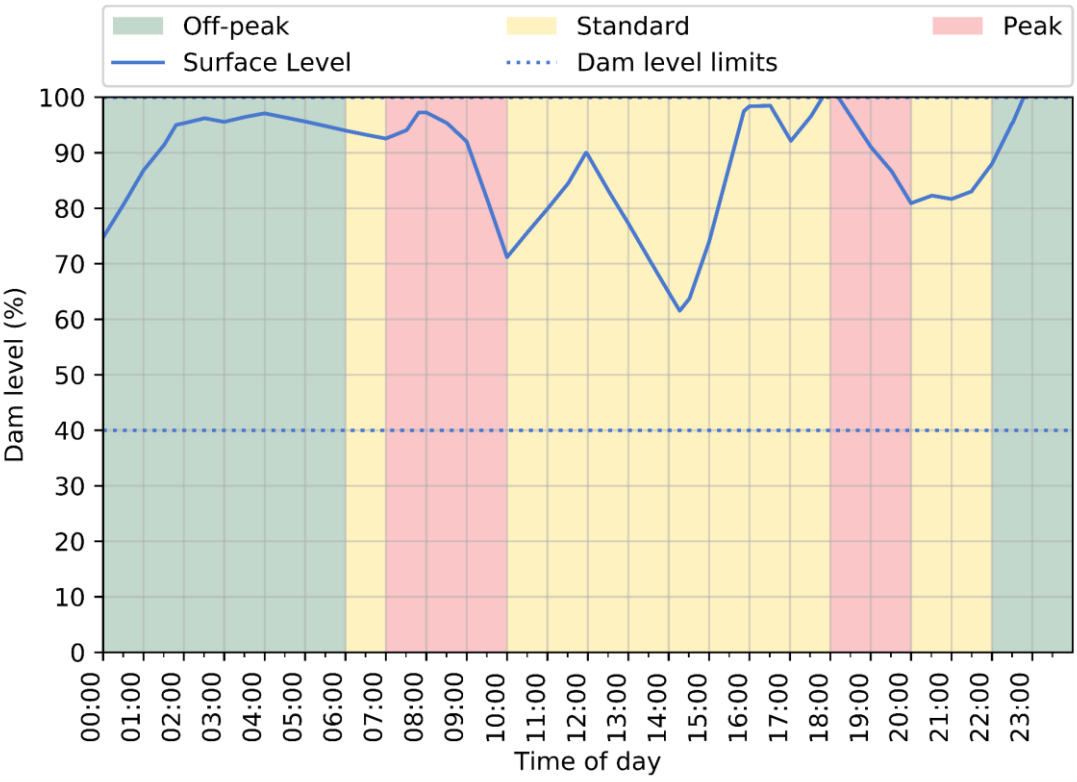


FIGURE 71 – CS3 n-FACTOR SIMULATION RESULTS: SURFACE DAM LEVEL

Figure 72 shows the total power consumption by the CS3 dewatering system, as obtained when controlled using the n-factor control system. It is noted that a significant morning load-shift was performed.

Using the Eskom Megaflex ≤300 km, 500 V – 66 kV tariff, the BPC for the operation of this day is R192 497.32, whereas the SPC is R270 042.86.

