University of West Bohemia in Pilsen
Faculty of Applied Sciences
Department of Cybernetics

MASTER'S THESIS

HIL simulator for analysis and demonstration of smart control algorithms for overhead cranes

Pilsen, 2020 Milan Vosáhlo

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Milan VOSÁHLO**
Osobní číslo:	**A17N0021P**
Studijní program:	**N3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Téma práce:	**HIL simulátor pro analýzu a demonstraci chytrých algoritmů řízení pohybu jeřábů**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1) Analyze current trends in MBSE (model based system engineering) and especially the role of hardware-in-the-loop simulation both in terms of industrial application and possible enhancement of related education methods.
2) Compare the available technologies for creating HIL simulators in terms of price, performance and other factors.
3) Create a suitable dynamic model of an overhead crane with variable weight and rope length
4) Describe the principles of smart crane control, including input shapers.
5) Create a REXYGEN-based HIL simulator for a typical crane
6) Analyze the possibility of implementation of a simulator control part running on a commercial hardware designed for control of real cranes.

Rozsah diplomové práce:	**40-50**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování diplomové práce:	**tištěná**
Jazyk zpracování:	**Angličtina**

Seznam doporučené literatury:

[1] Reitinger, J., Čech, M., Schlegel, M., Balda, P. New tools for teaching vibration damping concepts: ContLab.eu (2014) IFAC Proceedings Volumes (IFAC-PapersOnline), 19. pp. 10580-10585.
[2] Astrom, K. J., Hägglund, T. „PID Controllers: Theory, Design, and Tuning." 2nd Edition, Research Triangle Park : ISA-The Instrumentation, Systems and Automation Society, 1995.
[3] Reitinger, J., Cech, M., Goubej, M. Advanced input shaping filter 3D virtual laboratory (2013) Proceedings of the 2013 International Conference on Process Control, PC 2013, art. no. 6581465, pp. 528-533.
[4] Schlegel, M., Goubej, M. Feature-based parametrization of input shaping filters with time delays (2010) IFAC Proceedings Volumes (IFAC-PapersOnline), 43 (2 PART 1), pp. 247-252.
[5] J. Sobota, M. Goubej, J. Konigsmarkova, M. Cech. Raspberry Pi-based HIL simulators for control education (2019). In Proceedings of IFAC ACE 2019. Philadelphia (UA).

Vedoucí diplomové práce:	**Ing. Martin Čech, Ph.D.** Katedra kybernetiky
Datum zadání diplomové práce:	**1. října 2019**
Termín odevzdání diplomové práce:	**25. května 2020**

[illegible]

Západočeská univerzita v Plzni
L.S.
Fakulta aplikovaných věd

[illegible]

Doc. Dr. Ing. Vlasta Radová
děkanka

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 1. října 2019

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 7. 7. 2020 . . . .

Declaration

I present my master’s thesis elaborated at the end of my studies at the Faculty of Applied Sciences of the University of West Bohemia in Pilsen.

I declare that I carried out this master’s thesis independently, only with the cited sources and literature.

Acknowledgements

I would like to express my sincere thanks to my supervisor, Ing. Martin Čech, Ph.D., for valuable advice during the process of writing this thesis.

I would also like to extend my gratitude to Ing. Jaroslav Sobota, Ph.D. for technical support with HIL related problems, to Bc. Jan Váverka for model derivation assistance, to Jan Procházka for CAD crash course and to Ing. Ondřej Severa for HMI quick tutorial.

Abstract

This thesis is creating a Gantry crane Hardware-In-the-Loop simulator by following through the X-In-the-Loop process. It begins with mathematical model derivation and validation and PID controller tuning and input-shaper to cancel oscillations, is tested in Simulink in Model-In-the-Loop configuration, then in Rexygen in real-time simulation as a Software-In-the-Loop and ends as a HIL simulator of the system and another one of the controller with communication via Modbus and analog signal for control and sensor emulation. It also boasts 3D visualisation as part of the system HIL and web human-machine-interface to command the controller.

Keywords: HIL Simulation, Model-Based System Engineering, Input-Shaping Filters, XIL process, Gantry Crane, 3D Visualisation, Human-Machine Interface, Monarco HAT

Anotace

Tato práce se zabývá tvorbou Hardware-In-The-loop simulátoru portálového jeřábu a tím i celým X-In-the-Loop procesem. Začíná odvozením matematického modelu jeřábu, jeho validací, návrhem PID regulátoru a zapojením vstupně-tarovacího filtru pro utlumení houpání nákladu. Tento systém byl otestován v prostředí Simulink jako Model-In-the-Loop a poté v Rexygenu v reálném čase v Software-In-the-Loop konfiguraci. Nakonec je model systému a regulátoru spuštěn jako HIL na dvou počítačích, které spolu komunikují přes Modbus a analogově (pro emulaci řídícího signálu a senzorů). K tomu vykresluje HIL simulující systém 3D vizualizaci a HIL s regulátorem hostuje webové rozhraní pro ovládání systému.

Klíčová slova: HIL Simulace, MBSE, Vstupně-tvarovací filtry, XIL proces, Portálový Jeřáb, 3D Vizualizace, Uživatelské rozhraní, Monarco HAT

Contents

List of figures

List of abbreviations

3D	Three Dimensional
A/D	Analog to Digital
ARM	Processor architecture (Advanced RISC Machine)
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAN	Controller Area Network
D/A	Digital to Analog
FMI	Functional Mock-up Interface
HAT	Hardware Attached on Top
HIL	Hardware-In-the-Loop
HMI	Human-Machine Interface
HW	Hardware
I/O	Input/Output
INCOSE	The International Council on Systems Engineering
IoT	Internet of Things
KBSE	Knowledge-Based System Engineering
KM	Knowledge-Management
MBSE	Model-Based System Engineering
MIL	Model-In-the-Loop
OS	Operational System
PID	A controlled consisting of Proportional, Integral and Derivative components
PIL	Processor-In-the-Loop
SIL	Software-In-the-Loop
STEM	Science, Technology, Engineering, Mathematics
TCP/IP	Transmission Control Protocol/Internet Protocol
XIL	X-In-the-Loop

Chapter 1

Introduction

There are growing demands for industrial applications of the Internet of Things, predictive maintenance and trackability of individual parts (quality monitoring), which results in a necessity of greater effectiveness of model-based and knowledge-based system engineering methods. Engineering areas like lifecycle management, X-In-the-Loop (XIL) [27] and standards like FMI [2] have grown rapidly in recent years, mostly through automation of tasks that allows for iterative design control [17] at a faster pace than ever before. This progress is reflected in demands for system engineers that need to work in ever-more complex environments and therefore also needs for tools [20] that allow for grasping larger amounts of data and features easily.

Education systems need to adapt to this always shifting environment. For universities, this means that partnerships with companies in their respective industries are vital. For control engineering courses though, there are tendencies to avoid teaching control on real physical systems [28], as they take a lot of effort to maintain and have several other disadvantages. However, there are aspects of controller implementation that rarely appear in purely software simulations, which means that abandoning physical control systems and their specific challenges makes the control education poorer and the graduate cannot be immediately utilised for practical applications as there are obstacles that the student did not encounter. This can be solved by applying HIL simulators [18] with industry-grade I/O. This method used to be domain of only large companies (for example Dspace), bringing significant costs for employing these methods, but as tiny computers like arduino and Raspberry Pi gained popularity and became powerful enough to accomodate simulations in real-time, and option appeared to apply them for educational purposes on the cheap side. Sure, they are less robust, but due to their accessibility, problems can be solved swiftly and downtime brings little drawback at school, compared to a large company. This also allows for teaching the whole XIL process.

For education and sharing experience in companies, knowledge management methods are being employed [6]. They can get new employees up and running and make knowledge transfer among employees a smoother process, although at a cost of increased bureaucracy load as processes have to be noted constantly.

1.1 Motivation and Objectives

Main motivation for this thesis was getting hands-on experience on the XIL process as a whole and to apply skills the author learned during his exchange semesters, mainly working with CAD software and creating human-machine interface. The main objective was creating a HIL simulator of a gantry crane, which means a model had to be derived and verified and a controller designed for this model, applying the Model-based knowledge [25] in all of Model-In-the-Loop and Software-In-the-Loop phases of the XIL process. A part of this thesis touches the topic of input shaping [21] in order to cancel oscillations of the crane.

1.2 Structure of the thesis

The thesis structure goes as follows: Chapter 2 describes the current situation and solutions that are being developed and applied. Chapter 3 depicts a gantry crane model formation and controller synthesis. Along with the model, a 3D visualisation was drafted. Chapter 4 concerns the HIL simulation and its challenges, as well as finishing the 3D visualisation and HMI interface.

