



**university of
 groningen**

Development of a Graphical User Interface using Digital Twins in regards to Energy Storage

Bachelor Integration Project

BSc Industrial Engineering and Management
Faculty of Science and Engineering
University of Groningen

June 27, 2023

Author:

Zoraiz Ali Syed
S4226615

Supervisors:

First Supervisor: A. (Antonis) Vakis, Prof Dr
Second Supervisor: M. (Mehran) Mohebbi, MSc
Daily Supervisor: A. T. (Andreas) Asiikkis, MSc

0.1 Abstract

The use of Digital Twins is an up and coming technique to aid in the optimisation of products and processes. Digital Twins are a virtual representation of a physical entity that mimic the state of the physical in a virtual environment. Through the use of Digital Twins, parameters and factors affecting the functionality of a physical entity can be altered to observe their effects and achieve certain desired states. To aid with the visualisation of a Digital Twin, a Graphical User Interface (GUI) is used. A GUI can be said to be a visual representation of the Digital Twin. It is the connection between the user and the Twin, whereby the user can interact with the Digital Twin in an interactive and responsive manner to visualise the desired outputs.

This project focuses on the creation of a GUI for the Ocean Battery, a novel energy storage device. The created GUI allows the user to change certain parameters affiliated with the Ocean Battery and observe how changes in parameters relate to the efficiency of the system. It is aimed through the use of this GUI to allow for further developments in academia such as research related to sustainable energy storage as well as assisting future generations in learning about these concepts while practicing their programming skills.

Contents

- 0.1 Abstract 1

- 1 Introduction 4**
 - 1.1 Ocean Battery Background 4
 - 1.2 GUI Application 5

- 2 Research Design 7**
 - 2.1 Research Objective 7
 - 2.1.1 Research Questions 7
 - 2.2 Stakeholder Analysis 7
 - 2.3 System Analysis 8
 - 2.4 Methods & Tools 11

- 3 Digital Twin Characteristics 12**
 - 3.1 Overview 12
 - 3.2 Functionality 12
 - 3.3 Limitations 13

- 4 GUI Characteristics 14**
 - 4.1 Overview 14
 - 4.2 User Friendliness 14
 - 4.3 Responsiveness 15
 - 4.4 Robustness 15
 - 4.5 Limitations 15

- 5 Existing Material 17**
 - 5.1 Existing GUI 17
 - 5.2 Digital Twin Script 18

- 6 GUI Development 19**
 - 6.1 Updated Digital Twin 19
 - 6.2 Structure GUI-OB 20
 - 6.3 Process 21

- 7 Discussion 26**
 - 7.1 Charging Phase 26
 - 7.2 Modifying Equations 26
 - 7.3 User Limitation 26
 - 7.4 Academic Use 27

- 8 Conclusion 28**

- A Reference Material 31**

- B Manual 35**

Acronyms

OB - Ocean Battery

OG - Ocean Grazer

GUI - Graphical User Interface

DT - Digital Twin

1: Introduction

DTs have been used over time as a means of replicating physical entities for purposes of modelling and simulation in order to analyse said entities [1]. Through the use of DTs, physical entities are represented in a virtual manner such that the environment they are represented in can be altered to achieve a desired outcome. This project relates to the DT of the OB. The OB is a method of storing energy in kinetic form and converting this to electricity when required [2]. Currently, the application of the OB is limited to the company involved in its development. It was aimed through this project to make this technology more accessible to the general public, including academia, so as to allow for further insights that could aid with the development and application.

A way in which this technology can be made accessible is by the creation of a GUI. The creation of a GUI will allow the user to visualise what is happening in the DT. Through the GUI, the user will be able to see the relevant parameters and how they affect the processes of the entity. The user will be able to alter these parameters to achieve a desired state. GUIs can be created in multiple different ways, for example in Matlab or through a webpage. It was the aim of this project to determine and create the most optimal form of GUI that fulfils certain requirements presented by the user. In order to create the optimal GUI, the necessary parameters must be included in the DT. This project also aimed to determine what essential factors must be part of a DT to create an optimal GUI.

1.1 Ocean Battery Background

The OB is a component deployed in seawater that is a project of the OG company. In its essence, OB is an energy storage platform that aims for the long term storage and production of energy in a renewable manner. There are two main states by which the OB achieves this: charging and discharging. In the charging state, electrical energy produced through renewable sources pumps working fluid from a rigid reservoir into a flexible reservoir. This is seen as a conversion of electrical energy into potential energy. The flexible reservoir has the ability to store the fluid for long periods of time due to the acting pressures from the seawater. For discharging, the fluid stored in the reservoir travels through a turbine that converts the stored potential energy into electrical energy. The fluid travels from the flexible reservoir back into the rigid reservoir and this is enabled through pressure releases. Through these cycles of charging and discharging, the OB is able to convert renewable sourced electrical energy into potential energy, until the point this potential energy is required to be converted into electrical energy again. As mentioned in the European Union's energy storage report, storage of energy is essential to adjust the supply and demand of electricity as well as control price fluctuations [3]. A sustainable energy storage mechanism such as the OB is therefore essential for energy security and efficiency to be guaranteed for when it is most required.

Equally essential is the fact that due to rapid technological developments, changes to the OB can be made. Upgrades or improvements that can make the OB efficient have to be taken into account. It would be a cumbersome and expensive process to make a new OB every year to test and adopt changes. This is where a DT and a GUI prove to be useful. By having a virtual twin of the OB, changes can be made to observe how they effect the functionality of the energy storage platform and whether they are indeed worth adopting into the physical entity. These changes

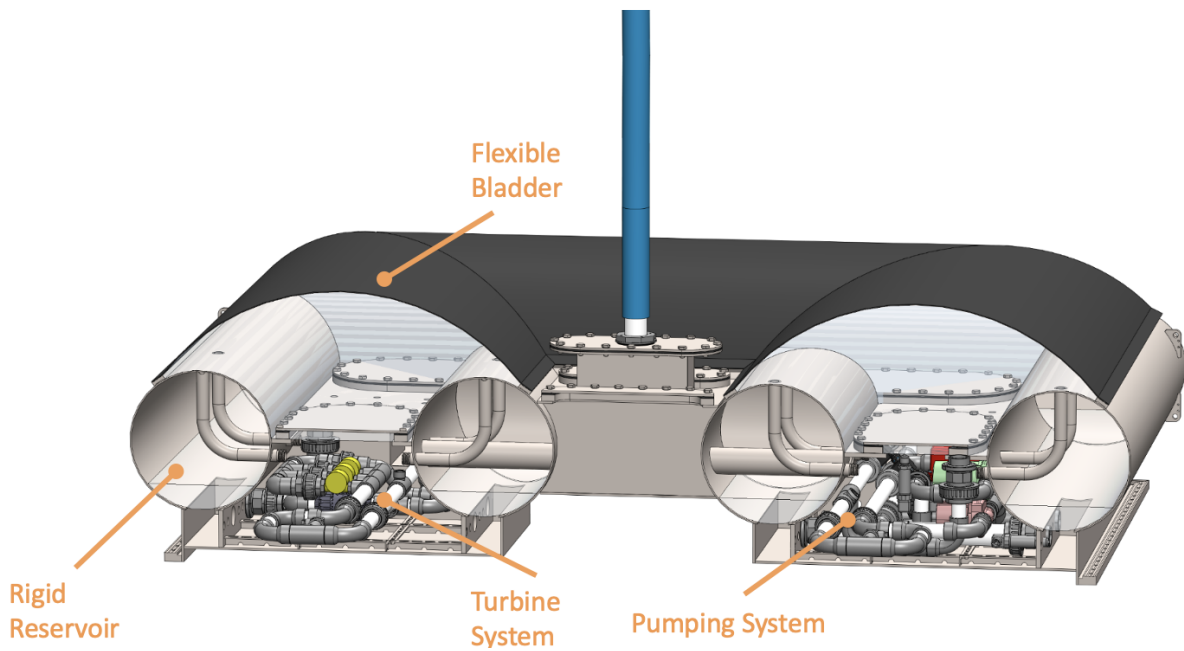


Figure 1.1: A representation of the Ocean Battery. The main systems such as the turbine, reservoir, pump and bladder can be seen. The pump is utilised in the charging phase whereas the turbine is utilised in the discharging phase. (OB schematics and artistic renderings, provided by courtesy of OG B.V.)

can be observed through the use of a GUI which can ensure that users from all backgrounds, and not only a technical background, are able to provide their input and observe the appropriate outcomes.

1.2 GUI Application

David Redmond-Pyle and Alan Moore present in their book a practical process for the creation of GUIs [4]. They discuss this from the perspective of a software developer but their design is based on the requirements of the user. A specific GUI was created to model the spreading of epidemics [5]. The creation of this GUI kept in mind the purpose, which was to create a computational tool for academics as well as for policy analysis. The design of the GUI allowed the users to alter parameters and view simulations without having to do large scale technical computations. Similarly, a GUI was created to model the Autoclave Curing Process in the Aerospace industry in order to enable operators to work together with artificial intelligence and understand the process better through the simulations [6]. This GUI was created using DTs of the process and aimed for operators to intuitively simulate and understand the production process. A GUI was also created using DTs to monitor a welding manufacturing process [7]. This GUI aimed to be intuitive and effective for the user while showcasing a visual representation of the manufacturing process.

The works showcased in the previous paragraph highlight the importance of creating a GUI that fits the purpose of its physical entity and follows the requirements of the user. It can be observed that some common user requirements relate to interacting with the GUI in an intuitive way, not requiring large computations and having an overall efficient yet effective simulation. It can also be noted that for the production of the GUI, the data required from the DTs mainly relates to the

involved parameters and the overall process. Not all the parameters have to be implemented in the GUI thus it is essential to identify the most valuable factors that can be used.

It is also good to note here that GUIs can be created in multiple different environments. Matlab is able to produce GUIs based on the given functions and code for a process [8]. Python has various frameworks that can be utilised for the creation of GUIs [9]. Languages such as Java and HTML can also be used to create GUIs [10] whereas software such as RaspberryPi also allows for the creation of user interfaces [11]. However, the current research into this topic has resulted in functions and code being provided on Matlab. In this case, it would be most efficient yet not optimal to create the GUI in Matlab. This is within the scope of this research and will be discussed.

2: Research Design

2.1 Research Objective

The objective of this research is to design a Graphical User Interface for the Ocean Battery based on information present in its Digital Twin. The GUI will be created in order for the general public to be able to access it for research, academics and further developments. To create the GUI, the necessary parameters have to be obtained through the DT and the DT in turn must accurately depict the state of the physical entities. This research will also look into considering what essential components are to be included in a DT and how an accurate representation of the physical entity can be ensured. It is aimed to create this GUs within a time-span of 3 months during which the progress of the GUI will be monitored by regular testing to ensure it complies with the requirements. A why-what analysis is shown in figure 2.1 and depicts what the problem is, why it needs to be solved and what is preventing it from being solved at the moment.

2.1.1 Research Questions

In order to satisfy the above research objective, the main research question to be asked is as follows: What are the constituents of a Graphical User Interface that allow it to perform its function in the most effective manner while using the data present in the Digital Twin?

Certain sub-questions have to be answered to determine an appropriate response to the main research question. These are as follows:

- What is the basic function of a DT?
- How does a DT accurately develop the state of the physical entity?
- What are the user requirements for the GUI?
- What is the most optimal configuration for the GUI?
- How does the existing DT and GUI compare to the desired DT and GUI?

2.2 Stakeholder Analysis

The problem owner in this system is the Ocean Grazer company since it is their Ocean Battery which is being visualised and any effects on the battery or due to the battery, will affect the company. Additional stakeholders with their level of interest and power can be seen in figure 2.2 and a small description follows:

- The Management of OG: This project affects the company as it is their technology which is utilised and any improvements or analysis on the technology will affect them directly. This is why the management is a key player with high interest and high power. Once the GUI is created, the management will have the final say on whether or not the GUI fits their purpose thus they have the power to guide the direction of this project.
- Researchers and Academics: Researchers and Academics of the general public, particularly

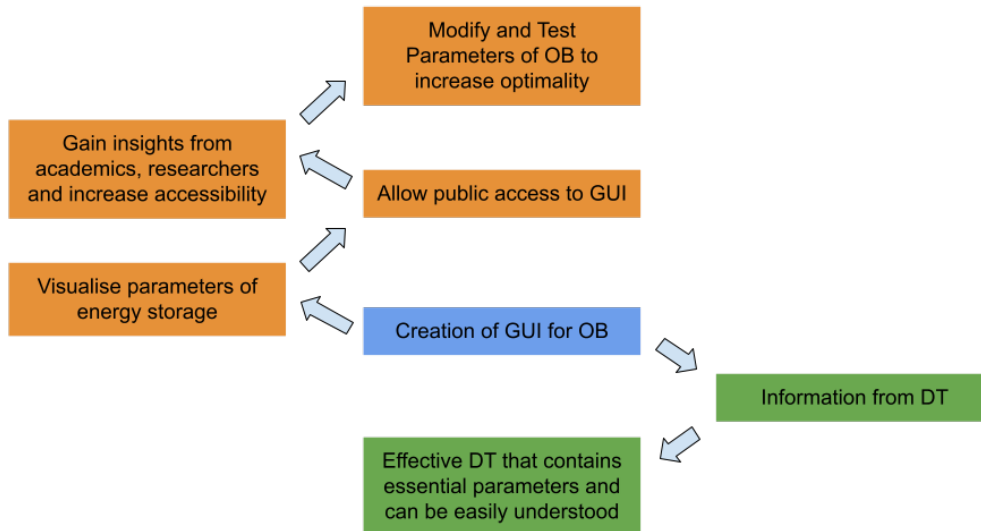


Figure 2.1: Why-What analysis of the described problem. The problem is depicted in the blue box. The boxes in orange showcase why this problem needs to be solved while the boxes in green showcase what is stopping this problem from being solved.

those interested in the field of energy storage are the subjects of this project. They have high interest as they are the users for whom the GUI is intended. The users can be those from the field of science and engineering as energy storage is a topic that is most likely to peak their interest. However, it is important to consider that the GUI will be made available to the general public and people who are simply interested in energy storage might also be interested to use it. This user can be further narrowed down to those that have the appropriate software to run the GUI, for example if the GUI is developed in Matlab then the user would need Matlab to run it.