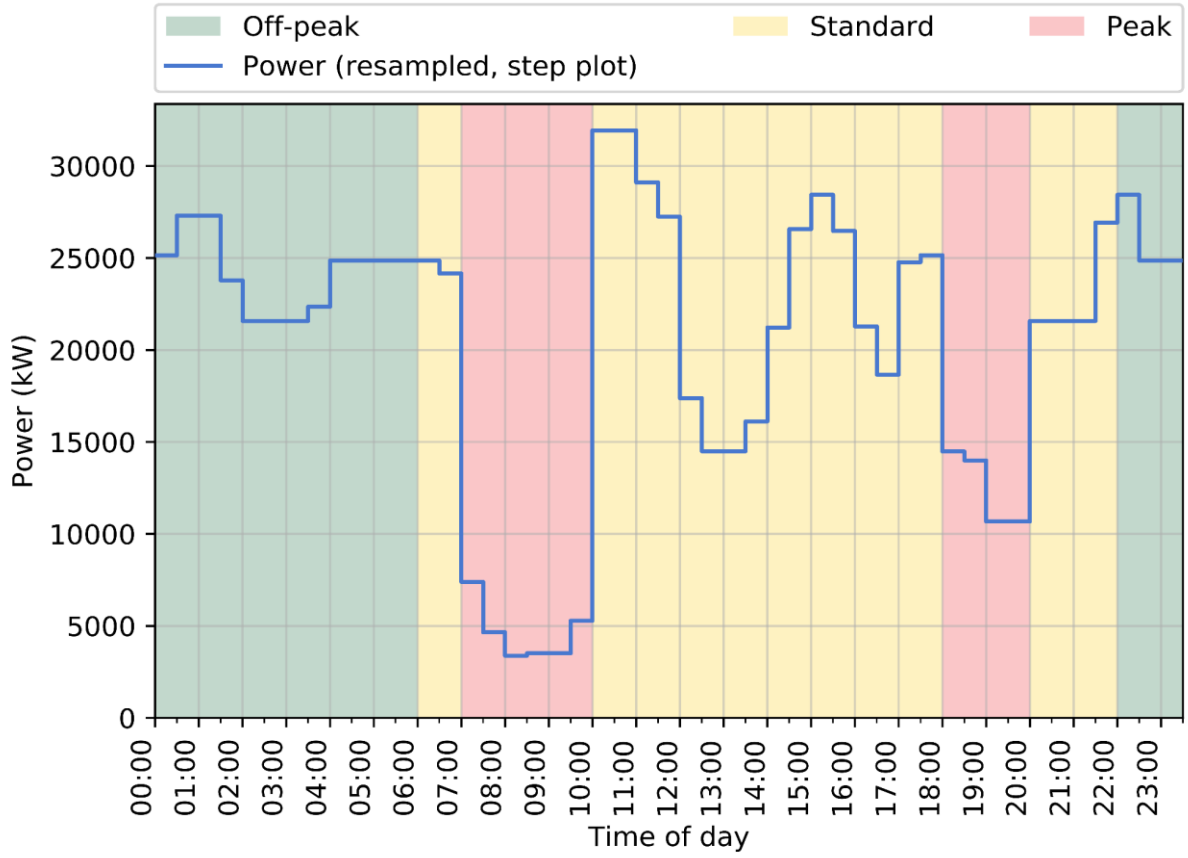


FIGURE 72 – CS3 n-FACTOR SIMULATION RESULTS: POWER CONSUMPTION

COMPARATIVE STUDY OF MINE
DEWATERING CONTROL SYSTEMS

APPENDIX E

MINE SCADA PUMP SCHEDULING ALGORITHM

APPENDIX E: MINE SCADA PUMP SCHEDULING ALGORITHM

```
{   scrGetPumpsRequired
    DESCRIPTION: Return the amount of required pumps to run
    VERSION: 3
    DATE: 2016-07-21
    CHANGES: Take into account the upper levels' dam levels, if it is full you should
             recommend to pump
}

dim req as integer;
dim nrOfPumps as integer;
dim pumpsRequiredNew as integer;
dim setpoint as float;
dim upperDamLevel as float;  ' Used to calculate the total dam level for the combined dams
                             on the upper level (downstream)
dim tempDamLevel as float;
dim counter1 as integer;

nrOfPumps = Me.UDA_Pumps_Available;  ' Another script determine the amount of pumps that
                                     have communcations
setpoint = 0;
upperDamLevel = 0;
tempDamLevel = 0;
counter1 = 0;

' Calculate the combined dam levels for the upper downstream level
if isgood(Me.FA_UL_DAM01) and (Me.FA_UL_DAM01 >= 0) and (Me.FA_UL_DAM01_MNT = False) then
    tempDamLevel = tempDamLevel + Me.FA_UL_DAM01;
    counter1 = counter1 + 1;
endif;
if isgood(Me.FA_UL_DAM02) and (Me.FA_UL_DAM02 >= 0) and (Me.FA_UL_DAM02_MNT = False) then
    tempDamLevel = tempDamLevel + Me.FA_UL_DAM02;
    counter1 = counter1 + 1;
endif;
if isgood(Me.FA_UL_DAM03) and (Me.FA_UL_DAM03 >= 0) and (Me.FA_UL_DAM03_MNT = False) then
    tempDamLevel = tempDamLevel + Me.FA_UL_DAM03;
    counter1 = counter1 + 1;
endif;
if isgood(Me.FA_UL_DAM04) and (Me.FA_UL_DAM04 >= 0) and (Me.FA_UL_DAM04_MNT = False) then
    tempDamLevel = tempDamLevel + Me.FA_UL_DAM04;
    counter1 = counter1 + 1;
endif;
if isgood(Me.FA_UL_DAM05) and (Me.FA_UL_DAM05 >= 0) and (Me.FA_UL_DAM05_MNT = False) then
    tempDamLevel = tempDamLevel + Me.FA_UL_DAM05;
    counter1 = counter1 + 1;
endif;
if isgood(Me.FA_UL_DAM06) and (Me.FA_UL_DAM06 >= 0) and (Me.FA_UL_DAM06_MNT = False) then
    tempDamLevel = tempDamLevel + Me.FA_UL_DAM06;
    counter1 = counter1 + 1;
endif;
if isgood(Me.FA_UL_DAM07) and (Me.FA_UL_DAM07 >= 0) and (Me.FA_UL_DAM07_MNT = False) then
    tempDamLevel = tempDamLevel + Me.FA_UL_DAM07;
    counter1 = counter1 + 1;
endif;
if isgood(Me.FA_UL_DAM08) and (Me.FA_UL_DAM08 >= 0) and (Me.FA_UL_DAM08_MNT = False) then
    tempDamLevel = tempDamLevel + Me.FA_UL_DAM08;
    counter1 = counter1 + 1;
endif;
if isgood(Me.FA_UL_DAM09) and (Me.FA_UL_DAM09 >= 0) and (Me.FA_UL_DAM09_MNT = False) then
    tempDamLevel = tempDamLevel + Me.FA_UL_DAM09;
    counter1 = counter1 + 1;
endif;
if isgood(Me.FA_UL_DAM10) and (Me.FA_UL_DAM10 >= 0) and (Me.FA_UL_DAM10_MNT = False) then
    tempDamLevel = tempDamLevel + Me.FA_UL_DAM10;
    counter1 = counter1 + 1;
endif;
```

```
if (tempDamlevel > 0) and (counter1 > 0)
    upperDamlevel = tempDamLevel/counter1;
endif;

if me.DEBUG then
    LogMessage("_____________________________________ " + me.tagname +
                " _____________________________________");
    logMessage("tempDamLevel: " + tempDamLevel);
    logMessage("counter1: " + counter1);
    logMessage("upperDamLevel: " + upperDamLevel);
    LogMessage("_____________________________________ END
                _____________________________________");
endif;

' Check if the combined upper dam level is full
if upperDamLevel >= Me.UDA_UL_HL then  ' Me.UDA_UL_HL can be set but is 100% by default
    Me.UDA_UL_100 = true;  ' True means that later on we set the required pumps to run to 0
endif;
' if upper dam levels came down from full to the lower limit (95%), reset the bit
if upperDamLevel <= Me.UDA_DL_LL then 'Hysteresis Me.UDA_UL_LL would probably be around 95%
    Me.UDA_DL_100 = false;  ' False mean that the pumps can now run again
endif;

'Calculate required pumps to run, only when upper levels' dam levels is not full
if me.UDA_DL_100 = False then
    'ESKOM PEAK PERIOD
    if Me.UDA_Scheduled_Period = "High" then
        for req = 1 to nrOfPumps step 1;
            if me.VirtualDam.FA PV >= me.UDA_Profile_High[req] then
                ' The Virtual Dam Level is more than the current setpoint, update pumps
                   required and setpoint
                pumpsRequiredNew = req;
                setpoint = me.UDA_Profile_High[req];
            endif;
            if me.VirtualDam.FA_PV < (me.UDA_Profile_High[1] - me.UDA_HYST_MIN) then
            ' The dam level is low enough for all pumps to stop running
                Me.UDA_Pumps_Required = 0;
                setpoint = UDA_Profile_High[1];
            endif;
        next;
        if me.UDA_Pumps_Required >= (pumpsRequiredNew + 2) then
        ' When the dam level is falling; then stop one pump at a time if more than 2 too many
           is running
           me.UDA_Pumps_Required = pumpsRequiredNew + 1;
        endif;
    endif;

    'ESKOM STANDARD PERIOD
    if Me.UDA_Scheduled_Period = "Mid" then
        for req = 1 to nrOfPumps step 1;
            if me.VirtualDam.FA PV >= me.UDA_Profile_Mid[req] then
                ' The Virtual Dam Level is more than the current setpoint, update pumps
                   required and setpoint
                pumpsRequiredNew = req;
                setpoint = me.UDA_Profile_Mid[req];
            endif;
            if me.VirtualDam.FA_PV < (me.UDA_Profile_Mid[1] - me.UDA_HYST_MIN) then
            ' The dam level is low enough for all pumps to stop running
                Me.UDA_Pumps_Required = 0;
                setpoint = UDA_Profile_Mid[1];
            endif;
        next;
        if me.UDA_Pumps_Required >= (pumpsRequiredNew + 2) then
        ' When the dam level is falling; then stop one pump at a time if more than 2 too many
           is running
           me.UDA_Pumps_Required = pumpsRequiredNew + 1;
        endif;
    endif;
```

```
    'ESKOM OFF-PEAK PERIOD
    if Me.UDA_Scheduled_Period = "Low" then
       for req = 1 to nrOfPumps step 1;
          if me.VirtualDam.FA PV >= me.UDA_Profile_Low[req] then
             ' The Virtual Dam Level is more than the current setpoint, update pumps
                required and setpoint
             pumpsRequiredNew = req;
             setpoint = me.UDA_Profile_Low[req];
          endif;
          if me.VirtualDam.FA_PV < (me.UDA_Profile_Low[1] - me.UDA_HYST_MIN) then
          ' The dam level is low enough for all pumps to stop running
             Me.UDA_Pumps_Required = 0;
             setpoint = UDA_Profile_Low[1];
          endif;
       next;
       if me.UDA_Pumps_Required >= (pumpsRequiredNew + 2) then
       ' When the dam level is falling; then stop one pump at a time if more than 2 too many
          is running
          me.UDA_Pumps_Required = pumpsRequiredNew + 1;
       endif;
    endif;

    'ESKOM PERIOD NOT DEFINED
    if Me.UDA_Scheduled_Period == "None" then
       Me.UDA_Pumps_Required = 0;
    endif;

    'Update the required nr of pumps to more pumps if required
    if pumpsRequiredNew > Me.UDA_Pumps_Required then
       Me.UDA_Pumps_Required = pumpsRequiredNew;
    endif;

    me.UDA_VDL_SP = setpoint;  ' Set the setpoint UDA equal to our temporary variable
'If upper dam levels are full
elseif me.UDA_UL_100 == true then ' Upper dam level is full so no pumps may run
    Me.UDA_Pumps_Required = 0;
    Me.UDA_VDL_SP = 0;
endif;

'Set the flag bit to indicate if a pump must start or stop, also used to change the
colouring on screen
if Me.UDA_Pumps_Required > Me.UDA_Pumps_Running then
    Me.UDA_Start_Required = true;
else
    Me.UDA_Start_Required = false;
endif;

if Me.UDA_Pumps_Required < Me.UDA_Pumps_Running then
    Me.UDA_Start_Required = true;
else
    Me.UDA_Start_Required = false;
endif;
```

COMPARATIVE STUDY OF MINE
DEWATERING CONTROL SYSTEMS

APPENDIX F

DATA PROCESSING AND SIMULATION CODE

APPENDIX F: DATA PROCESSING AND SIMULATION CODE

This section contains an example of the data processing that was performed in this study. The source code of the simulations is also provided. All source code is also available electronically on GitHub (http://bit.ly/2rqp9IE), which provides an optimal browsing experience of data processing Jupyter notebooks and source code.

APPENDIX CONTENTS

F.1. Directory structure

The following directory structure describes the layout and location of files as used in the study:

```
│
├──data processing
│   │   holidays.csv
│   │
│   ├──Case_study_1
│   │   │   CS1 data 1 min_2017-09-19.pdi.gz
│   │   │   CS1 data 1s 05-31_2017-09-27_pivot.csv.gz
│   │   │   CS1 data processing.ipynb
│   │   │   CS1 scoring calculator.xlsx
│   │   │   scoring_results.csv
│   │   │
│   │   └──output
│   │           CS1_resampled_power_1_factor.csv
│   │           CS1_resampled_power_2_factor.csv
│   │           CS1_resampled_power_n_factor.csv
│   │
│   ├──Case_study_2
│   │   │   CS2 data 1min_2017-09-26.pdi.gz
│   │   │   CS2 data 1s 10-25_2017-09-26_pivot.csv.gz
│   │   │   CS2 data processing.ipynb
│   │   │   CS2 scoring calculator.xlsx
│   │   │   scoring_results.csv
│   │   │
│   │   └──output
│   │           CS2_resampled_power_1_factor.csv
│   │           CS2_resampled_power_2_factor.csv
│   │           CS2_resampled_power_n_factor.csv
│   │
│   └──Case_study_3
│       │   CS3 data 1min 09-22_2017-10-11.pdi.gz
│       │   CS3 data 1s 09-22_2017-10-11_pivot.csv.gz
│       │   CS3 data processing.ipynb
│       │   CS3 scoring calculator.xlsx
│       │   scoring_results.csv
│       │
│       └──output
│               K7_resampled_power_1_factor.csv
│               K7_resampled_power_2_factor.csv
│               K7_resampled_power_n_factor.csv
│
├──modules
│       pumpingsystem.py
│       __init__.py
│
└──simulations
    ├──Case_study_1
    │   │   simulation_CS1.py
    │   │
    │   ├──input
    │   │       CS1_dam_inflow_profiles.csv.gz
    │   │       CS1_data_for_validation.csv.gz
    │   │
    │   └──output
    │           CS1_simulation_data_export_1-factor.csv.gz
    │           CS1_simulation_data_export_2-factor.csv.gz
    │           CS1_simulation_data_export_n-factor.csv.gz
    │           CS1_simulation_data_export_validation.csv.gz
    │
    ├──Case_study_2
    │   │   K10_simulation_data_export_1-factor new.csv
    │   │   simulation_CS2.py
    │   │
    │   ├──input
    │   │       CS2_dam_inflow_profiles.csv.gz
```

```
    │       CS2_data_for_validation.csv.gz
    │
    └──output
            CS2_simulation_data_export_1-factor.csv.gz
            CS2_simulation_data_export_2-factor.csv.gz
            CS2_simulation_data_export_n-factor.csv.gz
            CS2_simulation_data_export_validation.csv.gz
    │
    ├──Case_study_3
    │   simulation_k7.py
    │
    ├──input
    │       CS3_dam_inflow_profiles.csv.gz
    │       CS3_data_for_validation.csv.gz
    │
    └──output
            CS3_simulation_data_export_1-factor.csv.gz
            CS3_simulation_data_export_2-factor.csv.gz
            CS3_simulation_data_export_n-factor.csv.gz
            CS3_simulation_data_export_validation.csv.gz
    │
    ├──Verification
    │
    ├──Part 1
    │       M2_dam_inflow_profiles.csv.gz
    │       simulation_M1.py
    │       simulation_M2.py
    │       simulation_M3.py
    │
    ├──Part 2
    │       simulation_D1.py
    │       simulation_D2.py
```

F.2. DATA PROCESSING EXAMPLE: CASE STUDY 1

Die following is an example of the data processing performed on raw and simulation data. Data processing was performed in Jupyter notebooks.