Chapter 2

Model-based system engineering

Model-based system engineering was created as a response to market demands, such as requirements for faster and smarter design and production processes [10] with traceable component defects. In other words, more functionality in a shorter time at the lowest cost. These days, more emphasis is put on software that needs to comply with many standards and regulations [20]. For improved response time and clearer specification between suppliers and manufacturers standards like FMI (Functional Mock-up Interface) have been introduced [2] and gained significant traction over the last few years, with large automotive and aerospace companies adopting these standards to improve their competitive value. With all of these factors considered, MBSE is a recommended practice by INCOSE (The International Council on Systems Engineering) [4], Department of Defense of the United States [13] and many other impactful organisations representing the industry.

Models are simplified digital representations of key system aspects depicted in a comprehensible way. Typically, a model is standardised so it is transferable across various workgroups. Validation of system features is done by simulation that allows for virtual integration even before some components are designed and before physical prototypes are built, saving both time and financial resources. It is also possible to automatically generate source code for model simulation or visualisation and automate other tasks in the design and production phase. Using this method, errors can be avoided or at least discovered at an early design phase when it is cheaper to correct them.

The system development life cycle (typically represented by the “V diagram” (Figure 2.1)) is an iterative process, meaning most of its parts are repeated several times as needs for change arise later in the process. Using older methods would require manually performing each individual step with (slightly) different input than during previous iteration — a tremendously costly process. The MBSE approach allows for automation of described steps as well as transitions between them resulting in man-hours being saved and fewer errors occurring.

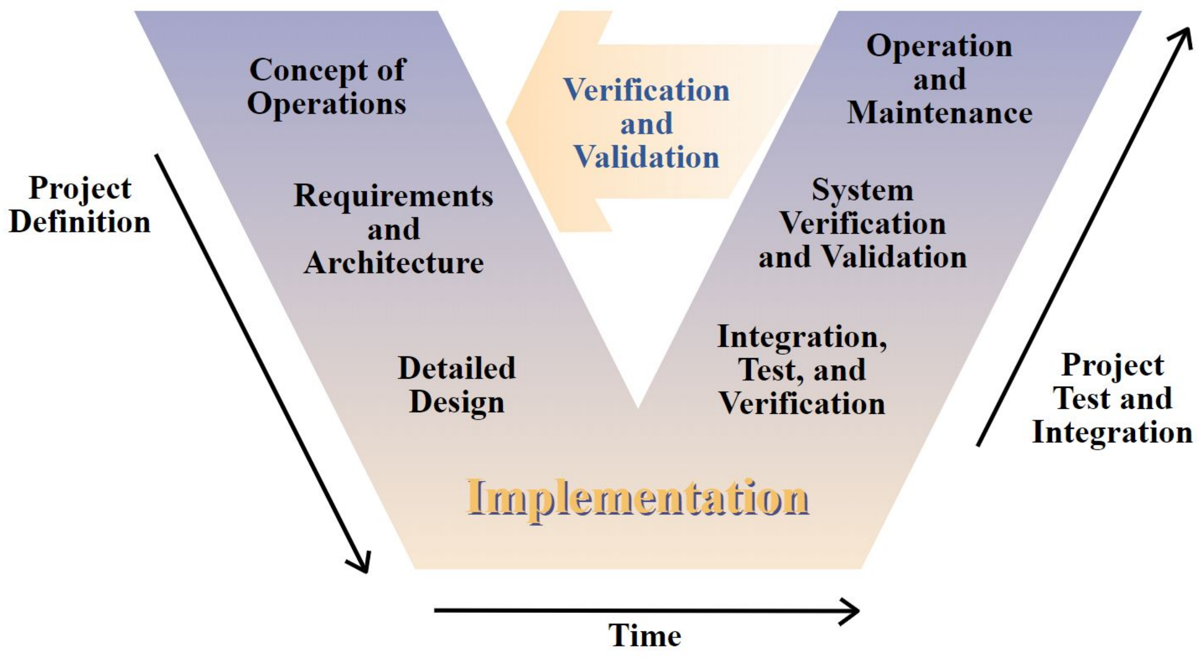

Concept of Operations
Requirements and Architecture
Detailed Design
Verification and Validation
Operation and Maintenance
System Verification and Validation
Integration, Test, and Verification
Implementation
Project Definition
Project Test and Integration
Time

Figure 2.1: *V-diagram, used to visualise the lifecycle of a product. The left half is decomposing the product into elemental parts or tasks and the right half builds up. What is not always obvious is that this is an iterative process that typically revisits all design stages several times* [14].

There are many standards regarding system engineering, focusing on various areas like product life-cycle management, audits, integration, application, process assessment, etc. Most of them are assured by the INCOSE.

To sum up, model-based system engineering is today's most advanced method employed during all phases of the life cycle of products and affects various aspects, including concept, development, production and applied operations. Using these methods results in quality and productivity improvements and enhanced risk management during the process.

2.1 Knowledge-based system engineering

When manufacturing complex systems, there is a pressing need to understand project specifications, share and re-use knowledge and experience, and maintain clear communication between the designer and the customer. This is where knowledge-based system engineering comes in. Its scope covers all life cycle management phases [7, 23] and also allows for design optimisation [25]. Computer-aided design (CAD) software is a direct product of knowledge-based engineering and it is also an early adopter of the object-oriented way of thinking in applied engineering. The design goes hand-in-hand with computer-aided engineering (CAE) which helps with analytics of the given system and provides manufacturing support.

Since many products are part of some product line or are sub-products for various uses, there is a demand for the re-use of schemes, look-up tables, algorithms and other knowledge. To satisfy these growing environments, time-saving knowledge-driven automation methods have been developed. However, they take longer to adopt and learn. With many products having similar qualities, companies also use these methods for product distribution and disposal.

For larger companies, investing in knowledge management [8] is necessary as the engineers working for them are fluctuating [22] all the time, which means new employees need to be taught frequently. Knowledge management distinguishes two main categories of knowledge: explicit and tacit. Explicit knowledge is the one that can be taught by reading manuals, understanding standards, learning material specifications, etc. Tacit knowledge is tied to a person and is harder to transfer. It is based on experience, consists of various heuristics and even rules of thumb that just work. This discipline, just about 30 years old [24], is not particularly popular among engineers and is often overlooked by companies as it brings additional bureaucracy. However, it can lead to a competitive advantage.

2.2 Hardware-In-the-Loop Simulation

When systems require active control, Hardware-In-the-Loop (HIL) simulation is a vital part of the model-based system development, especially now, when each device has large amounts of smart sensors and controllers. Cars typically have hundreds or even thousands of controllers, which makes them very complex systems with high safety requirements, meaning every possible fault has to be tested and prepared for before the manufacturing process begins. This point is even stronger for the aerospace industry, but even home appliances nowadays can connect to the internet and gardening tools have CAN bus I/O for diagnostics. HIL is the final phase of the XIL process that will be explained in this subchapter.

The XIL process is used both for the controller and the plant design. It is an iterative process that tests and optimizes both the controller and plant for greater efficiency, which is the main drive for automation of the process to lower the cost of each iteration. The XIL process consists of four main phases (which are largely automated) and the transitions between them [15].

The first phase of the process is called Model-In-the-Loop (MIL). This phase is used to design the core controller features and can be used to design the plant as well. The main advantage of this phase is that the controlled system does not have to be created yet, allowing for parallel development of the controller and the controlled system. This allows both of them to have an influence on each other before any costly prototype is made. The model of a controlled system should include as many dynamics and characteristics as possible as the simulated environment is controlled and does not have to run in real-time speed or faster.

The next phase is Software-In-the-Loop (SIL) which emulates the controller implementation as if it would run on actual hardware. This verifies the controller source code and its communication with the plant and sensors. The plant model can only cover the dynamics related to the controller to save processing time.

The third phase is called Processor-In-the-Loop (PIL). The controller is running on dedicated hardware which is later used in the actual product but the plant model is still a simulated one with emulated communication. This setup needs to run in real-time to determine whether the processing time is fast enough to satisfy all demanded functions of the controller.

The last phase is Hardware-In-the-Loop (HIL). The finished controller functionality is verified with either the prototype of the plant or a simulation of it using the same communication means (both analog and digital I/O with A/D and D/A converters) that will later be used. This step for example allows us to simulate handling signal noises and various faults that would be destructive for the prototype. The last batch of automated tests runs here and when successfully completed, the controller is ready and can be, with good confidence, connected to the actual system and mass-produced.

This process has been a major part of this thesis. Starting with a model and a controller simulated in Simulink environment in MIL configuration, advancing through SIL simulation in Rexygen on one computer with separate tasks, skipping the PIL phase as it wouldn’t bring any value in this use-case and ending as a HIL simulator with two computers connected to each other using various types of ports to emulate sending both digital information via Modbus and analog signal via standard 0-10 V I/O. Visualisation of this process can be seen in Figure 2.2.