- RUG: The university is a stakeholder with low power and interest since the developed deliverable and this project comes under the property of the university. The university will be the one to publish this project once it is completed thus the project has to meet certain standards set by the university.

2.3 System Analysis

The system in this case revolves around the creation of a visual interface for the analysis of energy storing platforms such as the Ocean Battery, as seen in figure 2.3. The goal of the system is to create a GUI for this platform. Through the GUI, optimal scenarios for the physical entities can be visualised and the parameters modified to reach these scenarios. In order to create the GUI, the necessary parameters have to be obtained through the DT. The DT in turn obtains the parameters from the physical entities. It must be noted here that it is important that the state of the DT must be similar to the state of the physical entity otherwise the DT will not contain accurate information and any visual representation will be inaccurate, resulting in optimal scenarios not being created.

The criteria for the GUI will be determined through literature by analysing the general idea

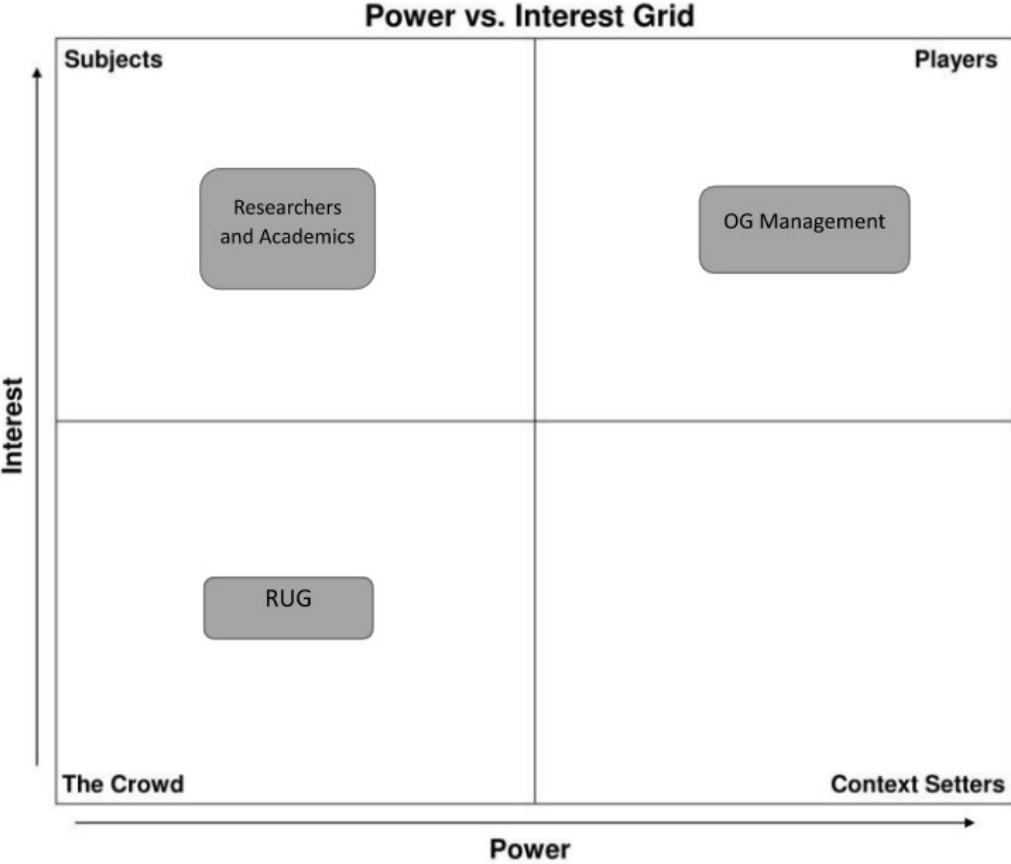


Figure 2.2: Stakeholder analysis depicting the various stakeholders according to their level of power and interest. The axes show increasing levels of power and interest.

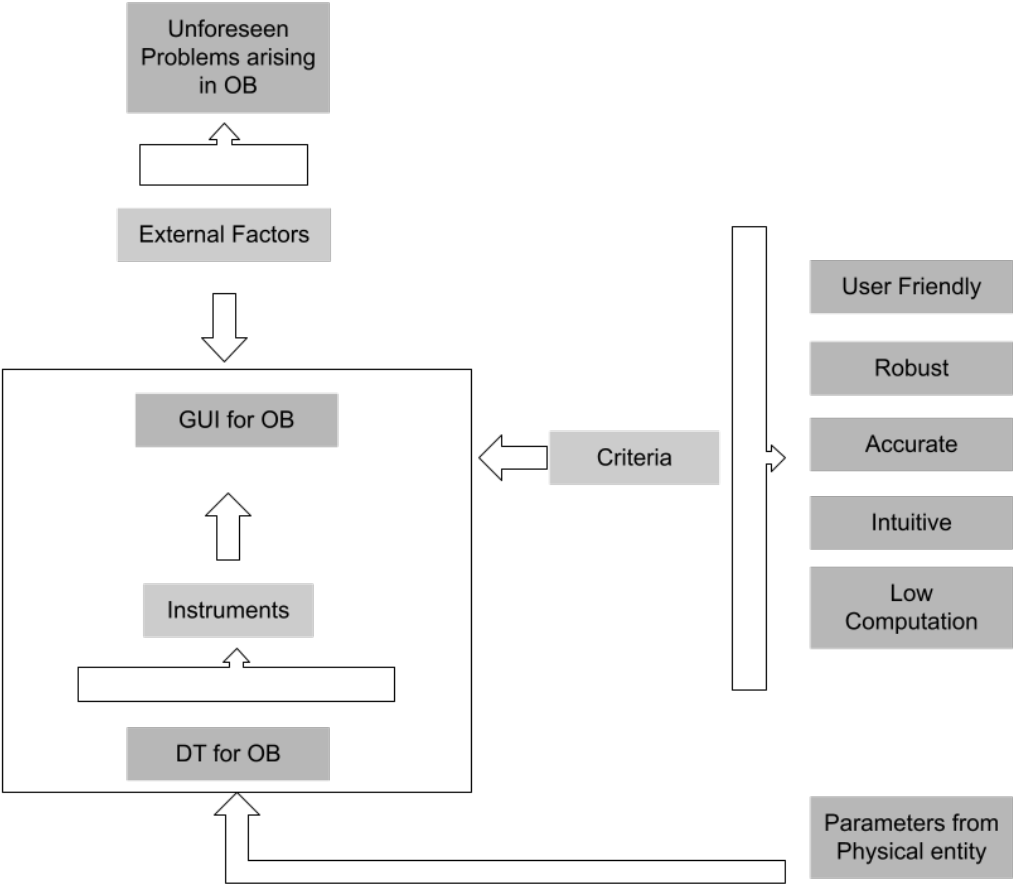


Figure 2.3: Overall view of the System related to the creation of GUIs using DTs. The scope of the system is limited to the GUIs and DTs.

for when GUIs were created for different purposes. A quick scan revealed that GUIs have to be user friendly, robust and of low computation. Specific criteria related to the user can be determined through surveys (discussed in section 2.4). The major external factor that could affect the creation of GUIs is any uncertainties or unforeseen situations that effect the physical entity and are not depicted in the digital twin. These unforeseen events can cause inaccuracies in the GUI leading to the desired situation not being created [12].

2.4 Methods & Tools

In order to carry out this research and satisfy the questions presented in 2.1.1, various strategies will be used. These include the use of case studies, grounded theory and desk research. The research is aimed to be more qualitative and of depth. The theory to be researched includes making comparisons of case studies and desk research to determine which situations can be applied or learnt from to apply to our situation. These research strategies can be used to answer the questions related to function, criteria, configuration and environment. They can also be used to determine user requirements however, user requirements can be more accurately determined through surveys as this will give practical information directly from the target user. The research will also utilise methods of trial and error to determine what the most optimal configurations are for the GUI to operate with.

By using case studies and desk research, it will be analysed what the general idea has been in the creation of GUIs. The general idea refers to the general concepts that developers have kept in mind when creating the GUIs. A quick analysis has revealed user friendliness and low computation to be some core concepts to keep in mind. Through further research, it is aimed to come up with more general concepts that will be integrated in the creation of the GUIs.

For the actual development of the GUI, MATLAB will be used as a tool, specifically MATLAB App Designer. This is an application within MATLAB that allows for creation of a user interface using simplistic "drag and drop" techniques. The developer is able to create a workspace and include certain components in it. These components are in built in App Designer. The components can be assigned functions in such a way that when the user interacts with them, the function is executed resulting in the desired effect. The added components will depend on the functionality of the GUI whereas the functions depend on the DT.

3: Digital Twin Characteristics

3.1 Overview

In their study, Jones et al [13] have done a thorough literature review to determine the characteristics that comprise a DT. They came up with 13 of these characteristics. Thus a DT can be summarised as follows: there exists a physical entity within a physical environment. The physical environment is composed of the parameters that affect the physical entity. These parameters are measured using tools and then transferred into a virtual (digital) environment. The virtual environment is created using data from the physical environment. The virtual environment houses the virtual entity which can be considered to be a model of the physical entity. The state of the virtual entity and the virtual environment must be the same as the physical entity and the physical environment i.e. the parameters, values and conditions must mimic each other. Within the virtual environment, the virtual entity can be tested and parameters can be altered to achieve a desired state. These conditions must then be transferred from the virtual environment to the physical environment to observe their real life effects. Thus a DT involves a cycle of data flowing from the physical environment to the virtual environment where this data is altered to reach a desired situation and once this target is achieved, the data flows back to the physical state. A physical entity is able to perform physical processes and these processes must be represented as virtual processes in the virtual environment.

3.2 Functionality

A question arises regarding the functionality of a DT and why there is a need for it. In their article related to the industrial applications of DTs, Jiang et al discuss the workflow of manufacturing [14]. They highlight the gaps within the industrial process where DTs can prove to be useful. The gaps exist in optimisation and responsiveness aspects. During the design of a manufacturing process, the optimal design is not determined due to certain manufacturing practices not being taken into account. There is also the concept of improving by learning where further adjustments can be made based on previously observed outcomes. Furthermore, fluctuations and uncertainties in the demand are not always taken into account in the design which can be hindrances at later stages. Having DTs of industrial processes allows for predictive controls to ensure an optimal design. Historical data can be evaluated to allow for new designs to capitalise on learning outcomes from previous designs. Having a DT also allows for improvements and their future effects to be monitored virtually before adopting changes on the physical entity itself. Jiang et al describe a DT as a "constantly learning living model". They emphasise on the DT being able to communicate with its physical entity through a digital thread. This threading technique allows for synchronisation of data, providing the virtual entity with real time updates it can utilise for processing. For synchronisation to occur, there has to be a complete representation of the physical entity and its processes by the virtual entity.

In their research highlighting the characteristics of DTs, Baricelli et al provide useful insights into what a DT should comprise of in order to accurately depict its physical entity [15]. They mention communication as a vital tool. The digital twin and its physical counterpart must ensure seamless data transfer through networking devices. The communication must be up to standard to ensure dynamic data flow which will allow for real time processing within the virtual environment. The

data the DT processes should not only be data obtained through the physical entity but also historical data, if it exists. Historical data may further comprise of the expertise of humans that are familiar in the field and have contributed to the subject. The DT must then be able to analyse the data and accurately adapt to the physical entity, throughout the life-cycle of the physical entity. By adapting throughout the life-cycle, the DT is able to predict outcomes that otherwise could not be known. This would allow the users to make the necessary judgements related to the physical entity which they can act upon to ensure its efficient processing.

The DT must also have the capability to model and simulate the behaviour of the entity it represents. This can be done through a user interface and will be further elaborated upon in section 4.

3.3 Limitations

Baricelli et al described certain limitations related to DTs [15]. Most notably, DTs have to ensure seamless, real-time data transfer between the physical and virtual entities. Factors such as internet connection, data size and data transfer rate might hinder the DTs ability to be responsive. Privacy concerns also have to be taken into account when developing DTs to ensure that data is secure and not vulnerable. The company or person in charge of creating the DT must not exploit the data and respect its use. Another limitation is how accessible the DT is. The user interface utilised by the DT must be convenient and the design of the DT must take the end user into consideration.

In their article, Deloitte mention the challenge of adopting the DT over an organisational ecosystem to realise their full potential [16]. It may be the case that a process is affected by various different components, each of which is affected by another component. These components may be in different locations and subject to different factors. For a DT of the process to be effective, DTs of the various components would have to be created which will need to interact with each other and the overall process DT. Due to the emergence of Industry 4.0, this challenge can slowly yet surely be overcome. Using the Internet of Things (IoT), data collection and data transfer is more efficient than ever and the technology is progressing rapidly. It should therefore not be an issue for DTs to collaborate with each other if the supplied data is easily attainable.

Regarding modelling of DTs, Robert Saracco discusses the issue of DTs having no set structure and architecture [17]. He says that there is no standard of creating models which can result in various different models not being compatible with each other to have a cohesive DT. This could add onto the costs of developing a DT as research would need to be done to ensure that all models are created in a compatible manner. Additional cost related issues relate to data collection, software creation and data transfer [15]. Cost issues might limit the development of DTs only to those who can afford it.

4: GUI Characteristics

4.1 Overview

Donald Norman's theory of action regards the steps taken to reach a desired goal [18]. He relates the psychological intentions a person has to the physical aspects of conducting a task [4]. In the context of this project, the relation between the psychological and the physical is bridged through the construction of a GUI. Gartner defines a GUI as an interface that allows the user to interact with a system [19]. Instead of typing in commands and lines of code, the user is able to simply perform a task by the use of icons, menus and a mouse which increases the users productivity and enables the task to be done effectively. In its essence, a GUI is a visual representation of information that the user can interact with. Most GUIs comprise of certain basic concepts. They represent information in a visually appealing manner, they utilise multiple windows for the user to engage with, they require minimum input from the user thus are mostly based on point and click interactions, they engage with the user by providing menus, toolbars and showing dialogue boxes, and they allow for customisation by the user.