In [1]:
```
import matplotlib.dates as mdates
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy.stats as stats
import seaborn as sns
from datetime import timedelta
from matplotlib.ticker import MaxNLocator
%matplotlib inline
```

In [2]:
```
# plotting parameters
figure_size = (6.25984252, 4.1456693) # = 15.9, 10.53 cm

def rgb(rgb_255_tuple):
    return tuple(v/255 for v in rgb_255_tuple)

colour_op = rgb((13, 103, 53))
colour_s = rgb((254, 205, 8))
colour_p = rgb((237, 29, 36))

sns.set_palette('muted')
```

Raw data processing

Inflow profile calculation

In [3]:
```
df = pd.read_csv('CS1 data 1 min_2017-09-19.pdi.gz')

# also make forward filled
time_interval = pd.Timedelta(minutes=1)
df = df.pivot_table(values='Value', columns='TagName', index=(pd.to_datetime(df['Date'].str.cat(df['Time'], sep=' ')) - time_int
erval))
df.index.name = 'DateTime'

df['Time_30'] = df.index.floor('30min').time

df['EskomDayOfWeek'] = df.index.dayofweek + 1
# replace holidays with Eskom days
h_path = '..\\holidays.csv'
holidays = np.genfromtxt(h_path,  dtype=np.dtype('M'), delimiter=',', usecols=0)
holidays = list(holidays)
holidays_numbers = np.genfromtxt(h_path, dtype=int, delimiter=',', usecols=1)
holidays_numbers = list(holidays_numbers)
for h_n, h_d in zip(holidays_numbers, holidays):
    df.loc[df.index.date==h_d,'EskomDayOfWeek'] = h_n
# drop weekends
df = df[df['EskomDayOfWeek'] <= 5]
df.drop('EskomDayOfWeek', axis=1)

df.head()
```

Out[3]:

TagName	KDCE_S01_31INS01_00IFT#505.FA_PV	KDCE_S03_30INS00_00IFT#501.FA_PV	KDCE_S03_30INS00_00ILT#501.FA_PV	KDCE_S03_30INS
DateTime				
2017-01-03 00:00:00	190.613326	276.329897	40.915051	7.613364
2017-01-03 00:01:00	191.185310	276.667415	40.869027	7.525664
2017-01-03 00:02:00	191.852380	277.123861	40.811069	7.457779
2017-01-03 00:03:00	192.684271	276.717341	40.735534	7.387303
2017-01-03 00:04:00	193.303978	277.537946	40.648078	7.292373

In [4]:
```
df = df.rename(columns = {'KDCE_S03_30INS00_00ILT#501.FA_PV': '30L D1',
                          'KDCE_S03_30INS00_00ILT#502.FA_PV': '30L D2',
                          'KDCE_S03_43INS00_00ILT#503.FA_PV': '43L D1a',
                          'KDCE_S03_43INS00_01ILT#503.FA_PV': '43L D1b',
                          'KDCE_S03_43INS00_00ILT#504.FA_PV': '43L D2a',
                          'KDCE_S03_43INS00_01ILT#504.FA_PV': '43L D2b',
                          'KDCE_S03_30INS00_00IFT#501.FA_PV': 'Flow from 43L',
                          'KDCE_S01_31INS01_00IFT#505.FA_PV': 'Flow to K1'})
```

In [5]:
```
status_col_dict = {}
for c in df.columns:
    if '_Machine.FA_RF' in c: # pump status
        item = c.split('_')[2].split('PMP')
        level = item[0] + 'L'
        pump_num = 'P' + item[1][1]
        status_col_dict[c] = level + ' ' + pump_num

df = df.rename(columns = status_col_dict)
```

In [6]:
```
df.head()
```

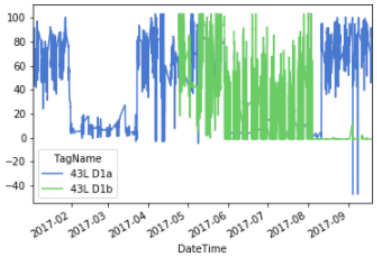
Out[6]:

TagName	Flow to K1	Flow from 43L	30L D1	30L D2	43L D1a	43L D2a	43L D1b	43L D2b	44L P1	44L P2	44L P3	44L P4	Time_30	EskomDayOfWeek
DateTime														
2017-01-03 00:00:00	190.613326	276.329897	40.915051	7.613364	79.410149	5.080543	NaN	NaN	1.0	1.0	0.0	0.0	00:00:00	2
2017-01-03 00:01:00	191.185310	276.667415	40.869027	7.525664	79.218411	5.095167	NaN	NaN	NaN	NaN	NaN	NaN	00:00:00	2
2017-01-03 00:02:00	191.852380	277.123861	40.811069	7.457779	78.861461	5.095167	NaN	NaN	1.0	1.0	0.0	0.0	00:00:00	2
2017-01-03 00:03:00	192.684271	276.717341	40.735534	7.387303	78.712480	5.082732	NaN	NaN	1.0	1.0	0.0	0.0	00:00:00	2
2017-01-03 00:04:00	193.303978	277.537946	40.648078	7.292373	78.354963	5.026379	NaN	NaN	1.0	1.0	0.0	0.0	00:00:00	2

Investigate the data

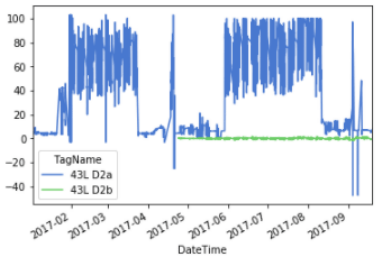
In [7]:
```
df[['43L D1a','43L D1b']].plot()
# use a
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x157b0deac88>
```



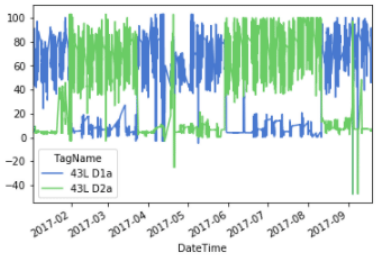
In [8]:
```
df[['43L D2a','43L D2b']].plot()
# use a
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x157ac59b048>
```



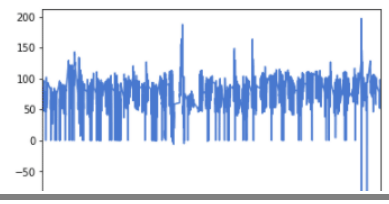
In [9]:
```
df[['43L D1a','43L D2a']].plot()
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x157b6fa47b8>
```



In [10]:
```
df[['43L D1a','43L D2a']].sum(axis=1).plot()
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x157bace1240>
```



Process selected data

In [11]:
```
df['44L Total pumps'] = df[[c for c in df.columns if '44L P' in c]].sum(axis=1, skipna=False)
df['44L Level'] = df[['43L D1a', '43L D2a']].sum(axis=1)  # new percentage on 10 ML, but 5 ML
df.drop(['30L D2', '43L D1a', '43L D2a', '43L D1b', '43L D2b', '30L D1', 'Flow to K1'], axis=1, inplace=True)
```

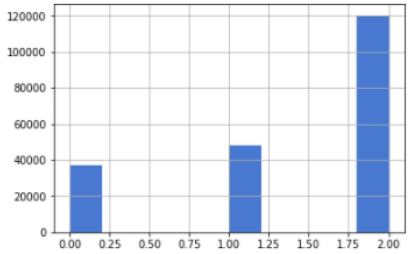
In [12]:
```
df.head()
```

Out[12]:

TagName	Flow from 43L	44L P1	44L P2	44L P3	44L P4	Time_30	EskomDayOfWeek	44L Total pumps	44L Level
DateTime									
2017-01-03 00:00:00	276.329897	1.0	1.0	0.0	0.0	00:00:00	2	2.0	84.490692
2017-01-03 00:01:00	276.667415	NaN	NaN	NaN	NaN	00:00:00	2	NaN	84.313578
2017-01-03 00:02:00	277.123861	1.0	1.0	0.0	0.0	00:00:00	2	2.0	83.956628
2017-01-03 00:03:00	276.717341	1.0	1.0	0.0	0.0	00:00:00	2	2.0	83.795212
2017-01-03 00:04:00	277.537946	1.0	1.0	0.0	0.0	00:00:00	2	2.0	83.381342

In [13]:
```
df['44L Total pumps'].hist()
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x157bd25e4e0>
```



In [14]:
```
flow_per_pump_if_1_run = df[df['44L Total pumps'] == 1]['Flow from 43L'].mean()
flow_per_pump_if_2_run = df[df['44L Total pumps'] == 2]['Flow from 43L'].mean()/2
count_per_pump_if_1_run = df[df['44L Total pumps'] == 1]['Flow from 43L'].count()
count_per_pump_if_2_run = df[df['44L Total pumps'] == 2]['Flow from 43L'].count()

print('Average flow per pump if one running = {:.1f} L/s'.format(flow_per_pump_if_1_run))
print('Average flow per pump if two running = {:.1f} L/s'.format(flow_per_pump_if_2_run))
ans = (flow_per_pump_if_1_run*count_per_pump_if_1_run + flow_per_pump_if_2_run*count_per_pump_if_2_run) / (count_per_pump_if_1_r
un+count_per_pump_if_2_run)
print('Weighted average flow per pump = {:.1f} L/s'.format(ans))
ans = float('{:.1f}'.format(ans))
print(ans)
```

```
Average flow per pump if one running = 168.7 L/s
Average flow per pump if two running = 133.0 L/s
Weighted average flow per pump = 143.0 L/s
143.0
```

In [15]:
```
def calculate_inflows(level_name, pump_flowrates, capacity, feeding_level_name=None):
    # feeding_level_name is the level feeding this level

    number_of_pumps = len(pump_flowrates)

    calc_pump_flows = [np.nan]
    calc_inflows = [np.nan]

    for i in range(1, len(df.index)):
        pump_status = {}
        for j,_ in enumerate(pump_flowrates):
            pump_str = 'P' + str(j+1)
            pump_status[pump_str] = df[level_name + ' ' + pump_str].values[i]

        l_new = df[level_name + ' Level'].values[i]
        l_old = df[level_name + ' Level'].values[i-1]
        pump_flow_from_prev_lev = 0 if feeding_level_name is None else df[feeding_level_name + ' PumpFlow'].values[i]

        if np.isnan([v for k,v in pump_status.items()] + [l_new] + [l_old] + [pump_flow_from_prev_lev]).any():
            ans_pumpflow = np.nan
            ans_inflow = np.nan

        delta_level = l_new - l_old
        if delta_level != 0:
            ans_outflow_pumps = 0
            for ps, pf in zip(pump_status.items(), pump_flowrates):
                ans_outflow_pumps += ps[1] * pf

            ans_inflow = ((l_new - l_old) / 100 * capacity) / 60 + ans_outflow_pumps - pump_flow_from_prev_lev
        else:
            ans_outflow_pumps = np.nan
            ans_inflow = np.nan

        calc_pump_flows.append(ans_outflow_pumps)
        calc_inflows.append(ans_inflow)
```

```
    df[level_name + ' PumpFlow'] = calc_pump_flows
    df[level_name + ' Inflow'] = calc_inflows
```

In [16]:
```
calculate_inflows('44L', [ans]*4, 5000000)
```

In [17]:
```
df.head()
```

Out[17]:

TagName	Flow from 43L	44L P1	44L P2	44L P3	44L P4	Time_30	EskomDayOfWeek	44L Total pumps	44L Level	44L PumpFlow	44L Inflow
DateTime											
2017-01-03 00:00:00	276.329897	1.0	1.0	0.0	0.0	00:00:00	2	2.0	84.490692	NaN	NaN
2017-01-03 00:01:00	276.667415	NaN	NaN	NaN	NaN	00:00:00	2	NaN	84.313578	NaN	NaN
2017-01-03 00:02:00	277.123861	1.0	1.0	0.0	0.0	00:00:00	2	2.0	83.956628	286.0	-11.458425
2017-01-03 00:03:00	276.717341	1.0	1.0	0.0	0.0	00:00:00	2	2.0	83.795212	286.0	151.486670
2017-01-03 00:04:00	277.537946	1.0	1.0	0.0	0.0	00:00:00	2	2.0	83.381342	286.0	-58.891881

Inflow profile

In [18]:
```
df2 = df.copy()

date_start = '2017-05-31'
date_end = '2017-06-01'

df2 = df2[(df2.index>=date_start) & (df2.index<date_end)]

inflow_profiles = []
overall_day_completeness_count = 0
overall_day_completeness_max = 0
overall_completeness_count = 0
overall_completeness_max = 0
for l in ['44L']:
    grouped = df2.set_index('Time_30')[l + ' Inflow'].groupby('Time_30')

    grouped_mean = grouped.mean()
    inflow_profiles.append(grouped_mean)

    print('{} completeness: {:6.2f}% → {:6.2f}%'.format(l, grouped.count().sum()/48/30*100, grouped_mean.count()/48*100))#groupe
d.count()/30
    overall_day_completeness_count += grouped.count().sum()
    overall_day_completeness_max += 48*30
    overall_completeness_count += grouped_mean.count()
    overall_completeness_max += 48
print('                    ------   -------')
print('All completenes:  {:6.2f}% → {:6.2f}%'.format(overall_day_completeness_count/overall_day_completeness_max*100, overall_co
mpleteness_count/overall_completeness_max*100))

pd.DataFrame(inflow_profiles).T.to_csv('..\..\simulations\Case_study_1\input\CS1_dam_inflow_profiles.csv.gz', compression='gzip'
)
```

```
44L completeness: 100.00% → 100.00%
                  ------   -------
All completenes:  100.00% → 100.00%
```

Data for verification of the simulation

In [19]:
```
df_real = pd.read_csv('CS1 data 1s 05-31_2017-09-27_pivot.csv.gz')
df_real = df_real.set_index('DateTime')
df_real.index = pd.to_datetime(df_real.index)
df_real.head()
```

Out[19]:

	KDCE_S03_30INS00_00IFT#501.FA_PV	KDCE_S03_30INS00_00ILT#501.FA_PV	KDCE_S03_43INS00_00ILT#503.FA_PV	KDCE_S03_43INS
DateTime				
2017-05-31 00:00:00	260.783131	36.701481	4.192639	71.770134
2017-05-31 00:00:01	260.949493	36.701481	4.192639	71.770134
2017-05-31 00:00:02	261.012313	36.701481	4.192639	71.681156
2017-05-31 00:00:03	261.015103	36.701481	4.192639	71.679736
2017-05-31 00:00:04	260.776504	36.633303	4.192639	71.753052

In [20]:
```
df_real = df_real.rename(columns = {'KDCE_S03_30INS00_00ILT#501.FA_PV': '30L D1',
```

```
                                    'KDCE_S03_43INS00_00ILT#504.FA_PV': '43L D2a',
                                    'KDCE_S03_30INS00_00IFT#501.FA_PV': 'Flow from 43L'})

status_col_dict = {}
for c in df_real.columns:
    if '_Machine.FA_RF' in c: # pump status
        item = c.split('_')[2].split('PMP')
        level = item[0] + 'L'
        pump_num = 'P' + item[1][1]
        status_col_dict[c] = level + ' ' + pump_num
df_real = df_real.rename(columns = status_col_dict)

df_real['44L Status'] = df_real[[c for c in df_real.columns if '44L P' in c]].sum(axis=1, skipna=False)

df_real = df_real.rename(columns = {'30L D1': '30L Level'})
df_real['44L Level'] = df_real[['43L D1a', '43L D2a']].sum(axis=1)  # new percentage on 10 ML

df_real.drop(['43L D1a', '43L D2a', 'Flow from 43L', '30L Level'], axis=1, inplace=True)

df_real.head()
```

Out[20]:

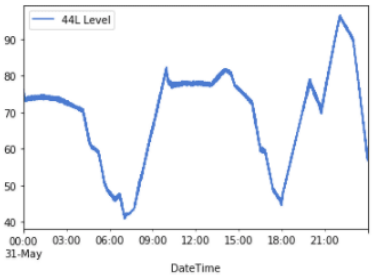
	44L P1	44L P2	44L P3	44L P4	44L Status	44L Level
DateTime						
2017-05-31 00:00:00	0.0	1.0	0.0	1.0	2.0	75.962773
2017-05-31 00:00:01	0.0	1.0	0.0	1.0	2.0	75.962773
2017-05-31 00:00:02	0.0	1.0	0.0	1.0	2.0	75.873795
2017-05-31 00:00:03	0.0	1.0	0.0	1.0	2.0	75.872375
2017-05-31 00:00:04	0.0	1.0	0.0	1.0	2.0	75.945691

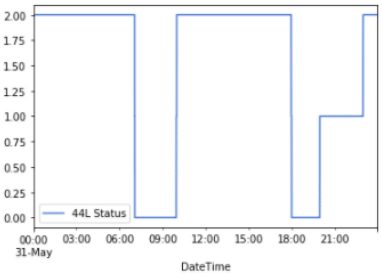
In [21]:
```
date_range_start = '2017-05-31'
date_range_end = '2017-06-01'

level_list = ['44L']
level_tag_list = [l + ' Level' for l in level_list]
status_tag_list = [l + ' Status' for l in level_list]
tag_list = level_tag_list + status_tag_list

df_real2 = df_real.ix[date_range_start:date_range_end]

print('Data completeness:', df_real2[tag_list].count(axis=0).sum()/(2 * 60*60*24))

df_real2[level_tag_list].plot()
df_real2[status_tag_list].plot()
```

```
C:\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  if __name__ == '__main__':
```

```
Data completeness: 1.0
```

Out[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x157ba0157f0>
```





In [22]:
```
df_real2['44L Status'] = df_real2['44L Status'].interpolate('pchip')
df_real2['44L Level'] = df_real2['44L Level'].interpolate('pchip')
```

Save simulation testing data

In [23]:
```
df_real2[['44L Status', '44L Level']].to_csv('..\..\simulations\Case_study_1\input\CS1_data_for_verification.csv.gz', compressio
```

```
n='gzip')
```

Now perform the simulation in verification mode

If the simulation outputs are reasonably "the same" as the real data, the simulation is "correct"

In [24]:
```
df_sim = pd.read_csv('..\..\simulations\Case_study_1\output\CS1_simulation_data_export_verification.csv.gz')

df_sim['time_clean'] = pd.to_datetime(df_sim['seconds'], unit='s') + timedelta(days=17317)
df_sim.head()
```

Out[24]:

	seconds	44L Level	44L Status	Eskom ToU	Pump system total power	time_clean
0	0	75.962773	2.0	3	3800.0	2017-05-31 00:00:00
1	1	75.961277	2.0	3	3800.0	2017-05-31 00:00:01
2	2	75.959780	2.0	3	3800.0	2017-05-31 00:00:02
3	3	75.958283	2.0	3	3800.0	2017-05-31 00:00:03
4	4	75.956786	2.0	3	3800.0	2017-05-31 00:00:04

In [25]:
```
def rmse(real, sim):
    return np.sqrt(((real - sim) ** 2).mean())

def r2(x, y):
    return stats.pearsonr(x,y)[0] ** 2
```

Comparison of simulation and actual data

44L pump status

In [26]:
```
x = pd.to_datetime(df_real2.index.values).time
y1 = df_real2['44L Status'].values
y2 = df_sim['44L Status'].values

print('R squared = ', r2(y1, y2))
print('RMSE = ', rmse(y1, y2))
```

```
R squared =  1.0
RMSE =  0.0
```

In [27]:
```
fig, ax1 = plt.subplots(figsize=(figure_size[0], figure_size[1]/1.5), dpi=1200)

x = df_sim['time_clean']

ax1.plot(x, df_real2['44L Status'], marker=None, label='Actual data', zorder=10)
ax1.plot(x, df_sim['44L Status'], marker=None, label='Simulation data', zorder=11, color=sns.color_palette('muted')[2])

ax1.set_xlabel('Time of day')
ax1.set_ylabel('Number of pumps running')
ax1.set_ylim([0, 4])
ax1.yaxis.set_major_locator(MaxNLocator(integer=True))

ax1.axvspan('2017-05-31 00:00:00','2017-05-31 05:59:59', alpha=.25, color=colour_op, lw=0, zorder=0, label='Off-peak')
ax1.axvspan('2017-05-31 06:00:00','2017-05-31 06:59:59', alpha=.25, color=colour_s, lw=0, zorder=0, label='Standard')
ax1.axvspan('2017-05-31 07:00:00','2017-05-31 09:59:59', alpha=.25, color=colour_p, lw=0, zorder=0, label='Peak')
ax1.axvspan('2017-05-31 10:00:00','2017-05-31 17:59:59', alpha=.25, color=colour_s, lw=0, zorder=0)
ax1.axvspan('2017-05-31 18:00:00','2017-05-31 19:59:59', alpha=.25, color=colour_p, lw=0, zorder=0)
ax1.axvspan('2017-05-31 20:00:00','2017-05-31 21:59:59', alpha=.25, color=colour_s, lw=0, zorder=0)
ax1.axvspan('2017-05-31 22:00:00','2017-05-31 23:59:59', alpha=.25, color=colour_op, lw=0, zorder=0)

ax1.xaxis.set_major_locator(mdates.HourLocator())
ax1.xaxis.set_minor_locator(mdates.MinuteLocator(byminute=30))
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
fig.autofmt_xdate(rotation=90)
ax1.set_xlim((min(x),max(x)))

ax1.grid(which='major', alpha=0.5)
#ax1.grid(which='minor', alpha=0.25)

handles, labels = ax1.get_legend_handles_labels()
order = [2, 0, 3, 1, 4]
ax1.legend([handles[idx] for idx in order],[labels[idx] for idx in order], bbox_to_anchor=(0., 1.02, 1., .102), loc=3, ncol=3, m
ode='expand', borderaxespad=0.)