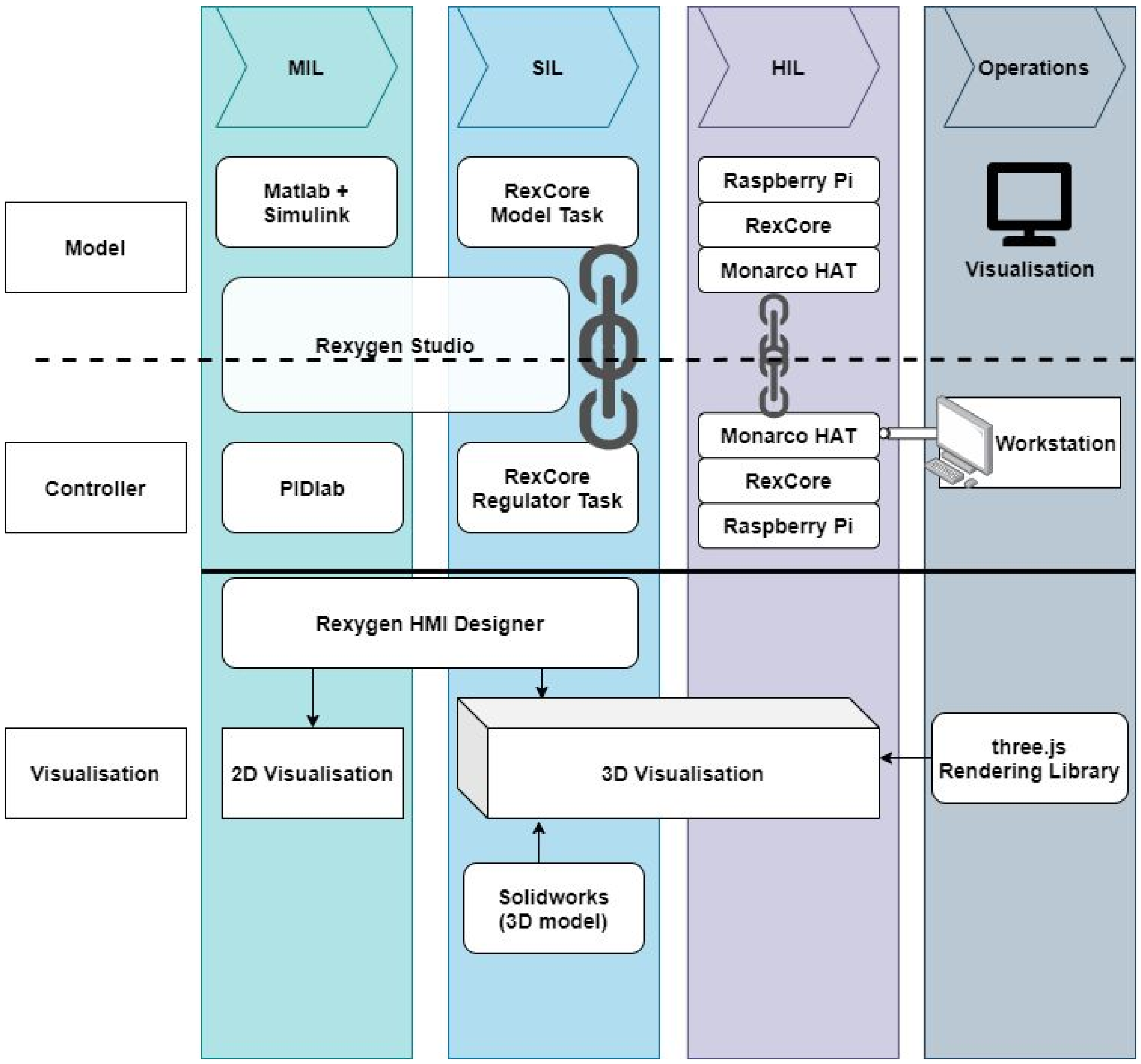


Figure 2.2: *Going through various phases of this thesis required use of several environments for each phase's challenges. This "timeline" chart shows which technology was used for what reason. All of these are explained later in this thesis.*

2.3 Industry focus

While MBSE ideas have been proposed in the '90s already [1], it wasn't until 2007 when the MBSE approach started to gain traction as it was endorsed by INCOSE in its MBSE 2020 Vision initiative [27]. This approach pushed the transition from document-centered specifications to model-centered ones. Around 2010, MBSE tools were standardised and commonly used in the industry. These days, proven model libraries are being commonly reused, saving precious time, and the standards have settled down with no major changes in functionality [19]. It was expected that MBSE would be used in other domains than engineering (like social, economic and political modeling) but that has not really been the case.

MBSE had the greatest impact in industries that manufacture complex products that need to cover several domains, and have large amounts of suppliers. Namely, automotive and aerospace industries have applied MBSE the most and also have benefited from it the most. One of the results of such development is the FMI standard [11]. It is an open standard developed mainly by Dassault Systémes and Daimler AG that was adopted by major companies in the industry and consequently by their suppliers. The standard specifies the format for exporting and importing simulation models (across domains and program environments) and running those simulations independently in the tool they were created in. This standard has two main functional features: model exchange and co-simulation. Model exchange describes models with standardised physical and informational variables so that it can be simulated using any solver. Co-simulation includes solver in C-functions, meaning the model can be simulated in steps with a solver that is not supported by the environment running the simulation. This standard has significantly decreased the amount of work needed to simulate the whole plant with parts from various suppliers using the design tools they prefer.

2.4 Leading MBSE Environments

Although there are many tools for applying MBSE methods, covering individual domains or specific use-cases, some universal or more popular ones are worth mentioning here. It is common for these environments to be able to share models or functions with one another, mainly as a result of industrial demands, not from the will of the corporations that create them as they would typically want a closed environment with customers solely relying on their products. Open-source software from academic and industrial associations created important incentives for interoperability of these environments, as they disrupted the market with features that allow for model exchange or co-simulation. Defining and maintaining interfaces between given platforms became vital parts of the platforms themselves.

2.4.1 Matlab

Matlab is a powerful programming tool used by many engineers and researchers worldwide. Its main domain is numerical matrix computing. It contains apps for system identification, controller synthesis, signal and image processing among many others. Main extension to Matlab is Simulink, a graphic environment for multi-domain simulation and model-based design. There are many libraries for Simulink for various domains, most notably SimScape, which covers a whole range of physical domains starting with electrical and mechanical model simulations ending with fluid and thermal dynamics. Besides simulations and calculations, it allows for code generation for various platforms, so algorithms created in Matlab can be used on specific hardware. Matlab can also host virtual laboratories and provides remote cloud computing servers. Matlab has a free alternative: GNU Octave. Naturally, free alternatives do not provide a professional level of support and so, for industrial applications, Matlab is prefered amongst companies.

2.4.2 Modelica-based programs

Modelica is a multi-domain language for component modeling. The language itself is open-source and is implemented in several programs.
To name a few:

- Openmodelica is an open-source simulation environment, commonly used in education.
- Jmodelica.org is an open-source platform for optimization and analysis of dynamic systems, which can also work as a virtual lab.
- Dymola (Dassault Systèmes), Amesim (Siemens), MapleSim (Maplesoft) and several others are commercial programs based on Modelica.

2.4.3 IBM Engineering Lifecycle Management

This bundle of applications supports product development from idea forming through deployment and continuous operation till disposal. The main merit is brought with everything being visualised in an easy way for humans to understand, which in the age of internet of things with unforeseen amounts of data and automatically generated specifications is an imposing feat. With systems having thousands of sensors and controllers, coordination of developing them all has to be provided in a systematic way. This is where task management comes in handy, as project managers can monitor the progress [9] of assignments. The tools can automatically collect progression data, lightening the workload of engineers that would otherwise have to fill in spreadsheets. Design management allows for sharing, reviewing and commenting on products in the design phase. Other features include: Requirements and quality management and automated reports builder.

2.4.4 LabView

LabView was created as a tool to increase productivity of scientists and engineers. Its graphical programming techniques are relatively easy to grasp and enable users to measure and monitor sensor data and process it using built-in tools for signal processing. Captured data can range from single sensors to big data, there are tools to accommodate needs of a multitude of occupations. The tools are also often used for academic demonstrations.

2.4.5 CAD

Computer aided design software is primarily used in mechanical engineering and architecture applications. The tools have features for optimisation of material used and manufacturing processes, but typically cannot satisfy all needs and phases of MBSE development. The software is able to perform finite elements simulations, for example for stress tests, deformation analysis or heat spreading through materials. Notable programs are: Autodesk Inventor, Solidworks and Catia, all of them offer basically the same basic functionality and differentiate mainly in business models.

2.5 Education perspective

Cooperation with industry has always been important for universities teaching STEM field courses. It is the only way to ensure that the substance of the courses stays relevant for the employer's needs. Because of that, digital twins of industrial systems appear during lectures to emulate rife struggles of engineers, as the digital twins are basically HIL simulators, running in real time creates challenges comparable to those in real industries [18]. These process models run either locally or on a distant server and are commonly accessed by LabView HMI. The models usually rely either on Matlab (with option to use cloud server computing) or easy Java sim (locally).