To create a successful GUI, there has to be a bridge between the psychological intention and the physical task as described by Norman. He suggested the idea of having 'gulfs', specifically, the gulf of execution and the gulf of evaluation. The gulf of execution looks into how the psychological desires of a person can be translated into actions while the gulf of evaluation looks into how the actions are perceived by the user and what meaning they hold. Once this bridge has been created, the activities to be conducted by the user can be identified and portrayed as a mental model. A mental model is defined as the understanding the user has of a system that leads the user to perform various activities in order to reach the various goals [4]. In the context of this project, the mental model consists of the users understanding of the OB and its behaviour. Through this mental model, the user can determine what goals they need to fulfill and what actions must be taken to fulfill them. By understanding what the mental model of the user is, the GUI can be developed in such a way that the user is aided in carrying out their tasks in a manner that is efficient and effective.

Further research into GUI characteristics revealed three main components that must comprise a GUI: it should be user friendly, responsive and robust. These will be further discussed.

4.2 User Friendliness

By ensuring that a GUI is user friendly, the developer is allowing for the user to be able to carry out their task in the most effective way possible, with minimum errors and maximum satisfaction. In their experiment using an interface related to that of an ATM, Tractinsky et al determined how efficiency increases with aesthetics [20]. Having an easy to navigate interface allows for users to minimise errors and complete tasks faster resulting in higher efficiencies. A simple layout based on intuitive design and clear guidance further makes it easier for users to interact with the interface. This was seen in the study of Nielsen and Molich, which determined that it was easier for users to learn how to operate a GUI if the design was simple [21]. Having a user friendly GUI further results in users having an overall positive experience when engaging with the GUI. Hassenzahl et al highlighted the satisfaction users obtain by using user-friendly interfaces [22].

It can thus be summarised here that by having a user friendly GUI design, the user will be able to perform their task more efficiently with few errors and a great deal of satisfaction.

4.3 Responsiveness

Equally important is for a GUI to be responsive, one of the main reasons being competition. It is quite straightforward for anyone with a bit of technical knowledge to create a GUI therefore the competitive advantage lies in how responsive the design is. A report by leading content delivery network services company Akamai showcases how having a slow response rate led to reduced customer engagement and eventually loss of customers [23]. Responsive feedback enhances the usability of a GUI as the user is able to coherently understand the cause and effect relationships of the system [24]. Coherent understanding plays on the emotional state of the user when it comes to human-technology interactions as having a proper understanding of how a system is working leads the user to process information in a clear manner which increases user satisfaction in terms of usability. Efficiency also increases with responsiveness as faster response rates lead to the user obtaining information and subsequently reacting to this information quicker, causing a cycle of efficiency to follow.

4.4 Robustness

Robustness ensures the longevity and sustainability of the object, in this case the OB GUI. Robustness was taken into consideration to account for the fact that future updates and enhancements would be made to the GUI thus it should be designed in a way so as not to "break" or lose its functionality when these adjustments are made. A robust GUI further ensures system stability by integrating error handling mechanisms that are able to handle unforeseen errors. This could be done in the form of error messages that specify exactly what is going wrong when changes are made. Moreover, GUIs should be compatible with multiple softwares and hardwares to ensure that they can be accessible to a wide user base and the user does not encounter any hindrances when using it.

4.5 Limitations

It was mentioned in section 4.1 that by understanding the mental model of the user, the design of the GUI can be ascertained. This is where the major limitation lies. A mental model is influenced by multiple factors, including existing mental models. This was highlighted by Johnson-Laird in his argument of how deductive reasoning is a result of mental manipulation that infers state of affairs [25]. What this means in non-philosophical terms is how we understand the things around us that lead us to make logical decisions. People have a different understanding of things around them, even though it may be the same thing. This difference is due to the various differences that are amongst people such as culture, upbringing, education, exposure etc. It is because of these differences that the mental models of each individual differs. Therefore, when creating the GUI, it is highly improbable that all mental models can be taken into account to create a '1 GUI fits all' interface. However, the way in which this can be countered is by ensuring that users have a similar general mental model about the system [4]. In the case of the OB, if the users have a similar understanding of what the DT of the OB is expected to do, they will have a similar mental model of what actions need to be undertaken to get the desired results. While

constructing the GUI, it is therefore essential to fully understand the DT itself to ensure that the mental model of the developer aligns with the expected mental model of the user.

5: Existing Material

5.1 Existing GUI

There is an existing GUI that was created by Lars Hof as part of an earlier Integration Project. Images of this GUI can be seen in Appendix A. This GUI is composed of 3 tabs: an overview tab, a charging tab and a discharging tab. Each tab has an upper section that relates to it in a specific way. The overview tab has an upper section that shows how the overall OB looks while also including tables that calculate the K values of various components. These K values represent the factor of Head Loss by various components and are affected by the number of components. This tab contains a general description of the OB with a table showcasing the main parameters. A 'start simulation' button is present in this tab which when pressed, runs a code in the backend that calculates various variables and presents this output in the form of graphs. The graphs shown in the overview tab are the same as the graphs shown in the charging and discharging graphs. The current view of the graphs in the overview tab depends on which of those two tabs was last open. All tabs also contain certain elements that show the energy saved and produced as well as the efficiency of the system. These elements are calculated when the simulation button is clicked and the backend code is run.

The charging and discharging tab have an upper section that shows the structure of the components involved in each phase. This also contains a drop down menu that shows the components of the OB. When a component is selected, an image of the component is shown on the top left with its description below and a table showcasing the parameters associated with that component. The graphs on each tab showcase the various heads and flow associated with either of the phases.

When the start simulation button is clicked in this GUI, it takes approximately 2 minutes for the code to run and graphs to be produced. The graphs that are already shown in the GUI tabs also begin to appear as individual figures. Furthermore, the parameter values given in the table of the GUI can be altered and the button clicked again to give new results corresponding to the new values. For the code to run with the new values, it takes approximately 5 minutes, which is quite a wait.

The existing GUI has an overall clear structure which showcases all the parameters and components involved. It also provides proper descriptions of the components and provides graphs which are relevant in their output. However, there are some excess components present in the GUI. The graphs in the overview tab for example give the same overview as can be seen when going to the charging or discharging phase. The drop down menus in each phase show the same components, not those associated with the specific phase. Moving between tabs has to be done by selecting the tab from the top ribbon. To view the various components and the parameters that affect them, the components have to be manually selected from the drop down menu. Although these are not major issues, they can be improved upon to make the GUI more convenient for the user, which is what the new GUI aims to do.

5.2 Digital Twin Script

A currently unpublished paper, written by Nienhuis et al [26], describes the behaviour of the OB. There are 2 main phases of the OB: Charging and Discharging. In the charging phase, electrical energy is converted into potential energy by pumping fluid which is stored in a rigid reservoir into a flexible reservoir. When in the flexible reservoir, pressure differences from within the reservoir and those from outside i.e. from the surrounding seawater, allow for the fluid to be stored as long term energy storage. In the discharging phase, a ball valve is opened that results in further pressure differences that allow the fluid to pass over a turbine and into the rigid reservoir again, generating electricity as a result. A Matlab script, also made by Nienhuis, describes the discharging phase of the OB while having a focus on the efficiency of the entire charging-discharging cycle. This script is the base upon which the GUI will be created.

The script begins by initialising certain variables. These include defining the time horizons, efficiency values and certain constants related to temperature and water. Initialisation further includes defining the geometry of the reservoir and ducting system as well as assigning the location of pressure sensors. Following this, the script defines the number and type of piping used in the various ducting sections, such as 90 degree bends and T-bends, as well as their K values. After initial values have been assigned, the script creates arrays for the variables that will be calculated. This is done to store the resulting data.

Following initialisation, the script begins the loop that iterates over the defined time. This is where the main calculations take place. The loop first calculates the flow velocities in various sections, which are used to find the Head Losses. The Head Losses are calculated for each ducting section using functions that have been created by Nienhuis. These functions utilise the geometry of the specific ducting system, the flow velocities, the number and type of piping used and certain coefficients related to friction as their input. The Head Losses for all 3 sections are calculated and summed up to find the total Head Loss.

Apart from Head Losses within the ducting system, dynamic Head Losses are also calculated. These are determined using the geometry of the rigid reservoir as well as the volume and flow which change with every iteration. The Head Losses in the ducting system, along with the location of pressure sensors, are used to find the Head Loss in the turbine, rigid reservoir and bladder.

Once all calculations are done, the Head Losses are corrected for head i.e. the values take into account the reference point of pressure sensors. The theory behind this can be found in detail in the paper of Nienhuis et al [26]. Following this, the efficiency of a cycle of charging and discharging is calculated using the efficiencies of the pump and turbine as well as the Head Losses in the ducting sections. The results are then plotted to observe the outputs.

This analytical script has been validated by taking experimental data from the OB and comparing it. The plotted outputs from the analytical code match closely with the experimental data which confirms that the script performs accurate calculations and can be considered a DT of the OB since it accurately mimics its performance.

6: GUI Development

6.1 Updated Digital Twin

The DT for this GUI is the code developed by Nienhuis. This code existed as a Matlab script that calculated variables over the various sections of the OB. One of the goals of making this GUI was to make it robust in such a way that future work does not affect its functionality. To aid in this, the Matlab script was sub divided into 4 scripts. This was done due to the following: once the GUI is created, analysis and comparison of various parameters can be done. Apart from this, further developments regarding the OB may arise as well such as more efficient calculation methods or dynamically changing parameters. These developments would have to be integrated with the GUI in order to allow the GUI to take these into account for future analyses and simulations. If the existing script that includes all the sections is included, it would be a hassle and an inefficient method for future developers to make changes to. They would have to go through thousands of lines of code and understand how every section works in order to determine where to make the changes. Instead, by dividing the full script into multiple scripts, future developers can simply make changes to the scripts that are specific to their purpose without having to worry about understanding the entire code. Of course, understanding the entire system is essential to make analyses regarding the OB however, developers would not need to understand the minute, technical details that are part of the code. Every script contains a description in the beginning explaining what it is about and what variables are calculated.

The first script consists of the main loop that iterates over time to calculate Head Losses in the various sections and the flow velocities. Initially, this script was also divided into smaller scripts that calculated the Head Loss in each section and the flow velocities separately. However, at the end of the loop, the value for the flow is updated which is used in the following iterations. It became quite messy to find a way to isolate and link the flow in each script therefore it was decided to keep one main loop script that calculates both parameters.

The second script calculates the Head Losses in the turbine, bladder and reservoir. Following this, the next script calculates the values for pressure taking into account the reference points of the various sensors. These reference points have been determined in the paper of Nienhuis et al [26]. The last script that is called relates to the efficiencies and calculates the efficiency of the OB of one cycle. A cycle includes both the charging and discharging phase.

As the GUI is run, all these functions are called which run for the assigned parameters and provide the relevant outputs. When the GUI first starts up, it takes about 10 seconds to load. This is a substantial improvement from the previous GUI which took about 2 minutes to load. For simulation, at the moment of writing, the GUI takes about 40 seconds to display results from the moment the run button is pushed. The previous GUI took about 5 minutes. Creating a responsive GUI was one of the objectives and this decrease in running time shows the increased responsiveness of the new GUI. It was determined that the time step used in the GUI is the limiting factor that increases the running time. The GUI iterates 70,000 times due to the specified time step. These iterations had to be done for various functions in the GUI. The function that took the most amount of time was determining the water level in the rigid reservoir based on the volume. By discussing with Nienhuis, a solution was found to alter the function that calculated this, resulting in a reduction of the run time. The run time can further be reduced by the addition

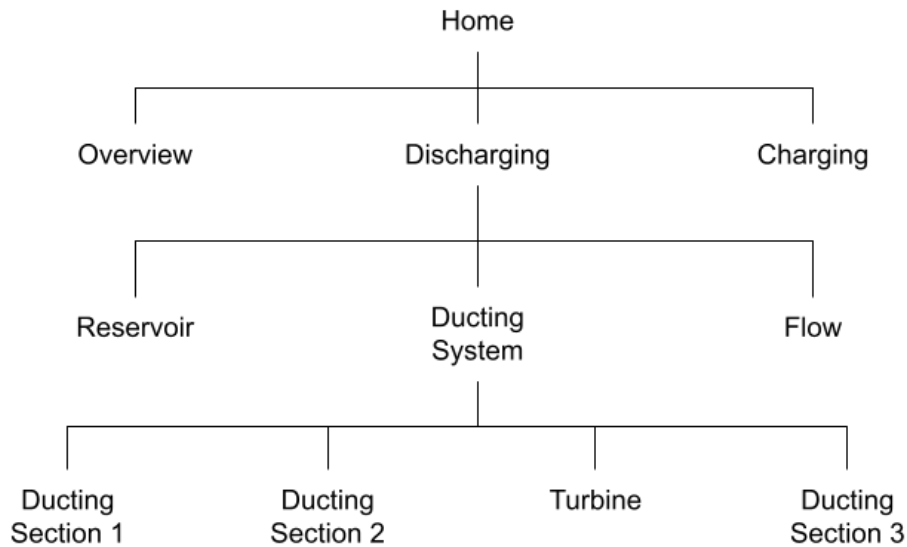


Figure 6.1: Tree diagram showcasing the layout of the GUI in terms of tabs. The lines represent links between the tabs. Additionally, every tab within the discharging section links back to discharging while every tab within the ducting system links back to ducting and discharging.

of a lookup table which will remove the need of the GUI iterating 70,000 times to find the height for the corresponding volume. This can be done by running the DT once to find the various values and assigning them to a table. A function can then be created that will use the values for the depth for the associated volume. This development is not yet implemented in the GUI but is currently in process.