fig.tight_layout()
plt.show()

fig.savefig('output/CS1_verification_44l_status.png', bbox_inches='tight')

plt.close('all')
```

44L dam level

In [28]:
```
x = pd.to_datetime(df_real2.index.values).time
y1 = df_real2['44L Level'].values
y2 = df_sim['44L Level'].values

print('R squared = ', r2(y1, y2))
print('RMSE = ', rmse(y1, y2))
```

```
R squared =  0.996373742526
RMSE =  1.24991693065
```

In [29]:
```
fig, ax1 = plt.subplots(figsize=figure_size, dpi=1200)

x = df_sim['time_clean']

ax1.plot(x, df_real2['44L Level'], marker=None, label='Actual data', zorder=10)
ax1.plot(x, df_sim['44L Level'], marker=None, label='Simulation data', zorder=11, color=sns.color_palette('muted')[2])

ax1.set_xlabel('Time of day')
ax1.set_ylabel('Dam level (%)')
ax1.set_ylim([0, 100])

ax1.axvspan('2017-05-31 00:00:00','2017-05-31 05:59:59', alpha=.25, color=colour_op, lw=0, zorder=0, label='Off-peak')
ax1.axvspan('2017-05-31 06:00:00','2017-05-31 06:59:59', alpha=.25, color=colour_s, lw=0, zorder=0, label='Standard')
ax1.axvspan('2017-05-31 07:00:00','2017-05-31 09:59:59', alpha=.25, color=colour_p, lw=0, zorder=0, label='Peak')
ax1.axvspan('2017-05-31 10:00:00','2017-05-31 17:59:59', alpha=.25, color=colour_s, lw=0, zorder=0)
ax1.axvspan('2017-05-31 18:00:00','2017-05-31 19:59:59', alpha=.25, color=colour_p, lw=0, zorder=0)
ax1.axvspan('2017-05-31 20:00:00','2017-05-31 21:59:59', alpha=.25, color=colour_s, lw=0, zorder=0)
ax1.axvspan('2017-05-31 22:00:00','2017-05-31 23:59:59', alpha=.25, color=colour_op, lw=0, zorder=0)

ax1.xaxis.set_major_locator(mdates.HourLocator())
ax1.xaxis.set_minor_locator(mdates.MinuteLocator(byminute=30))
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
fig.autofmt_xdate(rotation=90)
ax1.set_xlim((min(x),max(x)))

ax1.grid(which='major', alpha=0.5)
#ax1.grid(which='minor', alpha=0.25)

handles, labels = ax1.get_legend_handles_labels()
order = [2, 0, 3, 1, 4]
ax1.legend([handles[idx] for idx in order],[labels[idx] for idx in order], bbox_to_anchor=(0., 1.02, 1., .102), loc=3, ncol=3, m
ode='expand', borderaxespad=0.)

fig.tight_layout()
plt.show()

fig.savefig('output/CS1_verification_44l_level.png', bbox_inches='tight')

plt.close('all')
```

Processing of x-factored simulation data

In [30]:
```
def tou_shade(ax, date):
    y_m_d = '{}-{:02}-{:02}'.format(date.year, date.month, date.day)
    ax.axvspan(y_m_d + ' 00:00:00', y_m_d + ' 05:59:59', alpha=.25, color=colour_op, lw=0, zorder=0, label='Off-peak')
    ax.axvspan(y_m_d + ' 06:00:00', y_m_d + ' 06:59:59', alpha=.25, color=colour_s, lw=0, zorder=0, label='Standard')
    ax.axvspan(y_m_d + ' 07:00:00', y_m_d + ' 09:59:59', alpha=.25, color=colour_p, lw=0, zorder=0, label='Peak')
    ax.axvspan(y_m_d + ' 10:00:00', y_m_d + ' 17:59:59', alpha=.25, color=colour_s, lw=0, zorder=0)
    ax.axvspan(y_m_d + ' 18:00:00', y_m_d + ' 19:59:59', alpha=.25, color=colour_p, lw=0, zorder=0)
    ax.axvspan(y_m_d + ' 20:00:00', y_m_d + ' 21:59:59', alpha=.25, color=colour_s, lw=0, zorder=0)
    ax.axvspan(y_m_d + ' 22:00:00', y_m_d + ' 23:59:59', alpha=.25, color=colour_op, lw=0, zorder=0)

def sim_plot(case_study, factor, mode, level_name, level_limits, x, y, y_lim=100, save_fig=False, chart_type_override=None):
    fig, ax1 = plt.subplots(figsize=figure_size, dpi=1200)

    if mode != 'power' and mode != 'power_raw':
        for l in level_limits:
            ax1.plot([x[0], x[len(x)-1]], [l, l], ls=':', color=sns.color_palette('muted')[0], label='Dam level limits')

        lines1 = ax1.plot(x, df[level_name + ' Level'])

        ax2 = ax1.twinx()
        lines2 = ax2.plot(x, df[level_name + ' Status'], color=sns.color_palette('muted')[2])

        tou_shade(ax2, x[0])

        ax1.set_ylabel('Dam level (%)')
        ax1.set_ylim([0, y_lim])
        ax2.set_ylabel('Number of pumps running')
        ax2.yaxis.set_major_locator(MaxNLocator(integer=True))

        ax1.set_zorder(ax2.get_zorder()+1)
        ax1.patch.set_visible(False)

        handles, labels = ax1.get_legend_handles_labels()
        handles2, labels2 = ax2.get_legend_handles_labels()
        del handles[1]
        del labels[1]
        handles += handles2
        labels += labels2
        order = [3, 1, 4, 0, 5, 2]
        ax1.legend([handles[idx] for idx in order],[labels[idx] for idx in order], bbox_to_anchor=(0., 1.02, 1., .102), loc=3, n
col=3, mode='expand', borderaxespad=0.)

        ax1.yaxis.label.set_color(sns.color_palette('muted')[0])
        ax2.yaxis.label.set_color(sns.color_palette('muted')[2])

    else:
        lbl = 'Power (resampled, step plot)' if mode == 'power' else 'Power (raw data, 1 second)'
        ax1.plot(x, y, drawstyle='steps-post', label=lbl, zorder=3)
        tou_shade(ax1, x[0])
        ax1.set_ylabel('Power (kW)')
        ax1.set_ylim([0, y_lim])

        handles, labels = ax1.get_legend_handles_labels()
        order = [1, 0, 2, 3]
        ax1.legend([handles[idx] for idx in order],[labels[idx] for idx in order], bbox_to_anchor=(0., 1.02, 1., .102), loc=3, n
col=3, mode='expand', borderaxespad=0.)

    ax1.set_xlabel('Time of day')
```

```
    if mode != 'power' and mode != 'power_raw':
        ax1.yaxis.set_ticks(np.arange(0, 101, 10))
    ax1.xaxis.set_major_locator(mdates.HourLocator())
    ax1.xaxis.set_minor_locator(mdates.MinuteLocator(byminute=30))
    ax1.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
    fig.autofmt_xdate(rotation=90)
    ax1.set_xlim((min(x), max(x)))

    ax1.grid(which='major', alpha=0.5)

    fig.tight_layout()

    if save_fig:
        if mode == 'power_raw':
            chart_type = 'power_raw'
        elif mode == 'power':
            chart_type = 'power_resampled'
        else:
            chart_type = 'dam_level_and_status'

        filename = 'output/CS{}_{}_{}_{}.png'.format(case_study, level_name, factor, chart_type)
        fig.savefig(filename, bbox_inches='tight')

    return fig
```

1-factor

In [31]:
```
df = pd.read_csv('..\..\simulations\Case_study_1\output\CS1_simulation_data_export_1-factor.csv.gz')
df['time_clean'] = pd.to_datetime(df['seconds'], unit='s')
df.head()
```

Out[31]:

	seconds	44L Level	44L Status	Eskom ToU	Pump system total power	time_clean
0	0	75.962773	2.0	3	3800.0	1970-01-01 00:00:00
1	1	75.961277	2.0	3	3800.0	1970-01-01 00:00:01
2	2	75.959780	2.0	3	3800.0	1970-01-01 00:00:02
3	3	75.958283	2.0	3	3800.0	1970-01-01 00:00:03
4	4	75.956786	2.0	3	3800.0	1970-01-01 00:00:04

In [32]:
```
for c in df.columns:
    if 'Level' in c:
        print('{} Min: {}%  Max: {}%'.format(c.replace(' Level', ''), df[c].min(), df[c].max()))
```

```
44L Min: 28.629236878109136%  Max: 83.52864126261%
```

Plot dam level and status

In [33]:
```
fig = sim_plot(1, 1, 'level', '44L', [25, 85], df['time_clean'], None, y_lim=100, save_fig=True)
```

Plot power

Raw data

In [34]:
```
fig = sim_plot(1, 1, 'power_raw', '44L', None, df['time_clean'], df['Pump system total power'], y_lim=4000, save_fig=True)
```

Resampled data

In [35]:
```
resampled = pd.DataFrame()
resampled['total_30_minute'] = df.set_index('time_clean')['Pump system total power'].resample('30T').mean()

resampled.to_csv('output/CS1_resampled_power_1_factor.csv')

fig, ax1 = plt.subplots(figsize=figure_size, dpi=1200)
#fig.patch.set_facecolor('grey')

x = resampled.index
y = resampled['total_30_minute']

ax1.plot(x,y, 'x', label='Power (data points resampled to 30 minutes)', zorder=4)
ax1.plot(x,y, '--', label='Power (resampled, line plot)', zorder=2)
ax1.plot(x,y, drawstyle='steps-post', label='Power (resampled, step plot)', zorder=3)
ax1.set_xlabel('Time of day')
ax1.set_ylabel('Power (kW)')
ax1.set_ylim([0, 4000])

ax1.axvspan('1970-01-01 00:00:00','1970-01-01 05:59:59', alpha=.25, color=colour_op, lw=0, zorder=0, label='Off-peak')
ax1.axvspan('1970-01-01 06:00:00','1970-01-01 06:59:59', alpha=.25, color=colour_s, lw=0, zorder=0, label='Standard')
ax1.axvspan('1970-01-01 07:00:00','1970-01-01 09:59:59', alpha=.25, color=colour_p, lw=0, zorder=0, label='Peak')
ax1.axvspan('1970-01-01 10:00:00','1970-01-01 17:59:59', alpha=.25, color=colour_s, lw=0, zorder=0)
ax1.axvspan('1970-01-01 18:00:00','1970-01-01 19:59:59', alpha=.25, color=colour_p, lw=0, zorder=0)
ax1.axvspan('1970-01-01 20:00:00','1970-01-01 21:59:59', alpha=.25, color=colour_s, lw=0, zorder=0)
ax1.axvspan('1970-01-01 22:00:00','1970-01-01 23:59:59', alpha=.25, color=colour_op, lw=0, zorder=0)

ax1.xaxis.set_major_locator(mdates.HourLocator())
ax1.xaxis.set_minor_locator(mdates.MinuteLocator(byminute=30))
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
fig.autofmt_xdate(rotation=90)
ax1.set_xlim((min(x),max(x)))

ax1.grid(which='major', alpha=0.5)
#ax1.grid(which='minor', alpha=0.25)
```

```
plt.legend(bbox_to_anchor=(0., 1.02, 1., .102), loc=3, ncol=2, mode='expand', borderaxespad=0.)