Another important trend is emphasis on edge computing. With the internet of things being on the rise in industry, engineers realised that having vast raw data does not bring any advantage and instead, smart sensors are processing the data at a plant and sending the results, which allows higher levels of automation pyramid to focus just on operation (for example, predictive maintenance). In other words, logical operations are spread over several levels, each working on designated tasks, instead of computing everything at one spot.

One aspect that is being adopted by universities during the last decade is gamification of assignments. When the design of the task manages to motivate the student, not only the assignment can yield better results, but also the student retains more of the knowledge learned.

Naturally, universities aren’t driven by profit and therefore cost concerns for working solutions are greater, while quality concerns are lower. If a HIL simulator doesn’t work at a large company with hundreds of suppliers and thousands of employees, profit is lost. If a HIL simulator acts up at school, one lecture might be ruined but solving the issue can work as a learning example, therefore industrial-grade robustness is not necessary. As processing power is getting cheaper over time, platforms like Arduino and Raspberry Pi became popular worldwide, enabling teaching methods that simply were not feasible before. The main use of Raspberry Pi is a real-time HIL simulator with industrial-like I/O, but in other fields or specialisations this platform serves for other purposes.

2.6 Technologies used in the context of this thesis

This chapter explains every software environment and hardware used in order to accomplish the practical tasks of this thesis. A visualisation (Figure 2.3) that categorises these techs can be found at the end of this chapter.

2.6.1 Matlab+Simulink

Matlab has already been described earlier in this chapter (2.4.1). Thanks to a vast amount of built-in functions, it has been the single most important program used for this thesis. For purposes of this thesis, it has been used for example to calculate state-space matrices of derived models for the PID controller tuning and to compare the results of various simulations.

Simulink was in this thesis mainly used to simulate various forms of crane models in MIL simulations.

2.6.2 Rexygen studio

Rexygen Studio is a development environment for automation projects. It is similar to Simulink from the user's perspective but processes run purely in real-time on RexCore on target devices with features that allow process monitoring and diagnostics. The studio contains a library of blocks, most of which focus on the control of processes and industrial communication. In this thesis, the SIL part, including the transition to the HIL phase, has been done exclusively using Rexygen.

2.6.3 Rexygen HMI

Rexygen HMI Designer is a graphics editor based on InkScape used to create visualisation and UI for controlled processes. The visualisation with control elements created in this thesis was made using this tool.

2.6.4 Three.js

Three.js is an Open-Source JavaScript library used for 3D rendering in a web browser in an easy and lightweight way. The 3D part of the visualisation in this thesis is using this library.

2.6.5 PIDlab

PID Control Laboratory is a java applet tool for the rapid design of PID controllers [12]. Its authors claim that it is possible to design a controller in under a minute, complying with various demands for robustness, speed and other relevant expectations. The PID regulators used to control crane models in this thesis were tuned using this tool, as the resulting parameters can be used directly in Rexygen PIDx models.

2.6.6 Solidworks

Solidworks is a CAD program developed by Dassault Systèmes that was used in this thesis to make 3D model assemblies that were used in the 3D visualisation.

2.6.7 Raspberry Pi

Raspberry Pi is a tiny ARM-based computer developed for computer science education purposes. The platform is open, running a variety of OS (Linux, Android, Windows 10 ARM64 and several others) and has been used in many applications with tens of millions of units sold worldwide. This platform has been used for the HIL simulations in this thesis.

2.6.8 Monarco HAT

Monarco HAT is a Raspberry Pi extension that allows the small consumer-grade computer to work with industrial hardware. In Rexygen there are applets to set-up Modbus Master/Slave communication and blocks that handle signal routing, but libraries for implementation using C and other programming languages are available. In the case of this thesis, the RS485 and both analog and digital I/O were handled using this HW.

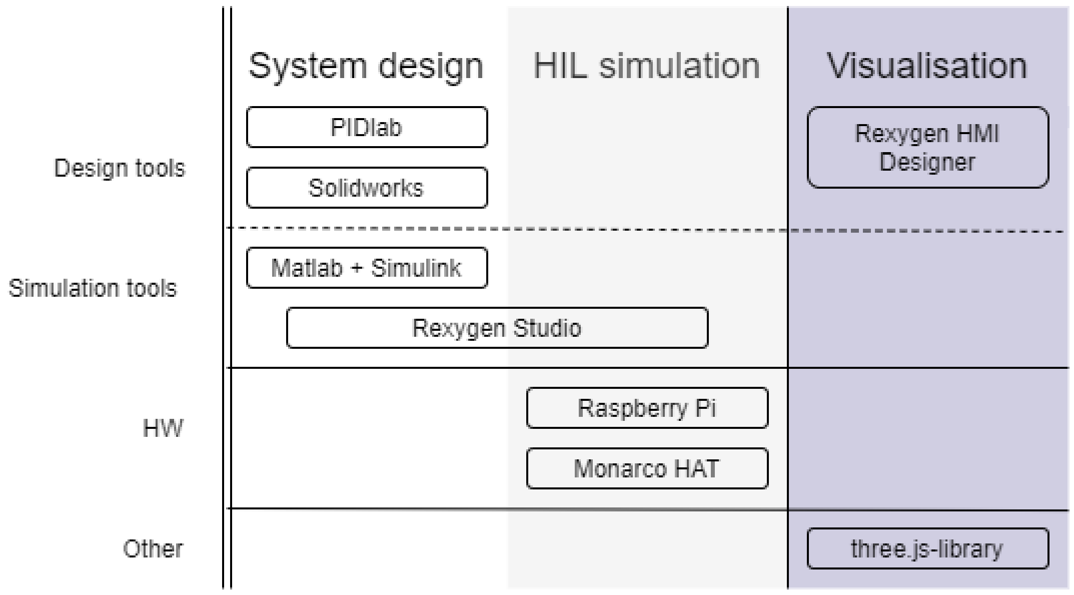


Figure 2.3: *Use of technologies described above*

Chapter 3

Gantry Crane Models

3.1 Simplified crane model

3.1.1 Creation of a basic model and its regulation

To get acquainted with the Rexygen Studio environment, a simple model that would behave similarly to a crane in 2 dimensions was created. The movement of the crane was imitated by a second-order continuous system. The horizontal position of the crane was considered as the output of this system, which served as input for the second-order oscillating model with complex poles that approximated the swinging of the load. Its output was the horizontal position of the load. For the crane model, a PID controller was designed in PIDlab. Once the crane control was working properly, an input-shaping filter was added between the setpoint variable and the PID controller. This filter was set with the parameters of the oscillating system (damping ratio and natural frequency) and managed to negate any oscillation after changes of the crane's position. Naturally, this system was not behaving like a crane but it served its purpose of getting familiar with PID tuning [16] and input-shaping within the Rexygen system.

This early attempt is captured in the figures 3.1, 3.2, 3.3 and 3.4. A switch was used for the PID setpoint control with an option to avoid using the shaping filter.