The development of the lookup table is a good example of the robustness of the GUI. A step wise approach can ascertain which script the development has to take place in. The main chunk of work is done in the main loop script and it is in this script that the variable for depth can be found. The depth is a further function of the P9 VRIGID function, which is one of the existing functions part of the DT. It is in this function that the table has to be created. Once changes have been made in this function, no further changes are required in any other sections and the GUI can retain its functionality.

6.2 Structure GUI-OB

Figure 6.1 shows a tree diagram that demonstrates how the GUI will be structured.

There will be an initial 'Home' page. On this page there will be 3 images, each linking to a different tab which are the overview tab, the charging tab and the discharging tab. The overview tab will contain a graph showcasing the heads in various components along with a table of general parameters. By clicking on the image that opens the charging/discharging tab, the user will see an overall image of the charging/discharging system. The various components of the system can be seen in this image and by clicking on that component, the user will be taken to the specific tab. In that tab, the user will be able to see the parameters involved in that component as well as an output graph that displays the results. The created tabs relate to the flow, turbine,

ducting system and reservoir in the discharging phase. The tables of parameters in all tabs will be editable, just as they were in the previous GUI. This will allow the user to choose values for the parameters and run the GUI to see what the output would be.

Navigation between the tabs is done through images that act as buttons. Using images visualises the section the user will navigate to and adds aesthetic appeal to the GUI.

6.3 Process

To accurately determine what features the user would prefer to see in the GUI, regular meetings were conducted with the supervisors who are the main target audience. Valuable insight was obtained related to the design of the GUI, the components to be included and the parameters from the DT. User suggestions revolved around having a clear structure that is intuitive to follow, having images that depict a visual representation of the components, making comparisons between simulations easily observable and making the simulations run at a swift pace.

When beginning development of the GUI, the objective was to spread out the information instead of having it concentrated in a limited space as was the case with the previous GUI. Spreading out information allows the user to process and understand content more thoroughly, as users are limited in their working memory and being exposed to a great deal of information all at once limits their capacity to grasp the information [27]. Additionally, the concept of grouping similar objects together results in increased comprehension by the user while improving aesthetic appeal [28]. This concept was utilised in the creation of tabs such that each tab contains specific information related to a particular section of the OB. Figure 6.2 showcases the "Home" page of the GUI with the various tabs visible at the top. While operating the GUI, user satisfaction is further increased if the user is visually attracted to the interface such that they consider it to be inviting and easy to work with [20]. This is aided with the creation of tabs as having a layout of balanced information, specific to its purpose, allows the user to understand better the task they are performing and comprehend the required information easily.

Once the tabs were created, it had to be determined what features they should include. The basic layout consisted of having a table of parameters, relevant to a particular section, and a graph that depicted the output. However, building onto the previous GUI, it was decided to include a description table that provided information about the tab. This included a brief explanation of what the tab was about and the contents of that section. The tab for the second ducting section can be seen in 6.3. The layout for this tab is the standard layout for all tabs and can be described as follows:

Each tab has a heading related to a specific section of the OB. Below the heading is a brief description of the section and any associated parameters. The parameters are then depicted in a table as well as graphs that depict the output. Most tabs have one graph while some have two, depending on the outputs as ascertained in the DT script. A visualisation of the parameters is also provided. As the DT script calculates the Head Losses in the various ducting sections, the parameters in these sections are the number and type of ducts used. These include a variety of bends and certain connection elements. The visualisation of these parameters is done by providing an image of the ducts for instance in figure 6.3, the second ducting section comprises of three type of bends; 90 degree bends, S bends and 90 degree elbow bends. Images of these bends can be seen in the tab. These images are a representation of the actual ducts used in the

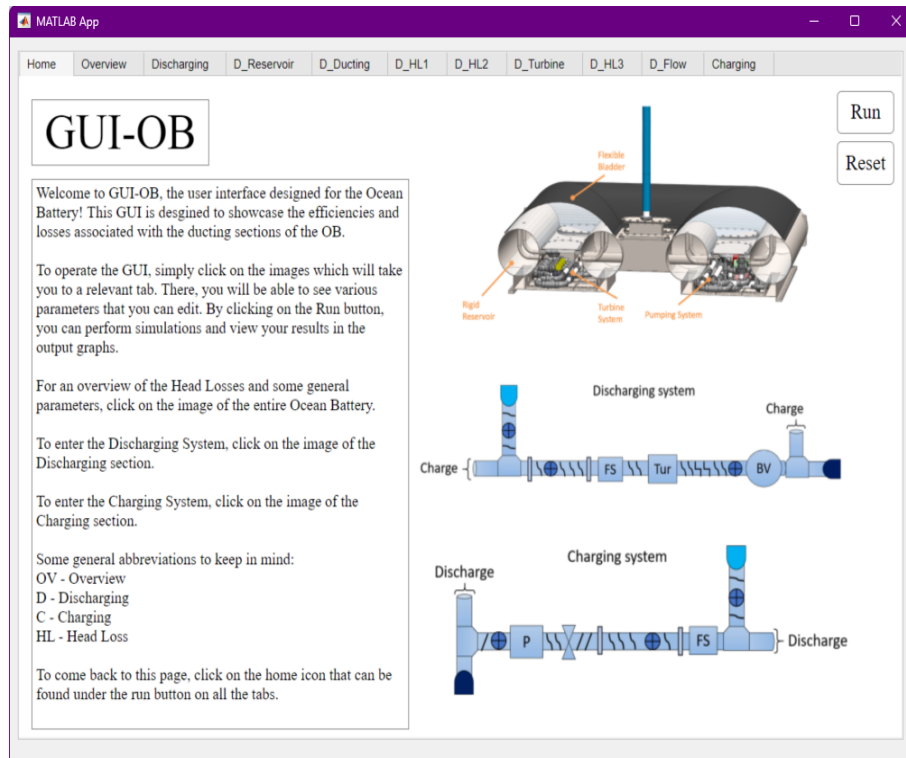


Figure 6.2: The "Home" page of the GUI. This is the page the user sees when the GUI starts up. On the top, the various tabs are visible. Navigation between the tabs is done through the use of clickable images.

OB as they were obtained using the CAD files used in their development. In the top right of the tab, certain buttons and icons can be seen. These can be classified as the navigation group as by clicking them, the user is able to navigate to the various sections. A home icon is seen which takes the user to the main "Home" page. Underneath this is an image of the discharging system which will take the user to the main Discharging page. Next to these are buttons depicting "Run" and "Reset". As the names suggest, clicking on the run button will begin the simulation while clicking on rest will reset the values of all parameters to their initial conditions. During development, it was slightly tricky to determine when the simulation began and when it ended. This was solved by the use of adding alert messages that signalled the beginning and end of the simulation as shown in figure 6.4. In the existing GUI, an alert message was generated when the simulation began but not when it ended. Although the end of the simulation can be observed by the depiction of new graphs, some simulations result in similar new outputs being generated which cannot be distinguished from existing outputs. Therefore an alert for the end of the simulation proves to be useful. Below the simulation buttons is an image of the ducting system, which takes the user to the main Ducting page. The concept was to allow the user to easily navigate between the main sections and the sub sections, as depicted in figure 6.1. Thus, for every tab within Discharging, the user is able to navigate back to Discharging with the click of an image, and for every tab within Ducting, the user is able to navigate back to Ducting as well as Discharging with the click of an image. This allows for easy navigation between sections and improves user experience by being intuitive and effective [29] [30]. To further aid with navigation, each section of the Ducting system contains images that when clicked, take the user to the other sections. Thus the user can make changes in one section and see whether that has any effect on the other sections.

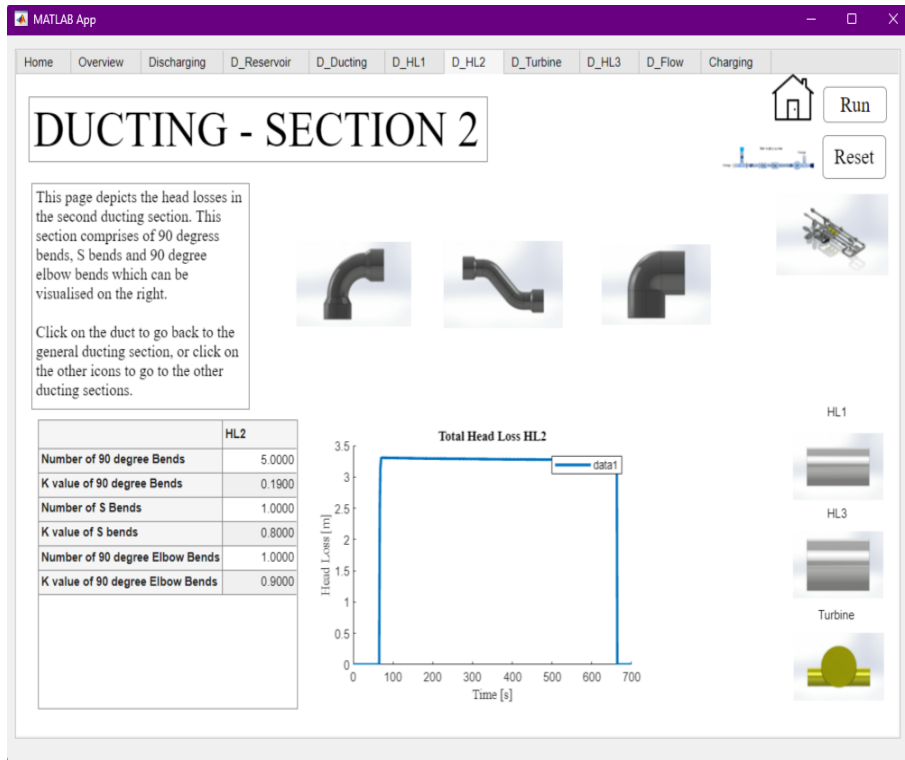


Figure 6.3: The second ducting section tab of the GUI. This tab includes a brief description of the components of this section, as well as a visualisation of these components. Links to other tabs and buttons to simulate are also visible.

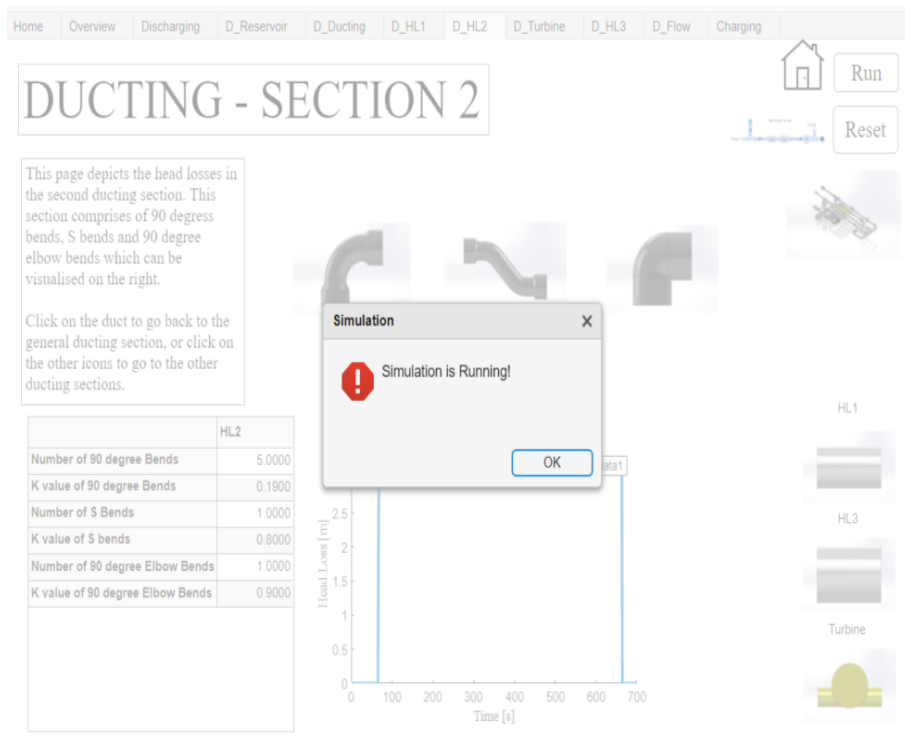


Figure 6.4: An alert message signalling that the run button has been pressed and the simulation is running.

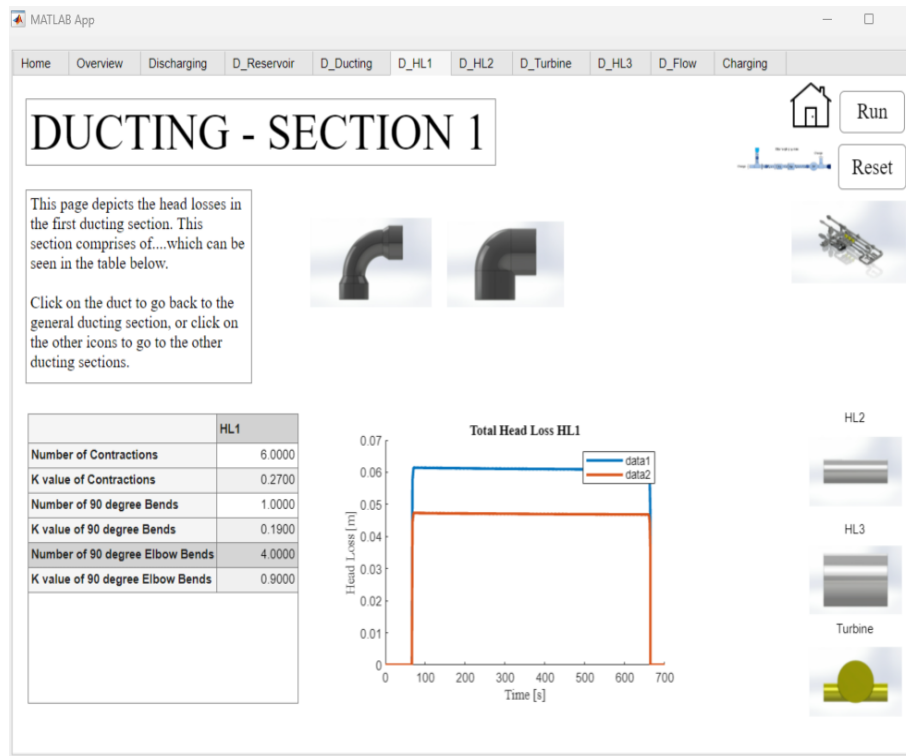


Figure 6.5: Comparison between 2 runs of the GUI. The results of changing parameters in the first ducting section can be clearly visualised on the same plot.