fig.tight_layout()
plt.show()

fig.savefig('output/CS1_1_power_resampled.png', bbox_inches='tight'),

plt.close('all')
```

2-factor

In [36]:
```
df = pd.read_csv('..\..\simulations\Case_study_1\output\CS1_simulation_data_export_2-factor.csv.gz')
df['time_clean'] = pd.to_datetime(df['seconds'], unit='s')
df.head()
```

Out[36]:

	seconds	44L Level	44L Status	Eskom ToU	Pump system total power	time_clean
0	0	75.962773	2.0	3	3800.0	1970-01-01 00:00:00
1	1	75.961277	2.0	3	3800.0	1970-01-01 00:00:01
2	2	75.959780	2.0	3	3800.0	1970-01-01 00:00:02
3	3	75.958283	2.0	3	3800.0	1970-01-01 00:00:03
4	4	75.956786	2.0	3	3800.0	1970-01-01 00:00:04

In [37]:
```
for c in df.columns:
    if 'Level' in c:
        print('{} Min: {}%  Max: {}%'.format(c.replace(' Level', ''), df[c].min(), df[c].max()))
```

```
44L Min: 28.629236878109136%  Max: 83.528641126261%
```

Plot dam level and status

In [38]:
```
fig = sim_plot(1, 2, 'level', '44L', [25, 85], df['time_clean'], None, y_lim=100, save_fig=True)
```

Plot power

In [39]:
```
resampled = pd.DataFrame()
resampled['total_30_minute'] = df.set_index('time_clean')['Pump system total power'].resample('30T').mean()
resampled.to_csv('output/CS1_resampled_power_2_factor.csv')

fig = sim_plot(1, 2, 'power', '44L', None, resampled.index, resampled['total_30_minute'], y_lim=4000, save_fig=True)
```

n-factor

In [40]:
```
df = pd.read_csv('..\..\simulations\Case_study_1\output\CS1_simulation_data_export_n-factor.csv.gz')
df['time_clean'] = pd.to_datetime(df['seconds'], unit='s')
df.head()
```

Out[40]:

	seconds	44L Level	44L Status	Eskom ToU	Pump system total power	time_clean
0	0	75.962773	2.0	3	3800.0	1970-01-01 00:00:00
1	1	75.961277	2.0	3	3800.0	1970-01-01 00:00:01
2	2	75.959780	2.0	3	3800.0	1970-01-01 00:00:02
3	3	75.958283	2.0	3	3800.0	1970-01-01 00:00:03
4	4	75.956786	2.0	3	3800.0	1970-01-01 00:00:04

In [41]:
```
for c in df.columns:
    if 'Level' in c:
        print('{} Min: {}%  Max: {}%'.format(c.replace(' Level', ''), df[c].min(), df[c].max()))
```

```
44L Min: 27.29933687812865%  Max: 80.68008112628145%
```

Plot dam level and status

In [42]:
```
fig = sim_plot(1, 'n', 'level', '44L', [25, 85], df['time_clean'], None, y_lim=100, save_fig=True)
```

Plot power

In [43]:
```
resampled = pd.DataFrame()
resampled['total_30_minute'] = df.set_index('time_clean')['Pump system total power'].resample('30T').mean()
resampled.to_csv('output/CS1_resampled_power_n_factor.csv')

fig = sim_plot(1, 'n', 'power', '44L', None, resampled.index, resampled['total_30_minute'], y_lim=4000, save_fig=True)
```

Scoring graph

In [44]:
```
scores = pd.read_csv('scoring_results.csv').set_index('Score')
scores
```

Out[44]:

	1-factor	2-factor	n-factor
Score			
Feature	16.666667	20.370370	33.333333
Performance	64.134236	64.134236	64.207095

In [45]:
```
sns.set()

score_sim = (scores['1-factor']['Performance'], scores['2-factor']['Performance'], scores['n-factor']['Performance'])
score_feat = (scores['1-factor']['Feature'], scores['2-factor']['Feature'], scores['n-factor']['Feature'])
ind = np.arange(len(score_sim))    # the x locations for the groups
width = 0.5       # the width of the bars: can also be len(x) sequence

plt.figure(figsize=figure_size)

p1 = plt.bar(ind, score_sim, width)
p2 = plt.bar(ind, score_feat, width, bottom=score_sim)


plt.ylabel('Scores')
plt.xlabel('Control system')
    #'x-factored simulation')
plt.xticks(ind, ('1-factor', '2-factor', r'$n$-factor'))
#plt.yticks(np.arange(0, 101, 10))
plt.gca().yaxis.set_major_locator(plt.NullLocator())
#plt.legend((p1[0], p2[0]), ('Performance score', 'Feature score'), loc='best')
plt.legend((p1[0], p2[0]), ('Performance score', 'Feature score'), bbox_to_anchor=(0., 1.02, 1., .102), loc=3, ncol=2, mode="exp
and", borderaxespad=0.)

for r1, r2 in zip(p1, p2):
    h1 = r1.get_height()
    h2 = r2.get_height()
    plt.text(r1.get_x() + r1.get_width() / 2., h1 / 2., "%.2f" % h1, ha="center", va="center", color='white')
    plt.text(r2.get_x() + r2.get_width() / 2., h1 + h2 / 2., "%.2f" % h2, ha="center", va="center", color='white')
    plt.text(r2.get_x() + r2.get_width() / 2., h1 + h2 + 1.5, "%.2f" % (h1 + h2),
         ha="center", va="center", color='black', weight='bold')


plt.grid(False)

plt.tight_layout()
plt.savefig('output/CS1_final_scores.png', bbox_inches='tight', figsize=figure_size, dpi=1200)
```

F.3. SIMULATION CODE

F.3.1. PUMPING SYSTEM MODULE

The pumping system module forms the core of the developed simulation engine.

pumpingsystem.py:

```
import logging
import math
import os
import sys

import pandas as pd

logging.basicConfig(stream=sys.stderr, level=logging.DEBUG)


class PumpingLevel:
    def __init__(self, name, capacity, initial_level, pump_flow, pump_power,
                 pump_schedule_table, initial_pumps_status, fissure_water_inflow,
                 hysteresis=5.0, UL_LL=95.0, UL_HL=100.0, fed_to_level=None,
                 pump_statuses_for_validation=None, n_mode_min_pumps=0, n_mode_max_pumps=3,
                 n_mode_min_level=33, n_mode_max_level=77, n_mode_control_range=5,
                 n_mode_bottom_offset=3, n_mode_top_offset=3):
        self.name = name
        self.capacity = capacity
        self.pump_flow = pump_flow
        self.pump_power = pump_power
        self.pump_schedule_table = pump_schedule_table
        self.fissure_water_inflow = fissure_water_inflow
```

```
        self.level_history = [initial_level]
        self.pump_status_history = [initial_pumps_status]
        self.fed_to_level = fed_to_level  # to which level does this one pump?
        self.last_outflow = 0
        self.hysteresis = hysteresis
        self.UL_LL = UL_LL
        self.UL_HL = UL_HL
        self.UL_100 = False
        self.max_pumps = len([1 for r in pump_schedule_table if [150, 150, 150] not in r])
        self.pump_statuses_for_validation = pump_statuses_for_validation  # this is only used
                                                                          in validation mode
        self.n_mode_min_level = n_mode_min_level
        self.n_mode_max_level = n_mode_max_level
        self.n_mode_min_pumps = n_mode_min_pumps
        self.n_mode_max_pumps = n_mode_max_pumps
        self.n_mode_control_range = n_mode_control_range
        self.n_mode_bottom_offset = n_mode_bottom_offset
        self.n_mode_top_offset = n_mode_top_offset
        # calculate starting and stopping levels for n-factor mode
        # 1 = peak, 2 = standard, 3 = off-peak
        self.n_mode_lower_bound = {3: n_mode_min_level,
                                   2: n_mode_min_level,
                                   1: n_mode_max_level - n_mode_control_range}
        self.n_mode_upper_bound = {3: n_mode_min_level + n_mode_control_range,
                                   2: n_mode_min_level + n_mode_control_range,
                                   1: n_mode_max_level}
        self.n_mode_last_change = '000'  # used for n-factor
        logging.info('{} pumping level created.'.format(self.name))
        if self.max_pumps != self.n_mode_max_pumps:
            logging.warning('{} pumping level SCADA and third party max pumps differ ({} vs
                            {})!.'.format(self.name, self.max_pumps, self.n_mode_max_pumps))

    def get_level_history(self, index=None):
        return self.level_history if index is None else self.level_history[index]

    # @levelHistory.setter
    def set_latest_level(self, value):
        self.level_history.append(value)

    def get_pump_status_history(self, index=None):
        return self.pump_status_history if index is None else self.pump_status_history[index]

    # @levelHistory.setter
    def set_latest_pump_status(self, value):
        self.pump_status_history.append(value)

    def get_scada_pump_schedule_table_level(self, pump_index, tariff_index):
        return self.pump_schedule_table[pump_index, tariff_index]

    def get_last_outflow(self):
        return 0 if self.fed_to_level is None else self.last_outflow

    def set_last_outflow(self, value):
        self.last_outflow = value

    def get_upstream_level_name(self):
        return self.fed_to_level

    def get_fissure_water_inflow(self, current_hour=None, current_minute=None, pumps=None):
        if isinstance(self.fissure_water_inflow, int) or
          isinstance(self.fissure_water_inflow, float):  # it is constant
            return self.fissure_water_inflow
        else:
            if self.fissure_water_inflow.shape[1] == 2:  # if 2 columns. Not f(pump)
                f1 = 0
                f2 = 1
                row = math.floor(current_hour)
            else:  # 3 columns. Is f(pump)
                f1 = 1
                f2 = 2
```