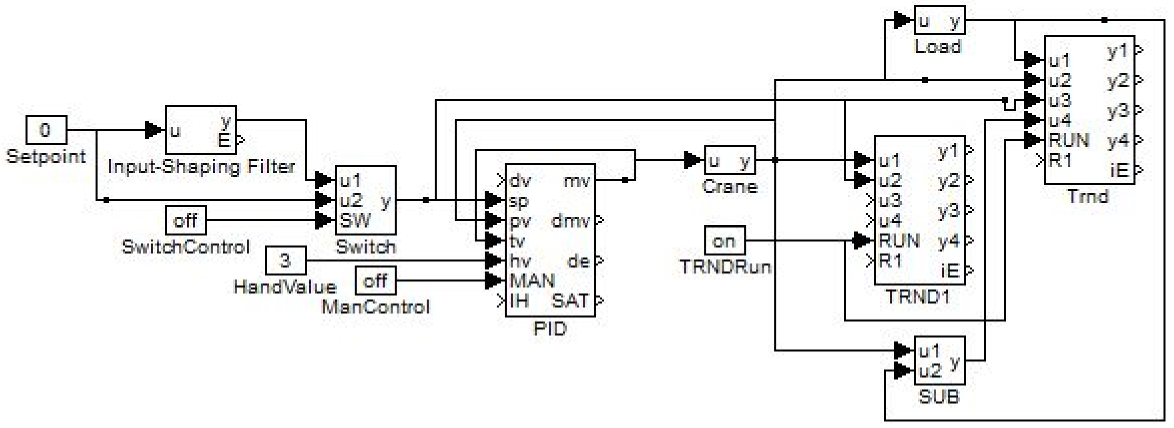


Figure 3.1: *Simple model control scheme with input-shaping filter*

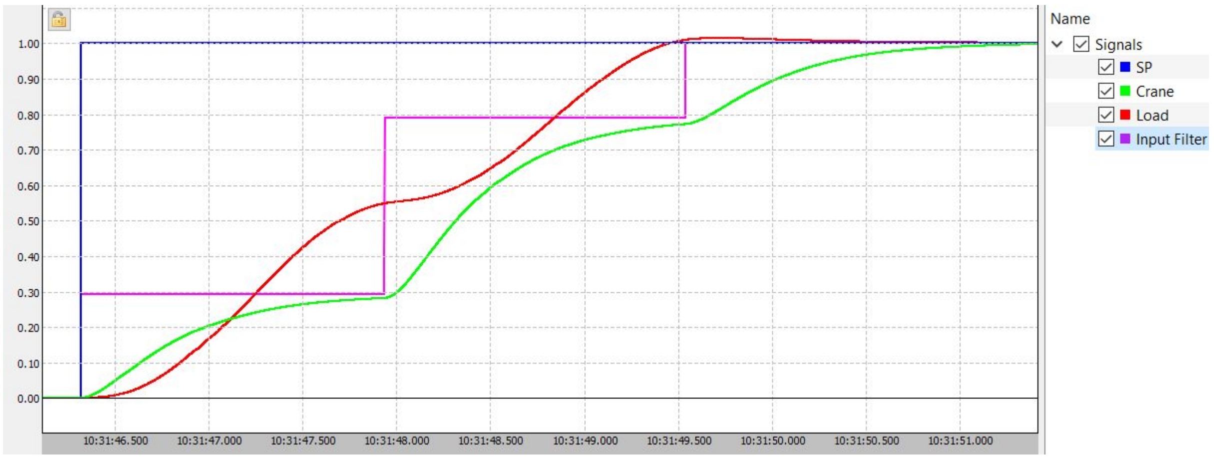


Figure 3.2: *Step response with input-shaping filter (purple)*

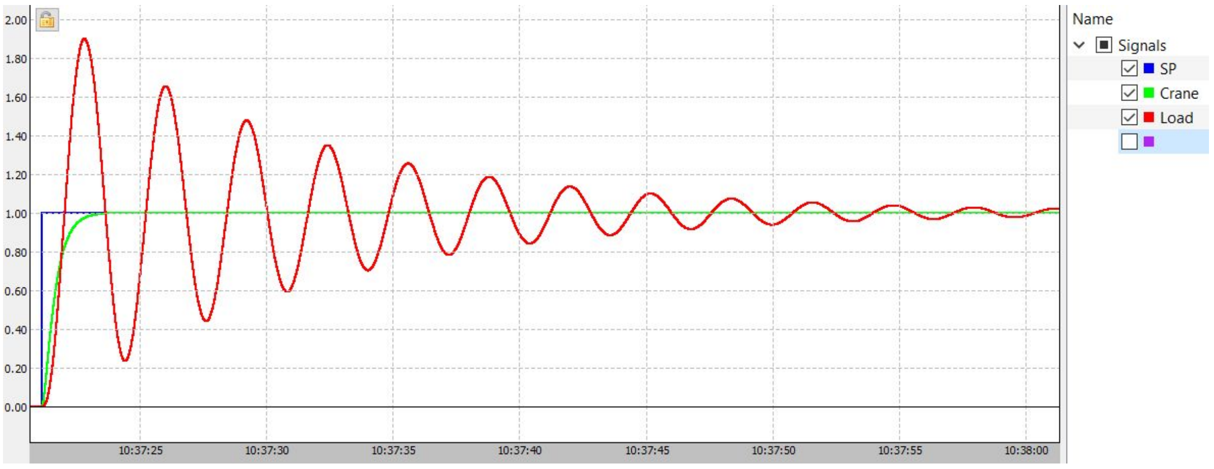


Figure 3.3: *Step response without input-shaping filter*

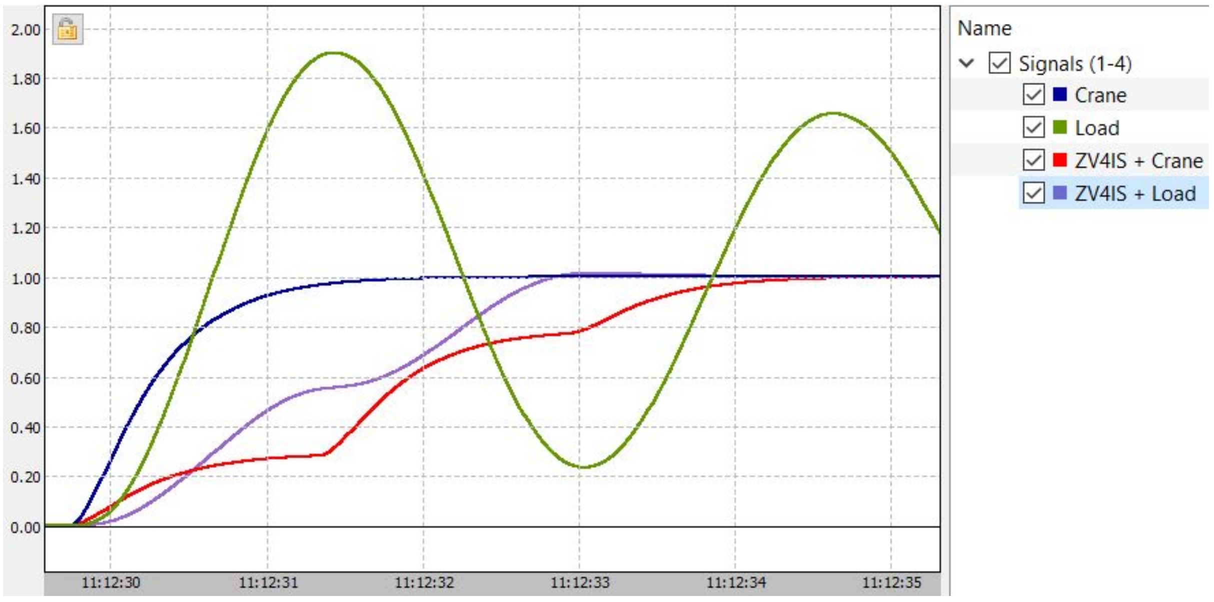


Figure 3.4: *Comparison of simulation results with the same timeframe to show how effective input-shaping can be*

3.1.2 2D Visualisation

This served as a practice example of making a visualisation for Rexygen process in Rexygen HMI designer. The first visualisation is quite simple (Figure 3.5): just two rectangles which pose as a cart and pendulum, a button that switches the input shaping filter on and off and a text box that sets the setpoint for the crane position. Those four features have been connected to their corresponding blocks in the process and in the executing file and an HMI block has been added. This block generates visualisation from a source folder when the simulation is started. The visualisation can be accessed in an internet browser.

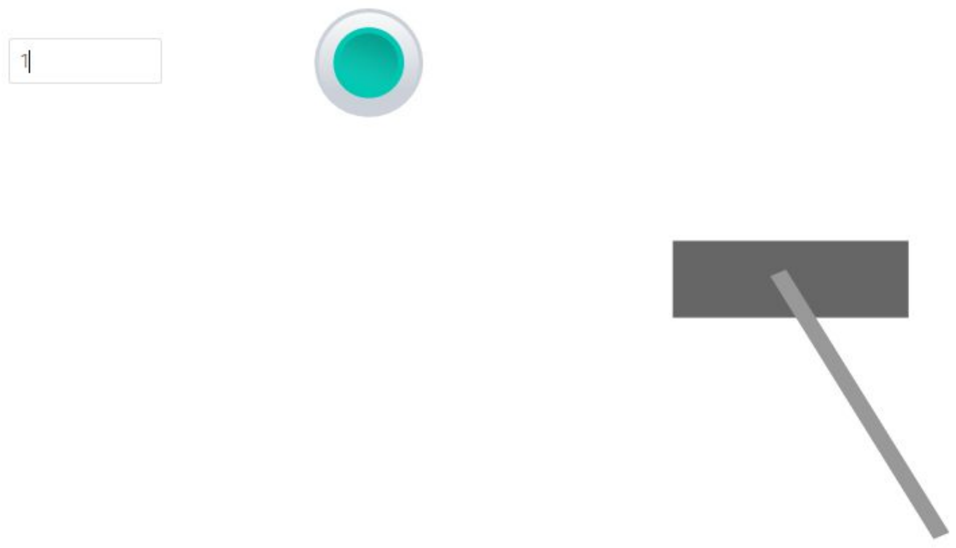


Figure 3.5: *Simple visualisation with setpoint text-box and a switch button for the input-shaping filter*

3.2 Differential Equation-based Crane Model

3.2.1 Introduction

As the last model's behaviour was in some cases obviously different to the one of a crane, a need arose for a more appropriate model. A pendulum on a cart is a suitable model and therefore was used further for the purposes of this thesis.