An element missing in the previous GUI was comparison between simulation runs. If the parameters were changed and a simulation was run, the output would only show the new results and previous results would disappear. This was not very useful for analysis. In the creation of this GUI, efforts were made to store both the old and new data, and they were fruitful. Figure 6.5 shows the results of two runs where parameters were changed in the first ducting section. The two data sets are clearly depicted in the graph and an analysis can be conducted on their comparison. In his book, Alberto Cairo highlights the importance of having visual comparison of data as this leads to recognition of patterns, trends and insights that would otherwise not be ascertained clearly [31].

Table 6.1 showcases a comparison done of the run time between the new GUI and the existing GUI. The start time is the time taken for the application to open whereas the simulation time is the time it takes for the backend code to run. It can be seen that the start time for the existing GUI is faster than the new GUI. This may be due to the fact that the existing GUI utilises only 2 MATLAB functions whereas the new GUI utilises 4 functions, which take slightly longer to run. However, the simulation time for the new GUI is much faster than the existing GUI. This was done by collaborating with Nienhuis to find solutions to make the DT code run faster. The limiting factor in the DT was the timestep which causes the DT to iterate over the code 70,000 times. Initially, it was planned to reduce the timestep however this caused further errors in the code as the timestep was linked to certain built-in MATLAB functions which did not run properly if the timestep changed. It was then decided to change the DT in a way that these built-in functions are not used. This was done for part of the DT, however, for another section this is still a work in process. It is due to this that there is a predicted simulation time mentioned in the table which is the estimated run time when this update is made.

Table 6.1: Table comparing the run times of the new GUI and existing GUI. Time was determined while the personal computer was both running on battery power and while charging.

	New GUI	Existing GUI
Start Time (battery power)	00:13	00:05
Simulation Time (battery power)	01:58	approx. 05:00
Start Time (charging)	00:03	00:03
Simulation Time (charging)	00:53	approx. 03:00
Predicted Simulation Time	approx. 00:15	-

A manual has been created that goes further into detail regarding the programming of the GUI (A). This manual can be consulted to understand how to properly run the GUI and what each section details.

7: Discussion

7.1 Charging Phase

In section 5, it was mentioned that the script which is the basis of the DT focuses mainly on the discharging phase. This is due to information not being available regarding the specifications of the ducting system related to the charging phase. This phase utilises a pump, which is the main point of difference as compared to the discharging that utilises a turbine. For the Matlab script, an assumption was made that allows for the behaviour of the pump to be similar to the behaviour of the turbine. This assumption was used for the calculation of the efficiencies. Specific information related to the pump and the various ducting sessions was not available at the time of writing this report. Due to this, the Head Losses experienced in this phase could not be modelled. However, for future work, once these specifications are known, they can be updated in the DT by being added to the Matlab script and consequently becoming a part of the GUI. The manner of doing this is detailed in the GUI manual.

7.2 Modifying Equations

Further work also needs to be done to determine whether the equations used in the current script produce the best possible results. It might be the case that the introduction of new technology or further research results in better equations calculating the Head Losses and efficiencies. For this purpose, the robust design of the GUI would ensure that these changes can be made without the GUI losing its functionality. It must thus be taken into account that the equations currently governing the behaviour of the OB might not be the most accurate ones.

7.3 User Limitation

There are two main ways by which the GUI can be used by the user, either with the aid of MATLAB or without. The GUI is packaged in a file and further details on running it can be found in the manual in B. For users who have MATLAB, running the GUI is quite straightforward as long as they have all the input files in the same directory. They simply need to open the application and it will run. For users who do not have MATLAB, the package file contains an adapted version of MATLAB called MATLAB Runtime which users first need to install. This will aid them in operating the GUI.

Additionally, it might be the case that some users (or even developers) prefer not to use MATLAB and wish to run the GUI in a different environment. For this, the DT files must be adapted and converted to suit the specifications of the environment. The syntax of MATLAB and that of, for example, Python is quite different thus conversion must keep that into account. As for the GUI itself, it has to be determined whether it is possible to convert the script made in App Designer into a script that can be utilised by a different environment. If this is not the case, then the structure, layout and functionality of the GUI must be recreated with the specific chosen environment.

7.4 Academic Use

The manner in which the GUI is created is done in a step by step, logical approach, taking into account the characteristics and the manner of thinking as presented in section 4. The usability of the GUI allows it to be used in academia for both research and teaching purposes. Those interested in the behaviour of energy storage platforms can alter the parameters of the GUI to model and compare the various different states. However, the GUI can also be used as a means of educating those interested in GUI development, Matlab applications as well as educating regarding the dynamics of the OB. The OB utilises many concepts related to Fluid Dynamics and Dynamics of Engineering Systems. It can provide a visual representation to students regarding the affect the various parameters have on the overall system. Furthermore, following the step by step approach as determined in the manual, students can use the GUI to practice their Matlab and programming skills. The usability of the GUI is dus not limited only to research but also encompasses education.

8: Conclusion

DTs, as virtual representations of physical entities, provide a means to simulate and understand the behavior of real-world systems in a virtual environment. By mimicking the state of physical entities, DTs enable researchers and engineers to explore different scenarios, test various parameters and observe the effects on system performance. To interact with and visualize the DTs, GUIs act as a bridge between users and the virtual representations. GUIs provide an intuitive and user-friendly interface that allows users to modify parameters, input data and interact with the DT in a responsive and interactive manner. Through the GUI, users can observe the virtual system's behavior and gain insights into the impact of different parameter settings or interventions. For the OB GUI to operate effectively and not be hindered by future developments, the DT was divided into various functions to ensure that future developers do not experience difficulty in interpreting and modifying the twin.

In the context of the OB project, the GUI facilitates the exploration and optimization of the OB system. The GUI was created to enhance user experience and usability by being responsive and robust, the main criteria determined through various case studies. Users can modify specific parameters presented in tables within the various tabs of the GUI and observe the resulting outputs shown in the graphs. By making changes through the GUI, users can observe how these modifications affect the efficiency and performance of the system. This capability supports research and development in sustainable energy storage and provides a platform for academic exploration.

Furthermore, the GUI can serve as an educational tool. It allows future generations to learn about sustainable energy concepts while simultaneously practicing their programming skills. The interactive and visual nature of the GUI enhances the learning experience, enabling users to experiment, visualize outcomes and understand the complex relationships within the system. This not only fosters a deeper understanding of sustainable energy storage but also promotes engagement and interest in related fields.

By looking back at the research questions presented in 2.1.1, it can be seen that the main question was answered by determining the characteristics of a GUI that make it most effective in carrying out its task. The function of a DT has also been discussed as being a virtual representation of a physical entity, which is accurately developed by ensuring that the states of the two entities mimic each other through efficient information transfer. The user requirements of the GUI were determined through regular meetings with supervisors, who are part of the intended audience. The optimal configuration of the GUI was also developed by using inspiration from the existing GUI and literature that detailed effective GUI design. The comparison between the existing GUI and the new GUI was done based on the run times, with the new GUI having an improved run time.

The combination of DTs and GUIs provides a powerful framework for optimization, research and education. The ability to simulate and visualize systems in a virtual environment empowers researchers and engineers to optimize product and process design. Simultaneously, GUIs facilitate user interaction, data visualization and experimentation, enabling users to gain insights and make informed decisions. The integration of DTs and GUIs holds significant potential for advancements in various fields, driving innovation, sustainability and educational outcomes.

Bibliography

- [1] A. El Saddik, “Digital twins: The convergence of multimedia technologies,” *IEEE Multi-Media*, vol. 25, no. 2, pp. 87–92, 2018. DOI: 10.1109/MMUL.2018.023121167.
- [2] *Ocean grazer*, Jan. 2023. [Online]. Available: <https://oceangrazer.com/>.
- [3] E. Union, *Energy storage*, 2023. [Online]. Available: https://energy.ec.europa.eu/topics/research-and-technology/energy-storage_en#:~:text=Besides%5C%20being%5C%20an%5C%20important%5C%20flexibility,to%5C%20prices%5C%20and%5C%20their%5C%20needs..
- [4] A. Moore and D. Redmond-Pyle, “Graphical user interface design and evaluation: A practical process,” 1995.
- [5] W. V. d. Broeck, C. Gioannini, B. Gonçalves, M. Quaggiotto, V. Colizza, and A. Vespignani, “The gleamviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale,” *BMC infectious diseases*, vol. 11, no. 1, pp. 1–14, 2011.
- [6] J.-S. Jwo, H.-Y. Hsieh, C.-H. Lee, *et al.*, “Simulation and modeling of a data twin service for the autoclave curing process,” *IEEE Access*, vol. 10, pp. 111 879–111 887, 2022.
- [7] Q. Wang, W. Jiao, and Y. Zhang, “Deep learning-empowered digital twin for visualized weld joint growth monitoring and penetration control,” *Journal of Manufacturing Systems*, vol. 57, pp. 429–439, 2020, ISSN: 0278-6125. DOI: <https://doi.org/10.1016/j.jmsy.2020.10.002>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0278612520301710>.
- [8] *Matlab gui*. [Online]. Available: <https://nl.mathworks.com/discovery/matlab-gui.html>.
- [9] Python, *Gui programming in python*, 2023. [Online]. Available: <https://wiki.python.org/moin/GuiProgramming>.
- [10] Electron, *Build cross-platform desktop apps with javascript, html, and css: Electron*, 2023. [Online]. Available: <https://www.electronjs.org/>.
- [11] FutureLearn, *How to create your first gui*, 2023. [Online]. Available: <https://www.futurelearn.com/info/courses/programming-with-guis/0/steps/60577>.
- [12] F. Abdoune, L. Rifi, F. Fontanili, and O. Cardin, “Handling uncertainties with and within digital twins,” in Feb. 2023, pp. 118–129, ISBN: 978-3-031-24290-8. DOI: 10.1007/978-3-031-24291-5_10.
- [13] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, “Characterising the digital twin: A systematic literature review,” *CIRP Journal of Manufacturing Science and Technology*, vol. 29, pp. 36–52, 2020, ISSN: 1755-5817. DOI: <https://doi.org/10.1016/j.cirpj.2020.02.002>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1755581720300110>.
- [14] Y. Jiang, S. Yin, K. Li, H. Luo, and O. Kaynak, “Industrial applications of digital twins,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, p. 20 200 360, Oct. 2021. DOI: 10.1098/rsta.2020.0360.
- [15] B. R. Barricelli, E. Casiraghi, and D. Fogli, “A survey on digital twin: Definitions, characteristics, applications, and design implications,” *IEEE Access*, vol. 7, pp. 167 653–167 671, 2019. DOI: 10.1109/ACCESS.2019.2953499.
- [16] Deloitte, *Digital twins*, Jan. 2020. [Online]. Available: <https://www2.deloitte.com/us/en/insights/focus/tech-trends/2020/digital-twin-applications-bridging-the-physical-and-digital.html/#endnote-7>.

- [17] R. Saracco, "Digital twins: Bridging physical space and cyberspace," *Computer*, vol. 52, no. 12, pp. 58–64, 2019. DOI: 10.1109/MC.2019.2942803.
- [18] D. A. Norman, "Cognitive engineering," *User centered system design*, vol. 31, no. 61, p. 2, 1986.
- [19] Gartner, *Definition of gui (graphical user interface) - gartner information technology glossary*, 2023. [Online]. Available: <https://www.gartner.com/en/information-technology/glossary/gui-graphical-user-interface>.
- [20] N. Tractinsky, A. Katz, and D. Ikar, "What is beautiful is usable," *Interacting with Computers*, vol. 13, pp. 127–145, Dec. 2000. DOI: 10.1016/S0953-5438(00)00031-X.
- [21] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '90, Seattle, Washington, USA: Association for Computing Machinery, 1990, pp. 249–256, ISBN: 0201509326. DOI: 10.1145/97243.97281. [Online]. Available: <https://doi.org/10.1145/97243.97281>.
- [22] M. Hassenzahl, M. Burmester, and F. Koller, "Attrakdiff: Ein fragebogen zur messung wahrgenommener hedonischer und pragmatischer qualität," in *Mensch Computer 2003: Interaktion in Bewegung*, G. Szwillus and J. Ziegler, Eds., Stuttgart: B. G. Teubner, 2003, pp. 187–196.
- [23] Akamai, *Akamai online retail performance report — akamai*, 2017. [Online]. Available: <https://www.akamai.com/newsroom/press-release/akamai-releases-spring-2017-state-of-online-retail-performance-report>.
- [24] M. Thuring and S. Mahlke, "Usability, aesthetics and emotions in human–technology interaction," *International Journal of Psychology - INT J PSYCHOL*, vol. 42, pp. 253–264, Aug. 2007. DOI: 10.1080/00207590701396674.
- [25] P. N. Johnson-Laird, *Mental models : towards a cognitive science of language, inference, and consciousness*. Cambridge, MA: Harvard University Press, 1983, 528 p. Excerpts available on Google Books. [Online]. Available: <https://hal.science/hal-00702919>.
- [26] R. Nienhuis, M. van Rooij, W. Prins, B. Jayawardhana, and A. Vakis, "Investigating the efficiency of a novel offshore pumped hydro energy storage system," 2023.
- [27] J. Sweller, J. J. G. Van Merriënboer, and F. Paas, "Cognitive architecture and instructional design," *Educational Psychology Review*, vol. 10, pp. 251–, Sep. 1998. DOI: 10.1023/a:1022193728205.
- [28] M. Wertheimer, "Laws of organization in perceptual forms," in *A Source Book of Gestalt Psychology*, W. Ellis, Ed., London: Routledge and Kegan Paul, 1938, pp. 71–88.
- [29] J. Preece, H. Sharp, and Y. Rogers, *Interaction design: beyond human-computer interaction*. John Wiley & Sons, 2015.
- [30] J. Nielson, "Match between system and real world: 2nd usability heuristic explained," *Nielsen Norman Group*, 2018. [Online]. Available: <https://www.nngroup.com/articles/match-system-real-world/>.
- [31] A. Cairo, *The Functional Art: An Introduction to Information Graphics and Visualization (Voices That Matter Series)*. New Riders, 2013, ISBN: 9780321834737. [Online]. Available: <https://books.google.nl/books?id=BiTlUGAACAAJ>.