```
                row = pumps * 24 - 1 + math.floor(current_hour)

            if math.floor(current_minute) <= 30:
                col = f1
            else:
                col = f2

            return self.fissure_water_inflow[int(row), int(col)]

    def set_UL_100(self, bool_):
        self.UL_100 = bool_


def get_eskom_tou(current_hour):
    ch = current_hour
    if (7 <= ch < 10) or (18 <= ch < 20):  # Eskom peak
        tou_time_slot = 1
    elif (0 <= ch < 6) or (22 <= ch < 24):  # Eskom off-peak
        tou_time_slot = 3
    else:  # Eskom standard
        tou_time_slot = 2

    return tou_time_slot


def get_current_day_hour_minute(seconds):
    cd = math.floor(seconds / 86400)  # cd = current day
    ch = (seconds - cd * 86400) / (60 * 60)  # ch = current hour
    cm = (seconds - cd * 86400 - math.floor(ch) * 60 * 60) / 60  # cm = current minute

    return cd, ch, cm


class PumpSystem:
    def __init__(self, name):
        self.name = name
        self.levels = []
        self.eskom_tou = [3]
        self.total_power = []
        logging.info('{} pump system created.'.format(self.name))

    def add_level(self, pumping_level):
        self.levels.append(pumping_level)
        logging.info('{} pumping level added to {} pump system.'.format(pumping_level.name,
                                                                         self.name))

    def get_level_from_index(self, level_number):
        return self.levels[level_number]

    def get_level_from_name(self, level_name):
        for l in self.levels:
            if l.name == level_name:
                return l

    def __iter__(self):
        return iter(self.levels)

    def perform_simulation(self, mode, seconds=86400, save=False):
        # 86400 = seconds in one day
        logging.info('{} simulation started in {} mode.'.format(self.name, mode))

        if mode not in ['1-factor', '2-factor', 'n-factor', 'validation']:
            raise ValueError('Invalid simulation mode specified')

        # reset simulation if it has run before
        if len(self.total_power) > 1:
            self.reset_pumpsystem_state()

        for t in range(1, seconds):  # start at 1, because initial conditions are specified
            _, ch, cm = get_current_day_hour_minute(t)
```

```
            tou_time_slot = get_eskom_tou(ch)
            self.eskom_tou.append(tou_time_slot)

            for level in self.levels:
                # scheduling algorithm
                if mode == '1-factor' or mode == '2-factor':
                    upstream_dam_name = level.get_upstream_level_name()
                    if mode == '1-factor' or upstream_dam_name is None:
                        upper_dam_level = 45
                    else:
                        upper_dam_level = self.get_level_from_name(upstream_dam_name).
                                          get_level_history(t - 1)

                    if upper_dam_level >= level.UL_HL:
                        level.set_UL_100(True)
                    if upper_dam_level <= level.UL_LL:
                        level.set_UL_100(False)

                    if not level.UL_100:
                        pumps_required = level.get_pump_status_history(t - 1)
                        pumps_required_temp = pumps_required

                        do_next_check = False

                        for p in range(1, level.max_pumps + 1):
                            dam_level = level.get_level_history(t - 1)
                            pump_level = level.get_scada_pump_schedule_table_level(p - 1,
                                         tou_time_slot - 1)

                            if dam_level >= pump_level:
                                pumps_required_temp = p
                                do_next_check = True

                            if dam_level < (level.get_scada_pump_schedule_table_level(0,
                                            tou_time_slot - 1) - level.hysteresis):
                                pumps_required = 0
                                do_next_check = False

                        if pumps_required >= (pumps_required_temp + 2):
                            pumps_required = pumps_required_temp + 1
                        if do_next_check:
                            if pumps_required_temp > pumps_required:
                                pumps_required = pumps_required_temp
                    else:
                        pumps_required = 0

                elif mode == 'n-factor':
                    prev_level = level.get_level_history(t - 1)
                    prev_pumps = level.get_pump_status_history(t - 1)
                    pump_change = 0

                    if level.name == '31L':
                        if self.get_level_from_name('20L').get_level_history(t - 1) > 70:
                            level.n_mode_max_pumps = 1
                        if self.get_level_from_name('20L').get_level_history(t - 1) < 60:
                            level.n_mode_max_pumps = 2
                        if level.get_level_history(t - 1) >= (level.n_mode_max_level) and
                          t < 42900:
                            level.n_mode_max_pumps = 2

                    if level.name == '20L':
                        if tou_time_slot == 1:
                            if level.get_level_history(t - 1) < 75:
                                level.n_mode_max_pumps = 1
                            if level.get_level_history(t - 1) < 60:
                                level.n_mode_max_pumps = 0
                            if level.get_level_history(t - 1) > 80:
                                level.n_mode_max_pumps = 1
                        else:
```

```
                    level.n_mode_max_pumps = 2

            if level.name == 'IPC':
                if tou_time_slot == 1:
                    level.n_mode_max_pumps = self.get_level_from_name('20L').
                                                                    n_mode_max_pumps
                    if level.get_level_history(t - 1) > 90:
                        level.n_mode_max_pumps = 1
                else:
                    if self.get_level_from_name('Surface').get_level_history(t - 1)
                      < 90 and t < 39600:
                        level.n_mode_max_pumps = 3
                    if level.get_level_history(t - 1) > 80 and t > 39600 and
                      t < 64800:
                        level.n_mode_max_pumps = 3
                    if self.get_level_from_name('Surface').get_level_history(t - 1)
                       < 90 and t > 57600:
                        level.n_mode_max_pumps = 3
                    if self.get_level_from_name('Surface').get_level_history(t - 1)
                       >= 95 and t < 39600:
                        level.n_mode_max_pumps = 2
                    if self.get_level_from_name('Surface').get_level_history(t - 1)
                       >= 97.5 and level.get_level_history(t - 1) < 60:
                        level.n_mode_max_pumps = 1
                    if level.get_level_history(t - 1) < 50 and
                      self.get_level_from_name('Surface').
                                        get_level_history(t - 1) >= 90 and t > 39600:
                        level.n_mode_max_pumps = 1
                    if t > 70200:
                        level.n_mode_max_pumps = 2
                    if t > 77400:
                        level.n_mode_max_pumps = 3
                    if t > 81000:
                        level.n_mode_max_pumps = 2

            max_pumps = level.n_mode_max_pumps

            for p in range(0, max_pumps):
                # check if pumps should be switched on
                check_lev = (level.n_mode_upper_bound[tou_time_slot] + p *
                             level.n_mode_top_offset)
                if prev_level >= check_lev:
                    this_change = check_lev
                    if this_change != level.n_mode_last_change:
                        pump_change = 1
                        level.n_mode_last_change = this_change
                        break
                # check if pumps should be switched off
                check_lev2 = (level.n_mode_lower_bound[tou_time_slot] - p *
                              level.n_mode_bottom_offset)
                if prev_level <= check_lev2:
                    this_change = check_lev2
                    if (level.n_mode_last_change == '000') or (this_change <
                       level.n_mode_last_change) or (tou_time_slot !=
                       self.eskom_tou[-2]):
                        pump_change = -1
                        level.n_mode_last_change = this_change
                        break

            pumps_required = prev_pumps + pump_change
            if pumps_required < level.n_mode_min_pumps:
                pumps_required = level.n_mode_min_pumps
            elif pumps_required > max_pumps:
                pumps_required = max_pumps

        else:  # validation mode, so use actual statuses
            pumps_required = level.pump_statuses_for_validation[t]

        # calculate and update simulation values
        pumps = pumps_required
```

```
                outflow = pumps * level.pump_flow

                level.set_last_outflow(outflow)

                additional_in_flow = 0
                for level2 in self.levels:
                    if level2.fed_to_level == level.name:
                        additional_in_flow += level2.get_last_outflow()

                level_new = level.get_level_history(t - 1) + 100 / level.capacity *
                            (level.get_fissure_water_inflow(ch, cm, pumps) +
                            additional_in_flow - outflow)
                level.set_latest_level(level_new)
                level.set_latest_pump_status(pumps)

        # calculate pump system total power
        # can do it in the loop above, though
        power_list = []
        for level in self.levels:
            power_list.append(pd.DataFrame(level.get_pump_status_history()) *
                              level.pump_power)
        self.total_power = pd.concat(power_list, axis=1).sum(axis=1).values

        logging.info('{} simulation completed in {} mode.'.format(self.name, mode))

        if save:
            self._save_simulation_results(mode, seconds)

    def _save_simulation_results(self, mode, seconds):
        df_list = []
        index = range(0, seconds)
        for level in self.levels:
            data_level = level.get_level_history()
            data_schedule = level.get_pump_status_history()
            data = {level.name + " Level": data_level,
                    level.name + " Status": data_schedule}
            df_list.append(pd.DataFrame(data=data, index=index))
        df = pd.concat(df_list, axis=1)

        data = {'Pump system total power': self.total_power,
                'Eskom ToU': self.eskom_tou}
        df = pd.concat([df, pd.DataFrame(data=data, index=index)], axis=1)
        df.index.name = 'seconds'

        os.makedirs(r'output/', exist_ok=True)
        df.to_csv('output/{}_simulation_data_export_{}.csv.gz'.format(self.name, mode),
         compression='gzip')
        logging.info('{} simulation data saved.'.format(mode))

    def reset_pumpsystem_state(self):
        self.eskom_tou = [3]
        self.total_power = []

        for level in self.levels:
            level.level_history = [level.level_history[0]]
            level.pump_status_history = [level.pump_status_history[0]]
            level.last_outflow = 0

        logging.info('{} pumping system successfully cleared.'.format(self.name))
```

The `pumping system` module is used to perform the verification simulations and simulations for the case studies as in the following sections.

F.3.2. Model verification simulation 1: dam with constant inflow

simulation_M1.py:

```
import modules.pumpingsystem as ps
import numpy as np

pump_system = ps.PumpSystem('M1')
pump_system.add_level(ps.PumpingLevel('0', 1000000, 0, 0, 0, np.zeros((1, 3)), 0, 10))

pump_system.perform_simulation('1-factor', save=True)
```

F.3.3. Model verification simulation 2: dam with variable inflow

simulation_M2.py:

```
import modules.pumpingsystem as ps
import numpy as np
import pandas as pd

# Inflow into dam
inflow = pd.read_csv('M2_dam_inflow_profiles.csv.gz')
inflow = np.reshape(inflow['0 Inflow'].values, (24, 2))

pump_system = ps.PumpSystem('M2')
pump_system.add_level(ps.PumpingLevel('0', 1000000, 10, 0, 0, np.zeros((1, 3)), 0, inflow))

pump_system.perform_simulation('1-factor', save=True)
```

F.3.4. Model verification simulation 3: dam with inflow and a running pump

simulation_M3.py:

```
import numpy as np

import modules.pumpingsystem as ps

pump_system = ps.PumpSystem('M3')
pump_system.add_level(ps.PumpingLevel('0', 1000000, 0, 40, 0, np.zeros((1, 3)), 1, 50))

pump_system.perform_simulation('1-factor', save=True)
```

F.3.5. Decision-making verification simulation D1: mine SCADA algorithm

simulation_D1.py:

```
import modules.pumpingsystem as ps
import numpy as np

schedule = np.array([[70, 70, 70],
                     [75, 75, 75]])

pump_system = ps.PumpSystem('D1')
pump_system.add_level(ps.PumpingLevel('0', 1000000, 0, 15, 0, schedule, 0, 20))

pump_system.perform_simulation('1-factor', save=True)
```

F.3.6. Decision-making verification simulation D2: REMS algorithm

simulation_D2.py:

```
import modules.pumpingsystem as ps
import numpy as np

pump_system = ps.PumpSystem('D2')
pump_system.add_level(ps.PumpingLevel('0', 1000000, 0, 15, 0, np.zeros((1, 3)), 0, 18,
                                      n_mode_max_pumps=2, n_mode_min_level=15,
                                      n_mode_control_range=5, n_mode_bottom_offset=5,
                                      n_mode_top_offset=2.5))

pump_system.perform_simulation('n-factor', save=True)
```

F.3.7. Simulation: Case study 1

simulation_CS1.py:

```
import modules.pumpingsystem as ps
import pandas as pd
import numpy as np

# Pump schedule as per SCADA. rows = pumps, columns 1:=Peak, 2:=Standard, 3:Off-peak
pump_schedule_44 = np.array([[80, 50, 30],
                             [85, 60, 40],
                             [150, 150, 150],
                             [150, 150, 150]])

# Inflows into dams
dam_inflow_profiles = pd.read_csv('input/CS1_dam_inflow_profiles.csv.gz')
inflow_44 = np.reshape(dam_inflow_profiles['44L Inflow'].values, (24, 2))

# Read actual data for initial conditions and validation
actual_values = pd.read_csv('input/CS1_data_for_validation.csv.gz')
actual_status_44 = actual_values['44L Status'].values
initial_level_44 = actual_values['44L Level'][0]

# Create pump system
pump_system = ps.PumpSystem('CS1')
pump_system.add_level(ps.PumpingLevel("44L", 5000000, initial_level_44,
                                      143, 1900, pump_schedule_44, actual_status_44[0],
                                      inflow_44,
                                      pump_statuses_for_validation=actual_status_44,
                                      n_mode_max_pumps=2, n_mode_min_level=30,
                                      n_mode_max_level=80))


# Perform simulations
pump_system.perform_simulation(mode='validation', save=True)
pump_system.perform_simulation(mode='1-factor', save=True)
pump_system.perform_simulation(mode='2-factor', save=True)
pump_system.perform_simulation(mode='n-factor', save=True)
```

F.3.8. Simulation: Case study 2

simulation_CS2.py:

```
import modules.pumpingsystem as ps
import pandas as pd
import numpy as np

# Pump schedule as per SCADA. rows = pumps, columns 1:=Peak, 2:=Standard, 3:Off-peak
pump_schedule_27 = np.array([[80, 30, 30],
                             [85, 40, 40],
```

```
                            [90, 85, 80],
                            [150, 150, 150],
                            [150, 150, 150]])
pump_schedule_12 = np.array([[80, 30, 30],
                            [85, 40, 40],
                            [90, 50, 50],
                            [150, 150, 150],
                            [150, 150, 150]])

# Inflows into dams
dam_inflow_profiles = pd.read_csv('input/CS2_dam_inflow_profiles.csv.gz')
inflow_27 = np.reshape(dam_inflow_profiles['27L Inflow'].values, (24, 2))
inflow_12 = np.reshape(dam_inflow_profiles['12L Inflow'].values, (24, 2))

# Read actual data for initial conditions and validation
actual_values = pd.read_csv('input/CS2_data_for_validation.csv.gz')
actual_status_27 = actual_values['27L Status'].values
actual_status_12 = actual_values['12L Status'].values
initial_level_27 = actual_values['27L Level'][0]
initial_level_12 = actual_values['12L Level'][0]

# Create pump system
pump_system = ps.PumpSystem('CS2')
pump_system.add_level(ps.PumpingLevel("27L", 3000000, initial_level_27,
                                      236.1, 2925.6, pump_schedule_27, actual_status_27[0],
                                      inflow_27, fed_to_level="12L",
                                      pump_statuses_for_validation=actual_status_27,
                                      n_mode_max_pumps=2, n_mode_control_range=20))
pump_system.add_level(ps.PumpingLevel("12L", 3000000, initial_level_12,
                                      194.6, 2656.6, pump_schedule_12, actual_status_12[0],
                                      inflow_12,
                                      pump_statuses_for_validation=actual_status_12,
                                      n_mode_max_pumps=3, n_mode_control_range=20,
                                      n_mode_min_level=36, n_mode_max_level=80))


# Perform simulations
pump_system.perform_simulation(mode='validation', save=True)
pump_system.perform_simulation(mode='1-factor', save=True)
pump_system.perform_simulation(mode='2-factor', save=True)
pump_system.perform_simulation(mode='n-factor', save=True)
```

F.3.9. SIMULATION: CASE STUDY 3

simulation_CS3.py:

```
import modules.pumpingsystem as ps
import pandas as pd
import numpy as np

# Pump schedule as per SCADA. rows = pumps, columns 1:=Peak, 2:=Standard, 3:Off-peak
pump_schedule_41 = np.array([[72, 42, 50],
                            [95, 78, 86],
                            [110, 110, 110],
                            [120, 120, 120],
                            [150, 150, 150]])
pump_schedule_31 = np.array([[77, 45, 45],
                            [92, 70, 60],
                            [110, 110, 110],
                            [120, 120, 120]])
pump_schedule_20 = np.array([[72, 47, 55],
                            [82, 70, 70],
                            [91, 87, 92],
                            [110, 110, 110]])
pump_schedule_IPC = np.array([[80, 45, 45],
                             [85, 70, 60],
                             [90, 82, 82],
                             [110, 110, 110],
                             [150, 150, 150]])
```

```
dummy_pump_schedule_surface = np.array([[150, 150, 150]])

# Inflows into dams
dam_inflow_profiles = pd.read_csv('input/CS3_dam_inflow_profiles.csv.gz')
inflow_41 = np.reshape(dam_inflow_profiles['41L Inflow'].values, (24, 2))
inflow_31 = np.reshape(dam_inflow_profiles['31L Inflow'].values, (24, 2))
inflow_20 = np.reshape(dam_inflow_profiles['20L Inflow'].values, (24, 2))
inflow_IPC = np.reshape(dam_inflow_profiles['IPC Inflow'].values, (24, 2))
inflow_surface = np.reshape(dam_inflow_profiles['Surface Inflow'].values, (24, 2))

# Read actual data for initial conditions and validation
actual_values = pd.read_csv('input/CS3_data_for_validation.csv.gz')
actual_status_41 = actual_values['41L Status'].values
actual_status_31 = actual_values['31L Status'].values
actual_status_20 = actual_values['20L Status'].values
actual_status_IPC = actual_values['IPC Status'].values
initial_level_41 = actual_values['41L Level'][0]
initial_level_31 = actual_values['31L Level'][0]
initial_level_20 = actual_values['20L Level'][0]
initial_level_IPC = actual_values['IPC Level'][0]
initial_level_surface = actual_values['Surface Level'][0]

# Create pump system
pump_system = ps.PumpSystem('CS3')
pump_system.add_level(ps.PumpingLevel("41L", 3000000, initial_level_41,
                                      216.8, 3508.4, pump_schedule_41, actual_status_41[0],
                                      inflow_41, fed_to_level="31L",
                                      pump_statuses_for_validation=actual_status_41,
                                      n_mode_max_pumps=2, n_mode_max_level=80,
                                      n_mode_control_range=30, n_mode_top_offset=5))
pump_system.add_level(ps.PumpingLevel("31L", 3000000, initial_level_31,
                                      146.8, 3283.6, pump_schedule_31, actual_status_31[0],
                                      inflow_31, fed_to_level="20L",
                                      pump_statuses_for_validation=actual_status_31,
                                      n_mode_max_pumps=2, n_mode_max_level=80,
                                      n_mode_control_range=20, n_mode_top_offset=5,
                                      n_mode_bottom_offset=5))
pump_system.add_level(ps.PumpingLevel("20L", 3000000, initial_level_20,
                                      171.8, 3821.0, pump_schedule_20, actual_status_20[0],
                                      inflow_20, fed_to_level="IPC",
                                      pump_statuses_for_validation=actual_status_20,
                                      n_mode_max_pumps=2, n_mode_control_range=20,
                                      n_mode_top_offset=7, n_mode_bottom_offset=5))
pump_system.add_level(ps.PumpingLevel("IPC", 3000000, initial_level_IPC,
                                      147.4, 3572.8, pump_schedule_IPC, actual_status_IPC[0],
                                      inflow_IPC, fed_to_level="Surface",
                                      pump_statuses_for_validation=actual_status_IPC,
                                      n_mode_max_pumps=2, n_mode_max_level=80,
                                      n_mode_control_range=10, n_mode_top_offset=5,
                                      n_mode_bottom_offset=3))
pump_system.add_level(ps.PumpingLevel("Surface", 5000000, initial_level_surface,
                                      0, 0, dummy_pump_schedule_surface, 0, inflow_surface,
                                      pump_statuses_for_validation=actual_status_IPC,
                                      n_mode_max_pumps=0))  # the status data doesn't matter


# Perform simulations
pump_system.perform_simulation(mode='validation', save=True)
pump_system.perform_simulation(mode='1-factor', save=True)
pump_system.perform_simulation(mode='2-factor', save=True)
pump_system.perform_simulation(mode='n-factor', save=True)
```