3.2.2 Non-linear system

A non-linear model of a pendulum on a cart has been derived as a simplified model of the crane. This model consists of two second-order differential equations. In comparison with many pendulum-on-a-cart models that can be found in most of control theory tutorials (like [5]), this one also takes the rotational resistance of the pendulum joint into consideration:

$$(M+m)\cdot x'' + b\cdot x' - b_r\cdot\theta' + m\cdot l\cdot\theta''\cdot cos(\theta) - m\cdot l\cdot\theta'^2\cdot sin(\theta) = F\ ,$$
$$(J+m\cdot l)^2\cdot\theta'' + m\cdot g\cdot l\cdot sin(\theta)\ =-m\cdot l\cdot x''\cdot cos(\theta)\,,$$

Where *M* is the mass of the cart, m is the mass of the pendulum’s load, *b* represents the friction of the cart, b_r represents the friction of the pendulum joint, *F* is the force applied to the cart, *J* is the moment of inertia of the pendulum, *x* is the lateral position of the cart (where *x'* and *x''* are its time derivatives) and *Θ* is the angle between the pendulum and the vertical axis (and *Θ*’ and *Θ*’’ are its time derivatives).

From these equations, a form that expresses the highest derivation order of variables was derived.

$$x'' = \frac{F - b\cdot x' - m\cdot l\cdot\theta''\cdot cos(\theta) + m\cdot l\cdot\theta'^2\cdot sin(\theta)}{(M+m)}$$

$$\theta'' = \frac{-x''\cdot cos(\theta) - g\cdot sin(\theta) - b_r\cdot\theta'}{l}$$

This model has been simulated both in Simulink and Rexygen. In Simulink, each equation is written in a *Fcn* block whose output leads to a set of two integrator blocks in a row. The output of each block, along with the Force signal, is multiplexed into a signal that serves as an input signal for the Fcn blocks (Figure 3.6). In Rexygen, a similar approach has been utilised: Rexlang blocks have been used for the equations which were written in C-like code. Since there is no multiplexor block available in Rexygen, outputs of the Rexlang blocks and of the integrators lead back to the Rexlang blocks individually (Figure 3.7).

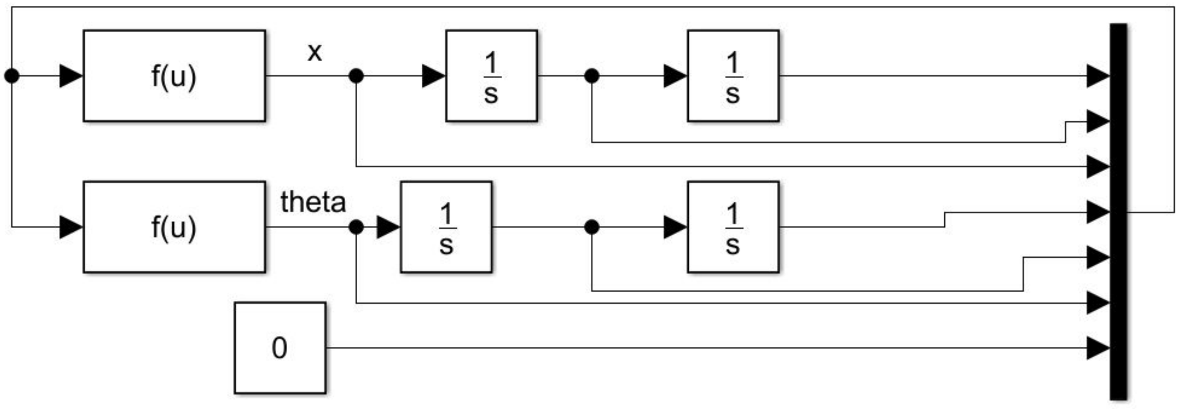


Figure 3.6: *Non-linear model in Simulink*

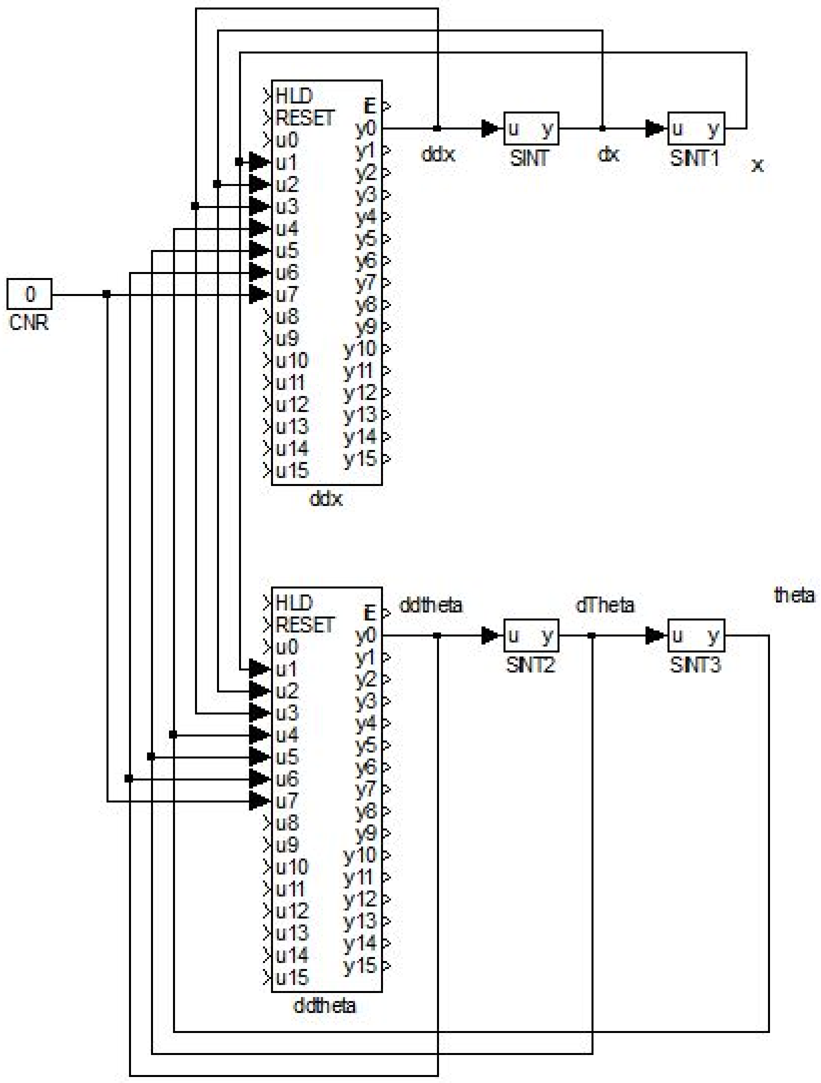


Figure 3.7: *Non-linear model in Rexygen*

3.2.3 State-space model

From the non-linear model, a state-space model has been derived. The linearisation point was selected to be the steady-state point, meaning the pendulum is hanging straight down and is still and the crane is still as well. The crane's lateral position has no effect on the stability of the system and was therefore chosen to be zero.

The state-space model has been defined by the following matrices:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -b/M & mg/M & mbr/M \\ 0 & 0 & 0 & 1 \\ 0 & b/(lM) & -(M+m)g/(lM) & -(M+m)br/(lM) \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 1/M \\ 0 \\ -1/(lM) \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Both in Simulink and Rexygen, these matrices have been pasted in the state-space blocks in basic libraries. This model has one input – the force – and four outputs – the position and the speed of the crane, and the angle and angular speed of the pendulum.

3.2.4 State-space model of a double integrator

Since the process simulated in Rexygen slightly lagged behind its Simulink counterpart, attempts were made to identify the source of this discrepancy. One of the suspected reasons was that the set of two subsequent integrators caused inconsistency as the second derivatives would take more than one computational period to go through both integrators. Therefore, a state-space block was created with the functionality of a set of two integrator blocks (Figure 3.8). The comparison of this solution with the other models both in Rexygen and Simulink environments can be seen in section 3.2.5.