Appendix A: Reference Material

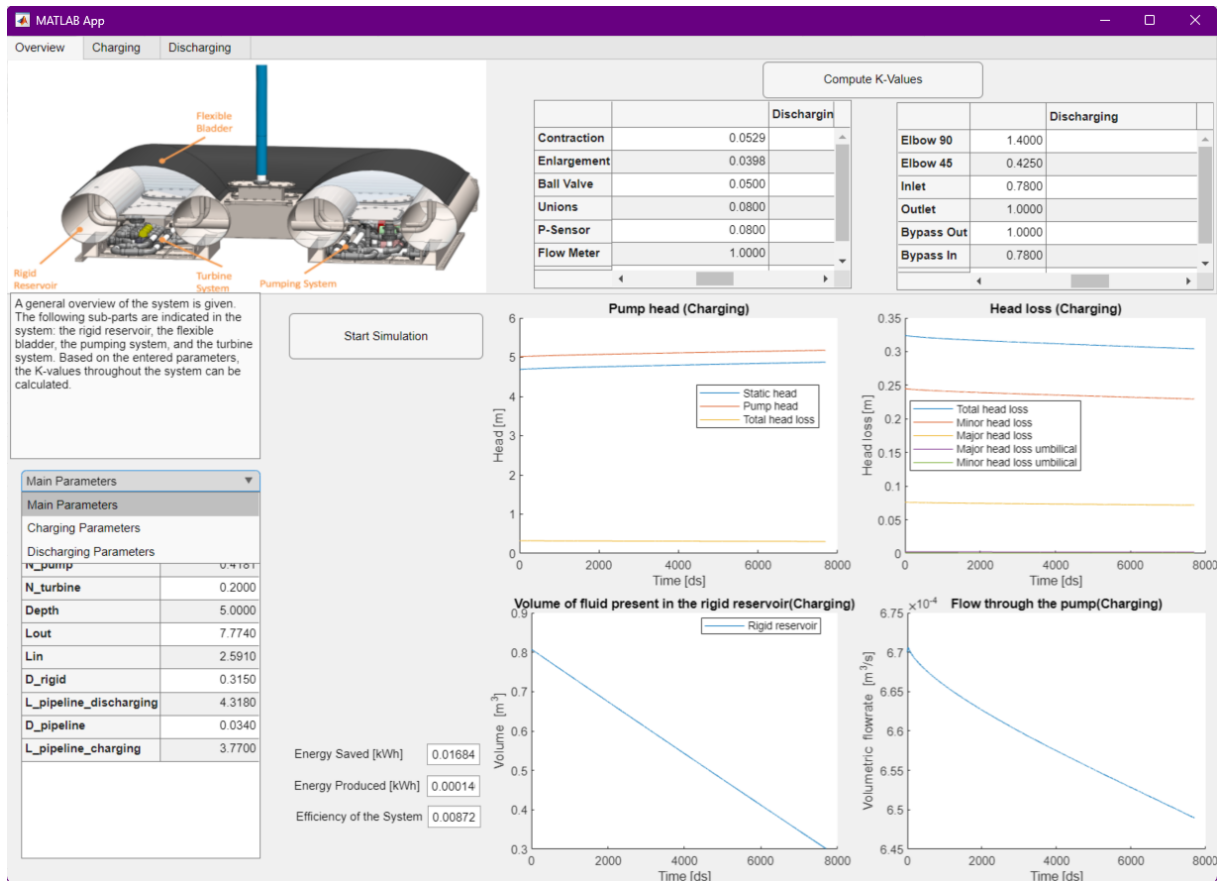


Figure A.1: View of the Overview tab of the existing GUI. The tab has a section above which shows the overall structure of the OB and tables that showcase the calculated K values. Towards the left is a text box that provides descriptions of various parts of the OB. Below the box is another table that shows the various parameters associated with the 2 different phases. The graphs depict the head, Head Loss, volume and flow, also associated with the phases.

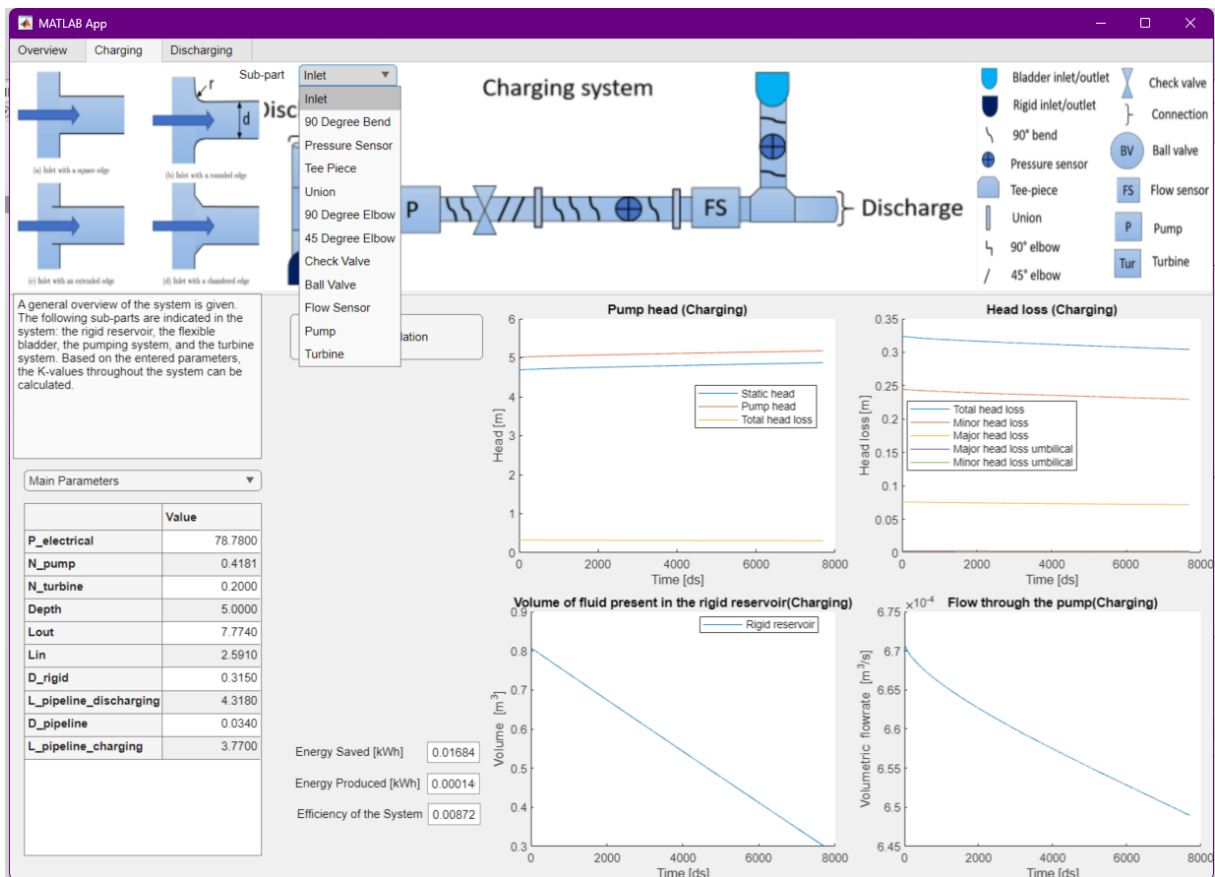


Figure A.2: View of the Charging tab of the existing GUI. The upper section of the tab shows an overview of the structure involved in the charging phase. A drop down menu allows the user to further select a part which will depict the image of the part on the top left with its description and associated parameters below.

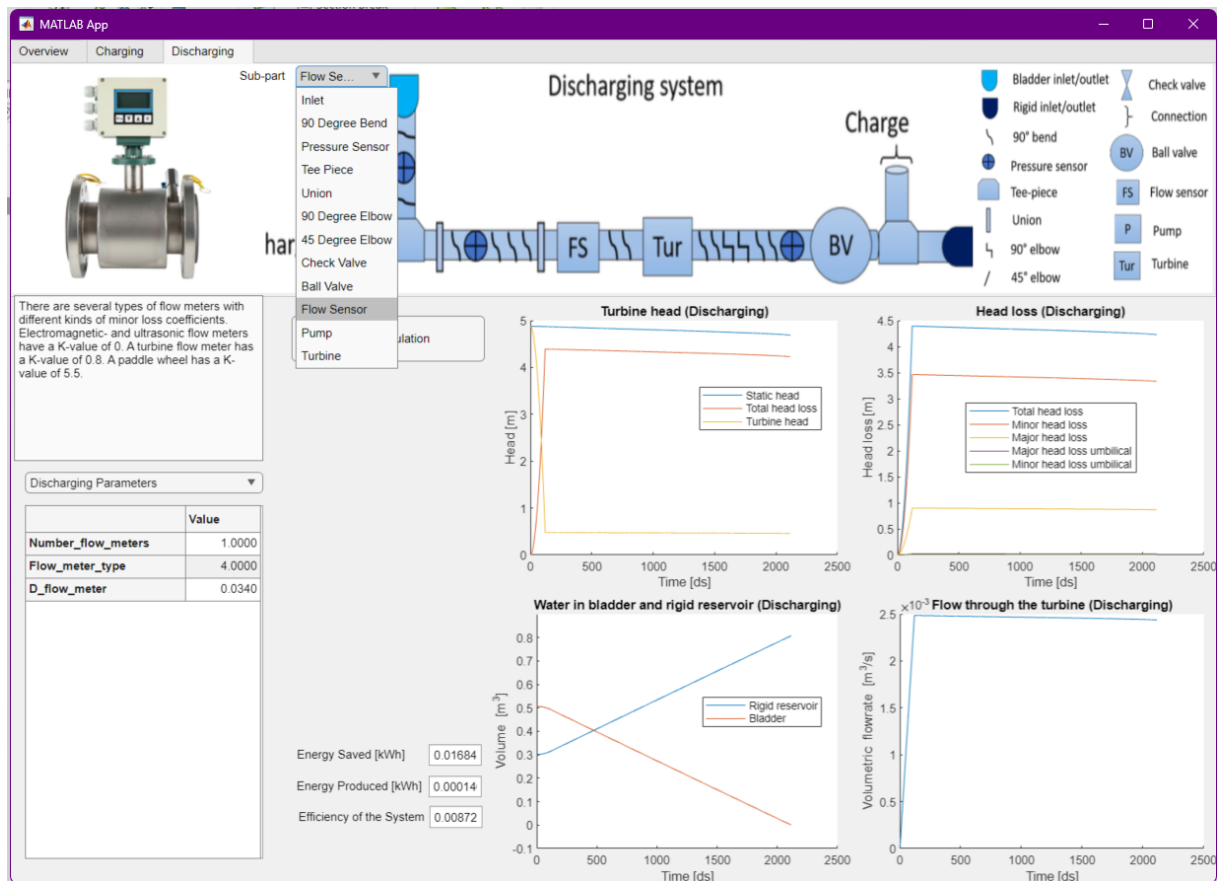


Figure A.3: View of the Discharging tab of the existing GUI. The upper section of the tab shows an overview of the structure involved in the discharging phase. A drop down menu allows the user to further select a part which will depict the image of the part on the top left with its description and associated parameters below.

Appendix B: Manual

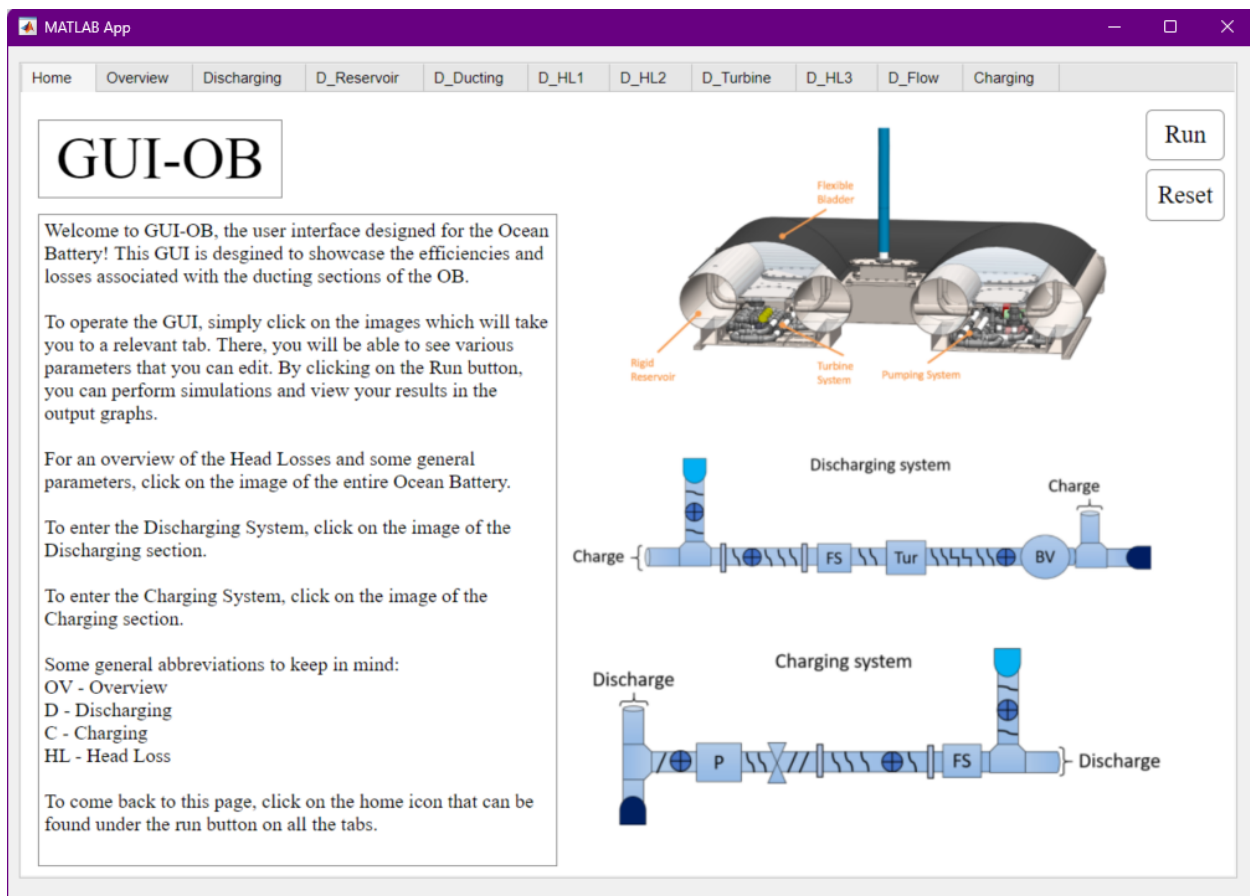
The manual begins on the following page.