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

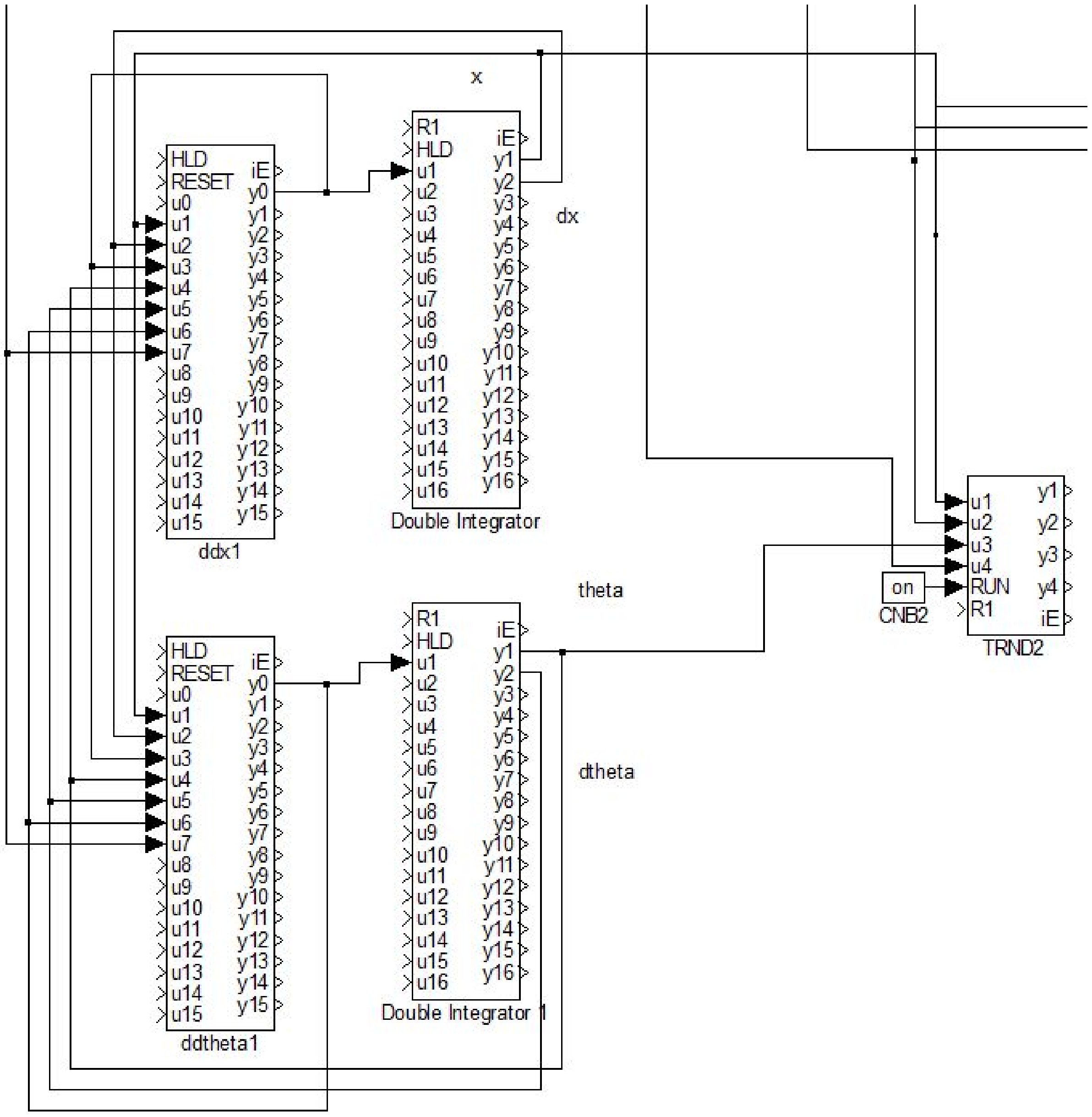


Figure 3.8: *State-space model of double integrator used to optimize real-time simulation*

3.2.5 Comparison of created models

All of the above mentioned models have been compared and the results can be found in this section. There is an example figure of each comparison focusing on one of the output variables. These variables are: crane position, crane speed, pendulum position, pendulum angular velocity.

First, the Simulink models are compared (Figure 3.9). The difference (Figure 3.11) between the models in this well-controlled environment with initial conditions close to the linearization point is negligible.

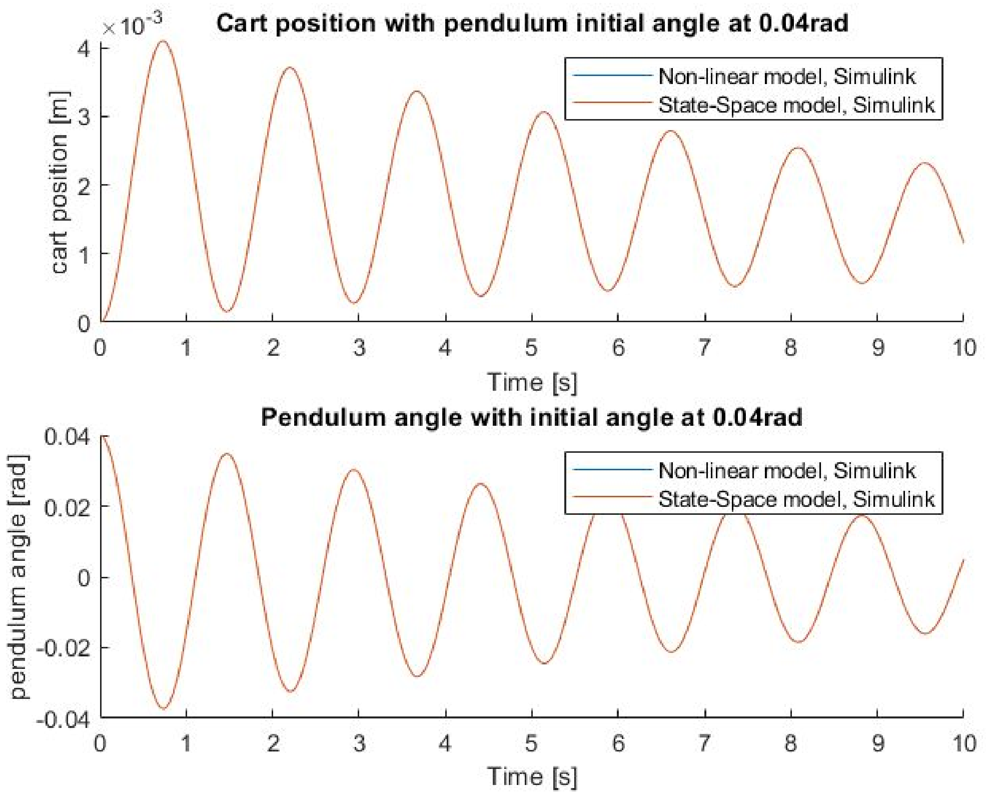


Figure 3.9: *Plot of model variables from Simulink. These models behave very similarly with only negligible differences between them.*