GUI-OB

User Manual

V1.0

27/06/23



GUI-OB

Welcome to GUI-OB, the user interface designed for the Ocean Battery! This GUI is designed to showcase the efficiencies and losses associated with the ducting sections of the OB.

To operate the GUI, simply click on the images which will take you to a relevant tab. There, you will be able to see various parameters that you can edit. By clicking on the Run button, you can perform simulations and view your results in the output graphs.

For an overview of the Head Losses and some general parameters, click on the image of the entire Ocean Battery.

To enter the Discharging System, click on the image of the Discharging section.

To enter the Charging System, click on the image of the Charging section.

Some general abbreviations to keep in mind:
 OV - Overview
 D - Discharging
 C - Charging
 HL - Head Loss

To come back to this page, click on the home icon that can be found under the run button on all the tabs.

Run

Reset

Flexible Bladder

Rigid Reservoir

Turbine System

Pumping System

Discharging system

Charge

Charge

Discharge

Charging system

Discharge

Written by

Zoraiz Ali Syed

Contents

ABBREVIATIONS.....	2
PURPOSE.....	3
FILE PACKAGE.....	3
RUNNING THE GUI (NOT FINAL).....	4
STRUCTURE.....	6
GUI-OB STRUCTURE.....	6
GUI-OB CODE STRUCTURE.....	8
FUNCTIONS.....	9
EDITING THE GUI.....	10
DESIGN VIEW.....	10
CODE VIEW.....	11
APPENDIX.....	13

ABBREVIATIONS

Blad - Flexible Bladder

C - Charging

D - Discharging

DD - Ducting in Discharging

F - Flow

GUI - Graphical User Interface

H - Head

HI - Head Inlet

HL - Head Loss

HL1 - Head Loss section 1

HL2 - Head Loss section 2

HL3 - Head Loss section 3

HO - Head Outlet

HP - Hydrostatic Pressure

L - Length

OB - Ocean Battery

OG - Ocean Grazer

OV - Overview

Res - Rigid Reservoir

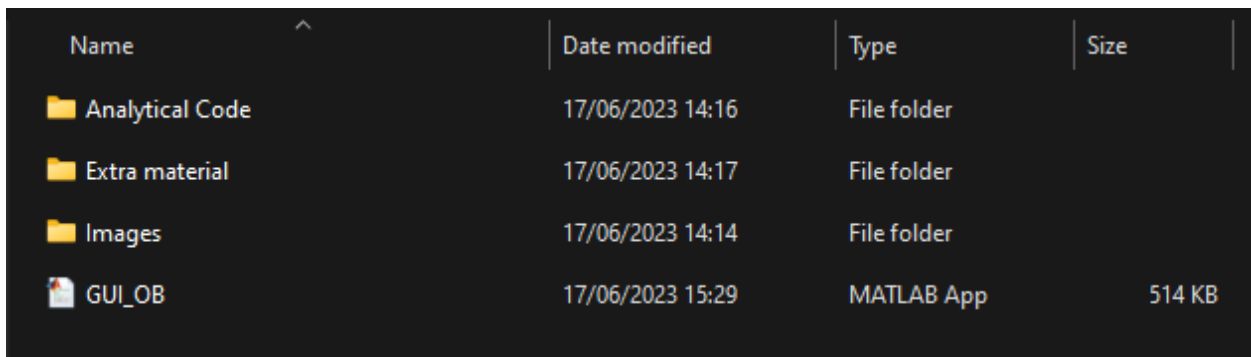
Turb - Turbine

PURPOSE

The user interface for the Ocean Battery is created to allow the user to modify parameters associated with the various sections within the OB and observe the resulting effects. This manual describes how the GUI-OB is set up and what the structure of the backend code is. This will allow future developments to be easily implemented in the GUI-OB without the need for creating new user interfaces.

FILE PACKAGE

The file package for GUI-OB is named GUI_OB_2023. It consists of 3 folders and 1 file, as seen below.



Name	Date modified	Type	Size
Analytical Code	17/06/2023 14:16	File folder	
Extra material	17/06/2023 14:17	File folder	
Images	17/06/2023 14:14	File folder	
GUI_OB	17/06/2023 15:29	MATLAB App	514 KB

Analytical Code refers to the folder that contains the matlab scripts and functions.

Extra material refers to the folder that contains material that is not used in GUI-OB but may be used later. This includes functions of the analytical code and images from the previous GUI.

Images refers to the folder that is used to input images for the GUI. This folder contains subfolders related to the CAD files of the OB which were used to create images. The images themselves which are input into the GUI can be found in the subfolder images_used.

GUI_OB is the main application. Opening this allows the user to modify the GUI code as well as run the application.

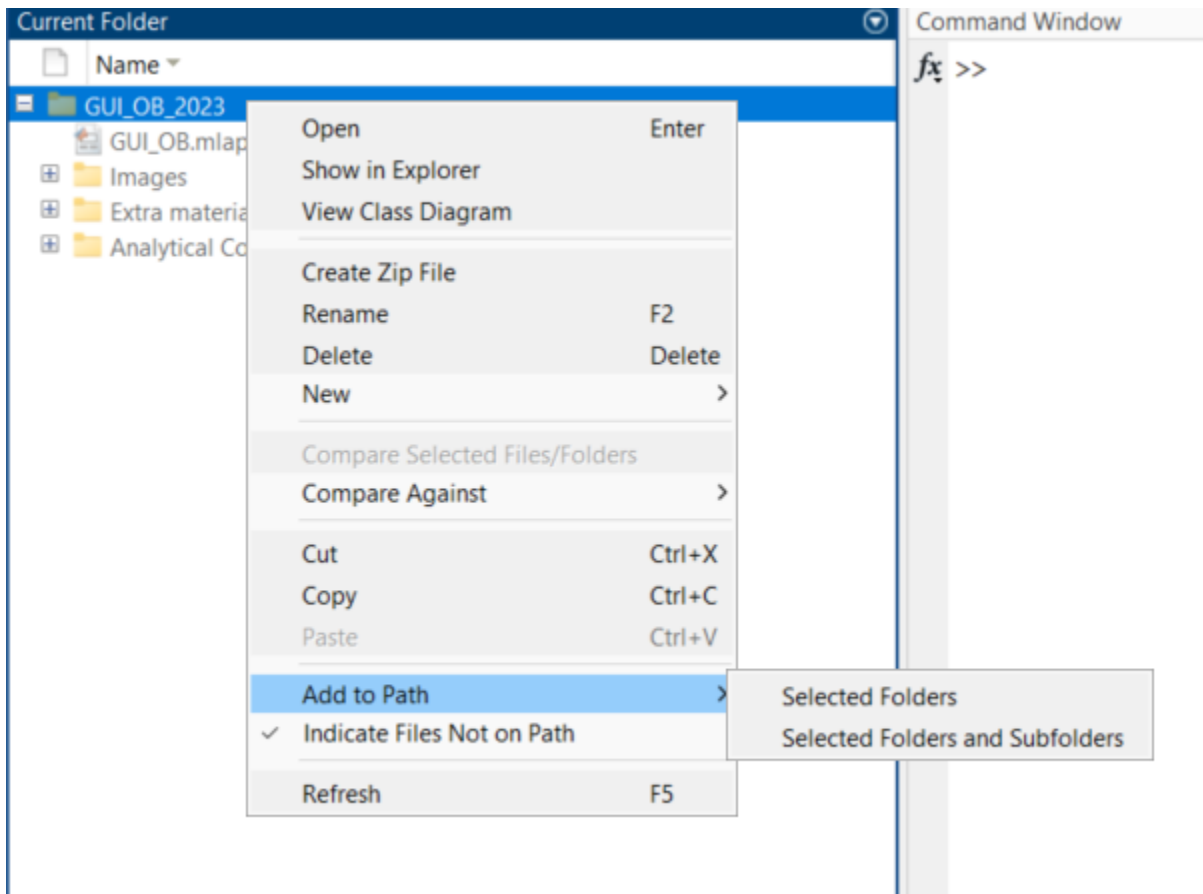
RUNNING THE GUI

Without Matlab

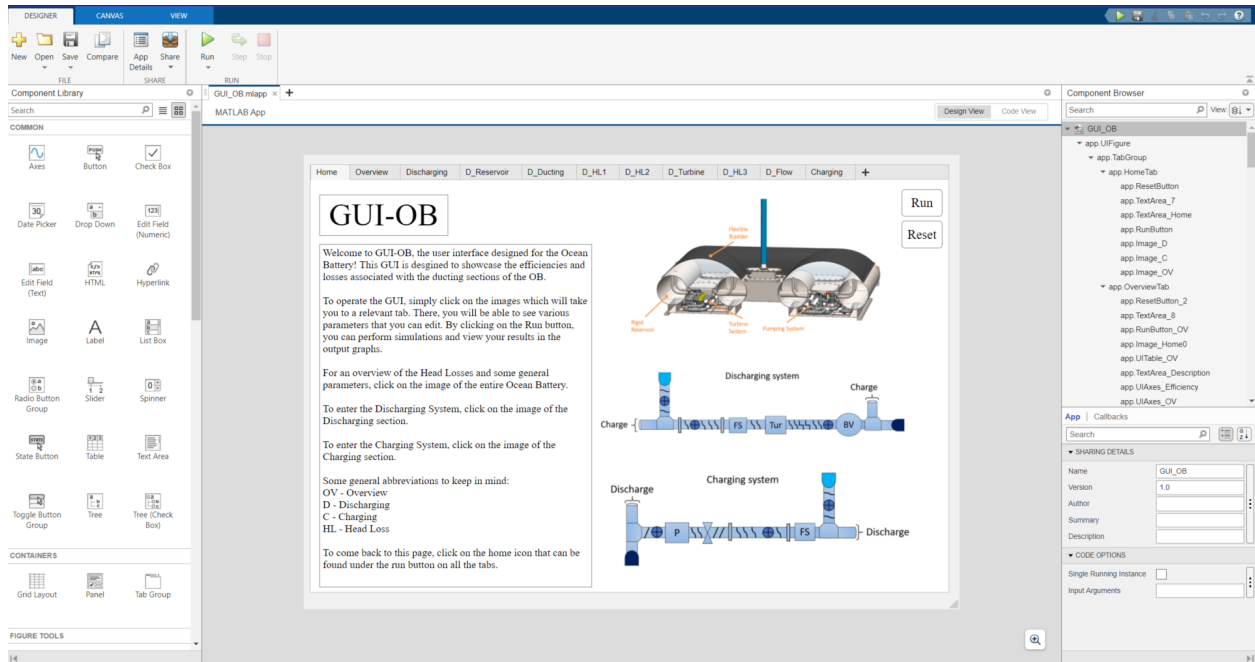
Run the filename “MyAppInstaller_mcr” and follow the instructions. Once the app is installed, follow the steps from below.

With Matlab

To run GUI-OB, the user must have Matlab installed on their workstation. After opening Matlab, the user must navigate to the location where the folder GUI_OB_2023 is located. In this location, the user first has to add all the subfolders to the file path. Without doing this, GUI-OB will not know the location of the functions and images used. Subfolders can be added to the path by right clicking on the folder, selecting “Add to Path” and sub selecting “Selected Folders and Subfolders”. This is visualised in the image below.

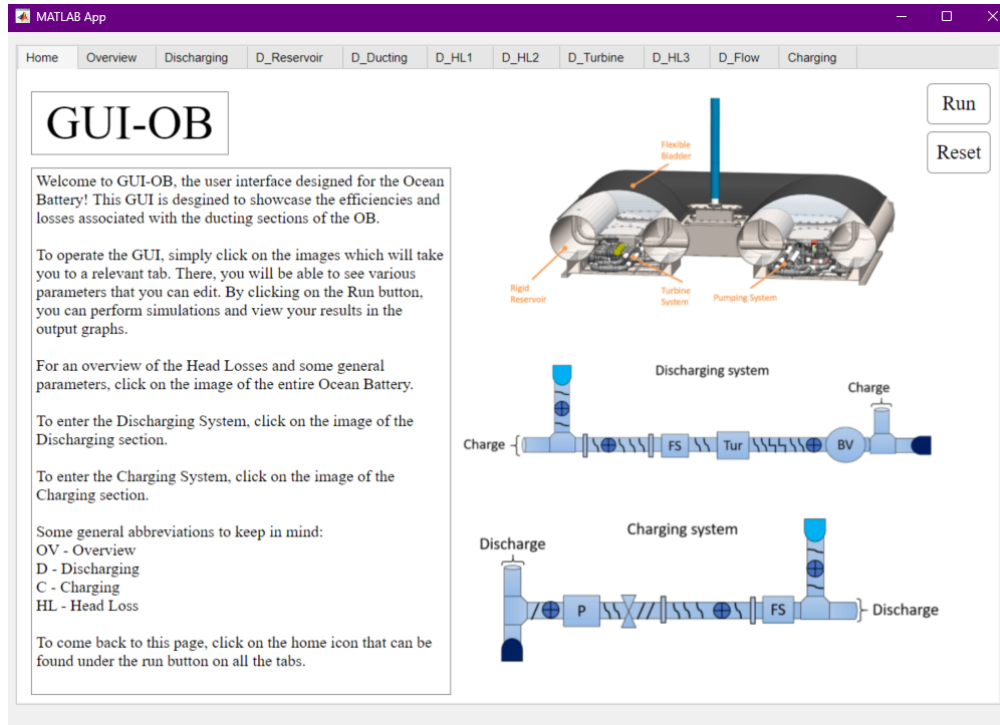


Once all folders have been added to path, the file GUI_OB can be opened. This will result in the following screen showing.



From this screen, the user can choose to run the GUI by pressing on the green Run arrow above. Additionally, the user can choose to customise the layout by staying on Design View and editing the GUI visually, or change the code of the GUI by going on Code View.

When the GUI is run, a new window appears which is shown below. This is the main “Home” page of the GUI. Navigating the GUI is intuitive and is aided by the descriptions in each tab. Navigation is done by clicking images which are a representation of different components of the OB. Each tab of the GUI is affiliated with a specific component of the OB and contains tables of parameters. These tables can be edited to change the values of parameters and observe the resulting effects. The outputs are visualised by the use of graphs. Each tab also contains additional images which are a representation of elements of that section.



STRUCTURE

GUI-OB STRUCTURE

GUI-OB has a structure consisting of multiple tabs that can be navigated through the use of images. By clicking on relevant images, the user will be taken to the associated page where various parameters and graphs can be seen. Each tab has a description related to what is depicted on it. The user is then allowed to edit parameters and observe the outputs by clicking on the “Run” button. If a user wishes to reset the parameters to their original value, it can be done by clicking on the “Reset” button.

Figure 1 below shows a tree diagram depicting the structure of the GUI-OB. When the application is opened, the user can see a “Home” tab. This tab contains images that link to an “Overview” tab as well as tabs related to the “Discharging” and “Charging” sections. The user can choose to begin the simulation right away by clicking the “Run” button which is also present on this tab.

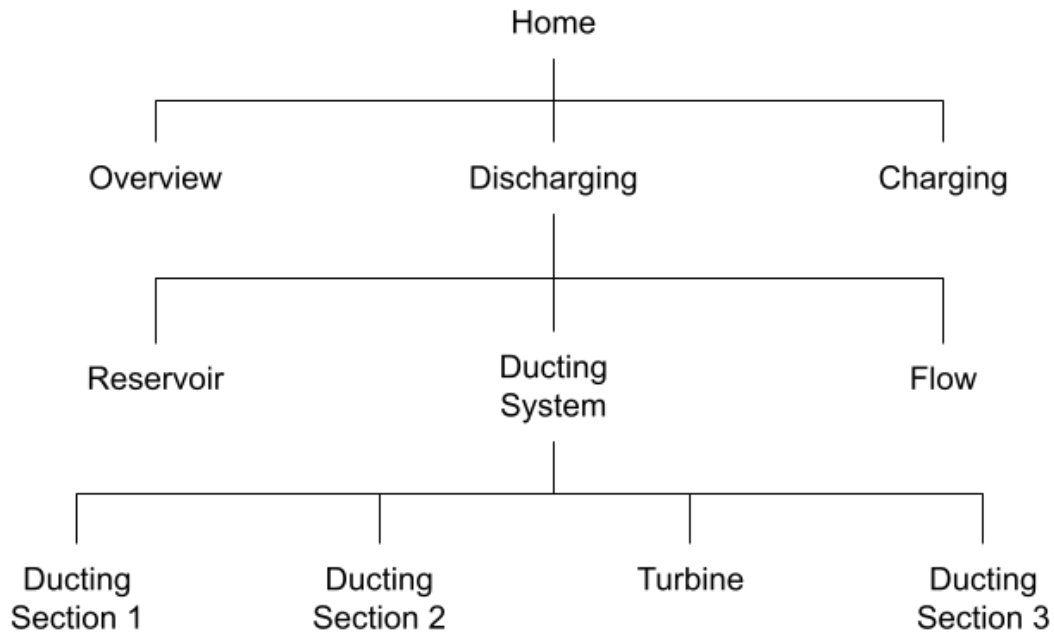


Figure 1: Tree diagram depicting the page structure in GUI-OB. The lines represent the linkages between pages. Important to note here is that all pages within the Discharging System link back to the Discharging phase as well, while all pages within the Ducting System link to each other and the main Ducting System page.

In the “Overview” tab, the user sees a table of parameters that include various efficiencies. A graph depicting the total efficiency of the OB over time is also shown as well as a graph depicting the various Head Losses in different sections. Like every tab, there are “Run” and “Reset” buttons as well as an icon that links back to the “Home” tab.

In the main “Discharging” tab, the user sees a description of the discharging phase along with a graph depicting the hydrostatic pressure. An image of the discharging system can also be seen along with images of various components of the system. These images are clickable and will take the user to the respective tabs of those components. A legend is also displayed that correlates with the image of the entire discharging system.

If the user clicks on the image linked to the “Reservoir” tab, a description of that section along with its parameters (which are editable) and output graphs can be seen. This structure is similar for the “Flow” tab, the “Turbine” tab and the various “Ducting Section” tabs. All the “Ducting Section” tabs, including the general tab,

have an additional feature in that they are all linked to each other by images. These images are representative of the various sections. To navigate through different parts of the Discharging system, the user has to navigate back to the main “Discharging” tab and then click on the image of a particular section. However, since the ducting systems are part of one component, their tabs are linked with each other.

The “Charging” tab currently only shows an image of the Charging system and its description. Due to specific data not being available for this system, the GUI does not go in depth into its components.

GUI-OB CODE STRUCTURE

The code for GUI-OB is structured as follows:

1. Properties Creation - In this section, the variables that are to be used in the GUI are defined. Matlab App Designer calls these variables properties and they are the first thing that the code should contain. If any additional variables are added to the model of OB, they should be included in this section as well.
2. Initialisation - This section assigns initial values to the variables when the GUI starts up. For the variables that have to be calculated later, this section initialises them by making them into an array (if required). Additional variables can be added to this section if they require an initial value or an array.
3. OB Parameters - In this section, the variables are assigned to the array “OB Parameters”. This is done to store all the variables in a single array which is later called in the various functions. This is also done when the GUI first starts up.
4. Run button - When the “Run” button is clicked, the GUI executes this code. A simulation message first appears to indicate that the simulation is running. This is followed by calling the various functions that are essential to the code. When all functions have been run, the graphs are created or updated and an end simulation message is displayed.

5. Image click - Whenever any image is clicked, it opens the related tab. The table of parameters as well as any graphs are then created for the relevant tab.
6. Table edit - Whenever new data is entered into the table, the previous data is overwritten and the table will display the new value that the user inputs. This data can then be used in simulations by clicking the run button. Additional input to the tables can be done by adding the variables in the if function as well as including them in the code for creation of tables.
7. Reset button - By clicking the “Reset” button, all values are reset to their original state and the graphs as well as the tables depict their original outputs.

The naming of images, tables and buttons has been done in such a way that it is representative of the relevant tabs. Additional components can be named in the same way to allow for clarity.

FUNCTIONS

There are 4 main functions that the GUI code utilises. These are as follows:

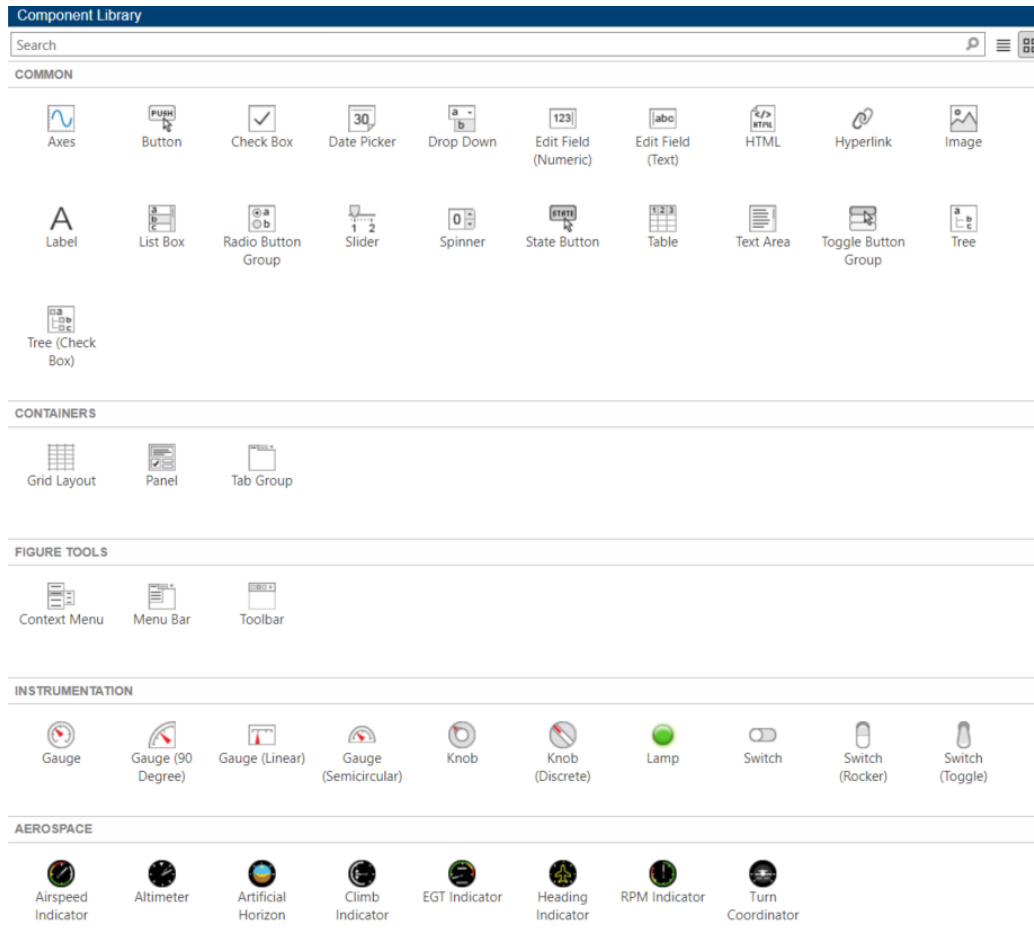
1. Main loop - This loop iterates over the specified time horizon. It calculates the flow velocities, Head Losses, friction coefficients and the depth. If changes have to be made in this code, it is important to factor in the loop and the fact that the calculations are mainly dependent on the flow and flow velocity values.
2. Head Loss - This function calculates the Head Losses in the bladder, turbine and rigid reservoir. As inputs, it utilises geometries defined in the initialisation section as well as certain elements calculated in the main loop such as the depth of water in relation to sensors, the viscosity and density.
3. Analytic Parameters - This function calculates the Head Losses from above but takes into account the reference point of sensors.
4. Efficiency - This function calculates the efficiency of the OB by using the efficiencies initialised for the motor and turbine, as well as the Head Losses calculated in the main loop.

For any changes made to the functions, it is vital to include them in the properties section of the GUI (if additional variables are added) as well as add them to the “OB Parameters” section.

EDITING THE GUI

DESIGN VIEW

In Design View, the user can add elements to the visual design of the GUI. These can be done through the elements present in the component library of Matlab, a snippet of which can be seen below. Amongst others, the user can add graphs, tables, buttons, labels and text boxes. The user simply drags the component from the component box and places it to the location where they would like the component to be. Once the component is added, the user can assign functions to it to determine any special behaviour. This could be the GUI performing a task when a button is clicked or allowing values to be updated in a table when it is edited. This is done by assigning callback functions and further documentation related to this can be found in the appendix.



CODE VIEW

In Code view, the user is able to make changes to the actual code of the GUI. This is where callback functions are programmed to perform tasks. The image below shows a callback function for editing values in a table of GUI-OB. Before functions are executed however, the variables have to be defined. This is done by creating properties in code view. Properties are the variables that will be utilised when the GUI is run. For example, GUI-OB calculates various Head Losses and the total Head Loss is stored in the variable HL_total. This variable is thus first defined as a property to be used. Programming the GUI has a simple general procedure: the user adds components to the GUI in the design view, assigns functions to these components and within the code view, defines what the function is meant to do.

```

end

% Cell edit callback: UITable_D_Turb
function UITable_D_TurbCellEdit(app, event)

%% Table

indices = event.Indices;
newData = event.NewData;

% allowing for edits in ducting turbine section table

if indices(1)==1
    if newData<0
        uialert(app.UIFigure,'Please use a positive value as input','Input Error');
    else
        app.OB_parameters.HL2_TU_90elbowbend_n = newData;
    end
elseif indices(1) == 2
    if newData<0
        uialert(app.UIFigure,'Please use a positive value as input','Input Error');
    else
        app.OB_parameters.HL2_TU_90elbowbend_k = newData;
    end
elseif indices(1) == 3
    if newData<0
        uialert(app.UIFigure,'Please use a positive value as input','Input Error');
    else
        app.OB_parameters.HL2_TU_contr_n = newData;
    end
elseif indices(1) == 4
    if newData<0
        uialert(app.UIFigure,'Please use a positive value as input','Input Error');
    else
        app.OB_parameters.HL2_TU_contr_k = newData;
    end
elseif indices(1) == 5
    if newData<0
        uialert(app.UIFigure,'Please use a positive value as input','Input Error');
    else
        app.OB_parameters.HL2_TU_enl_n = newData;
    end
end

```

APPENDIX

General Documentation Matlab App Designer

<https://nl.mathworks.com/products/matlab/app-designer.html>

Callback Functions in Matlab App Designer

https://nl.mathworks.com/help/matlab/creating_guis/write-callbacks-for-gui-in-app-designer.html