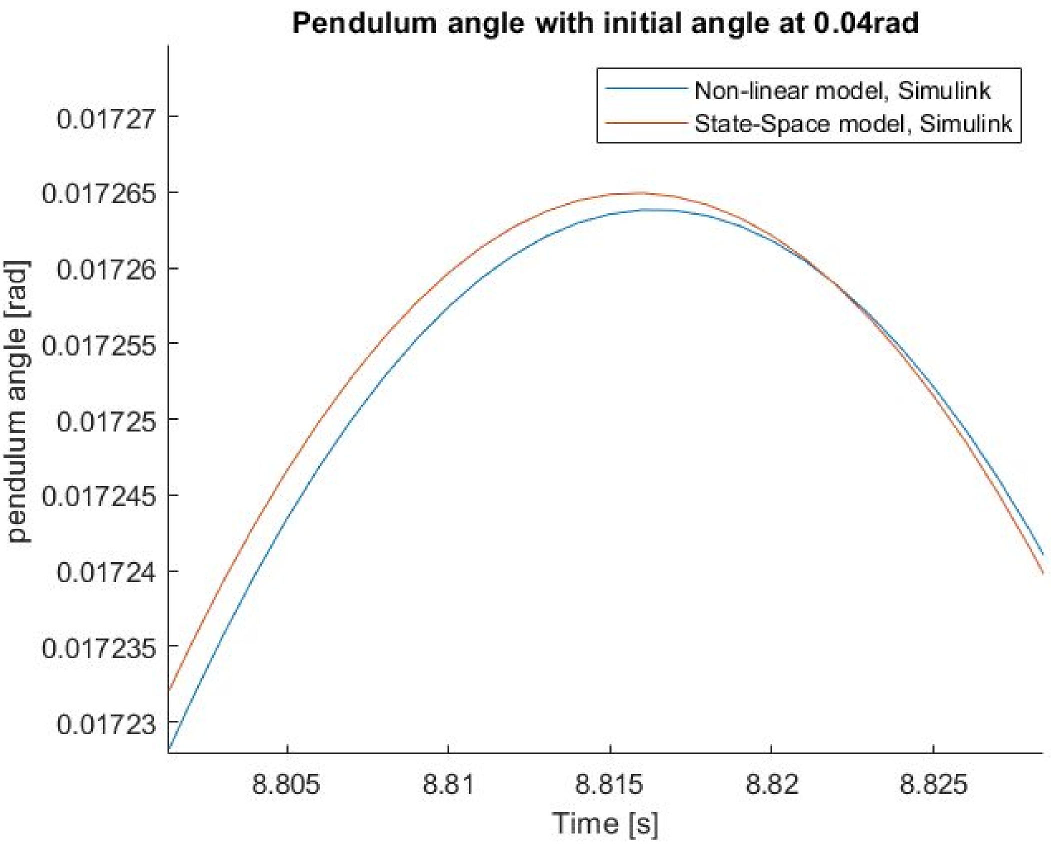


Figure 3.10: *Detail of the simulation result plotted in* Figure 3.9

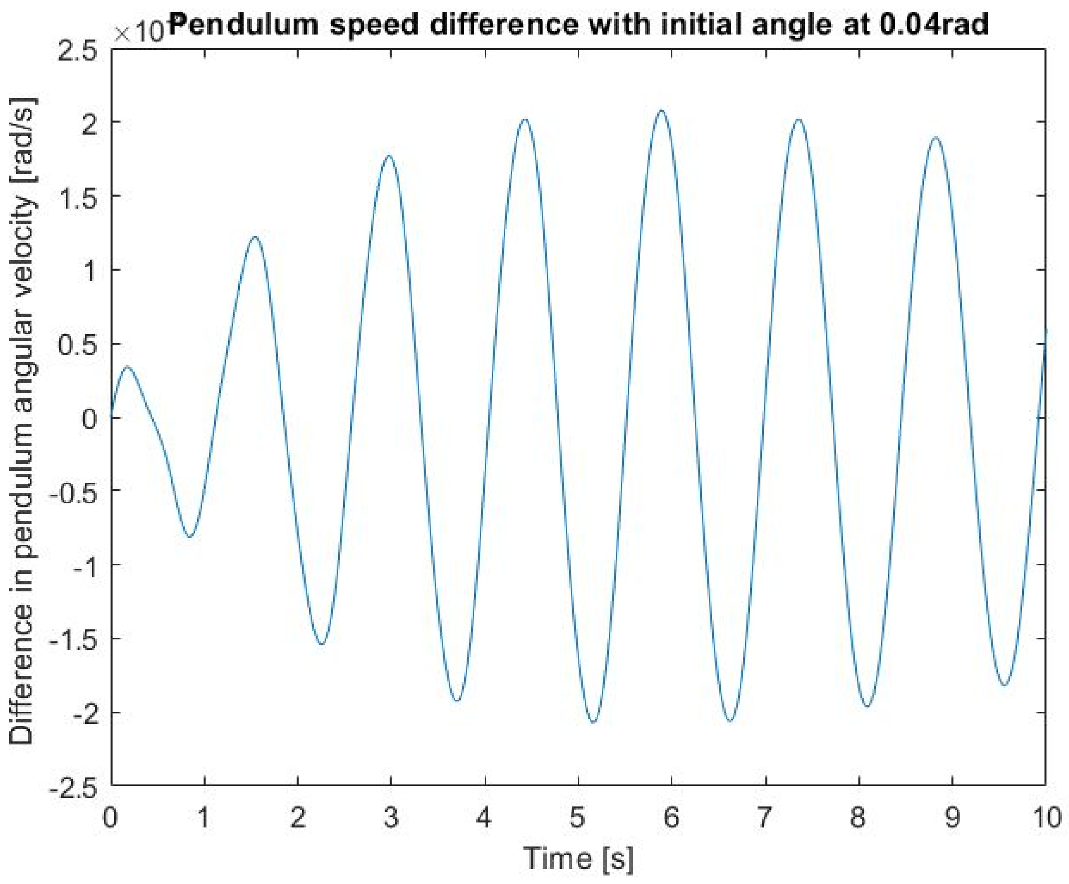


Figure 3.11: *The graph describes time dependent difference between the models. Its amplitude grows initially due to numerical differences until damping suppresses both of them as seen on the right half of the plot.*

The same has been done for the Rexygen models (Figure 3.12). Here the difference between the non-linear and state-space models is more noticeable (Figure 3.14), especially in the later stage of the simulation.

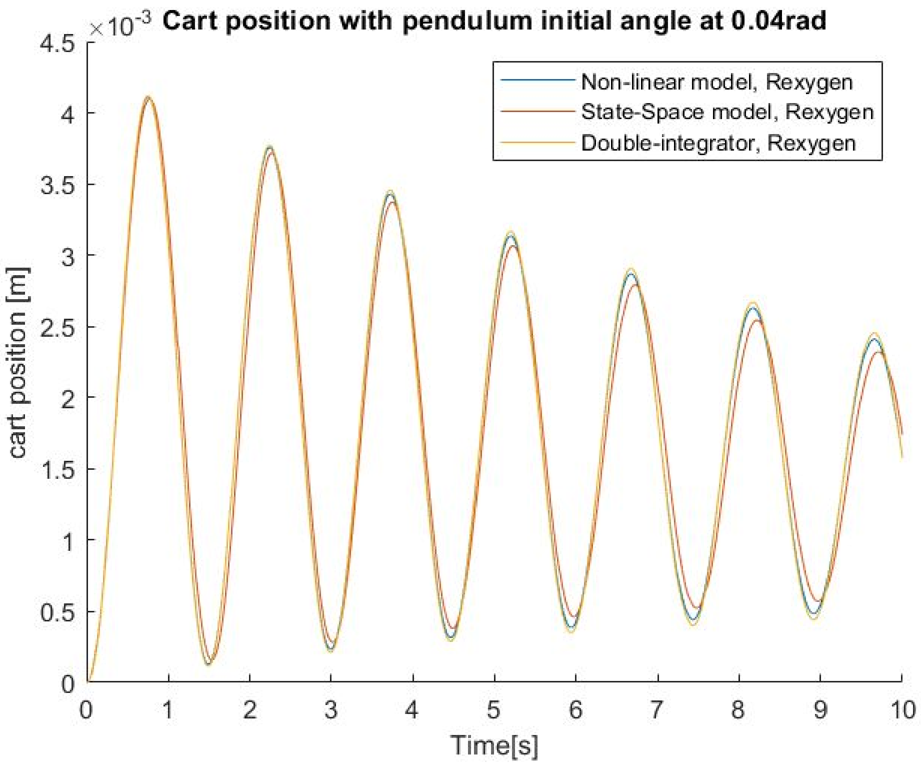


Figure 3.12: *Plot of simulations made in Rexygen. Unlike in Simulink, differences between models are noticeable and grow with time.*

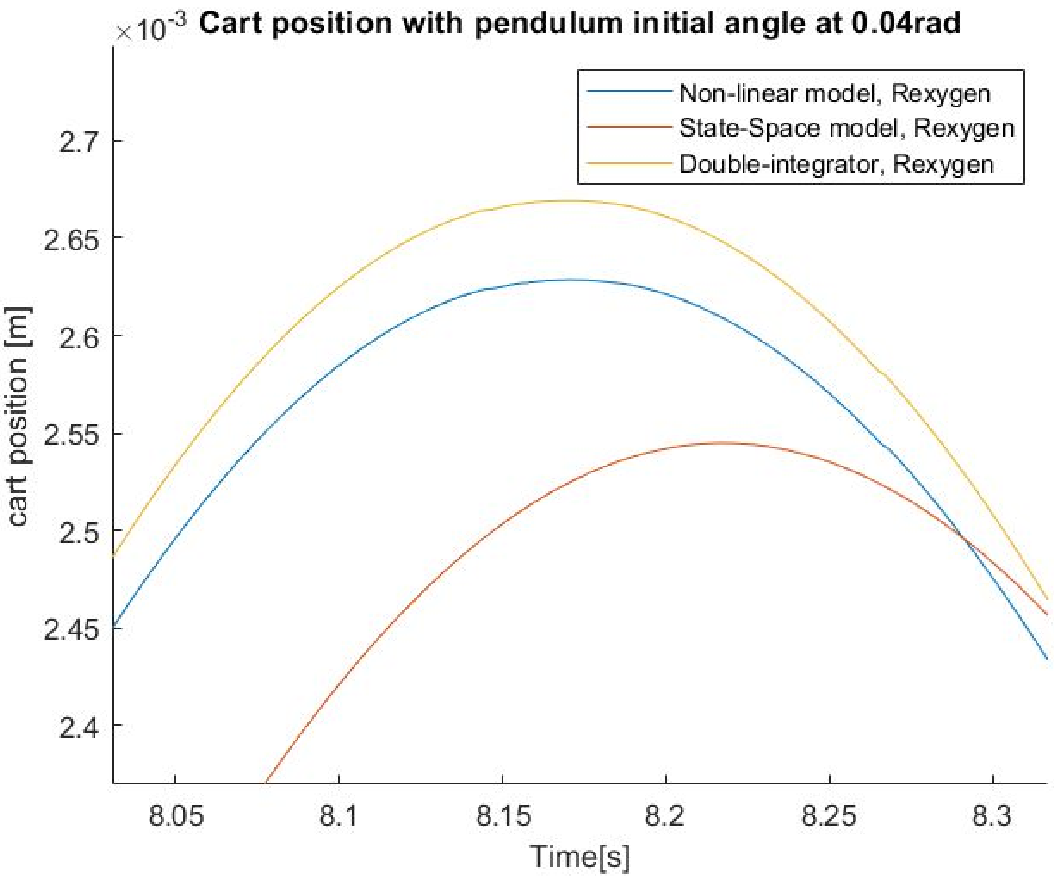


Figure 3.13: *Detail of simulation results from* Figure 3.12

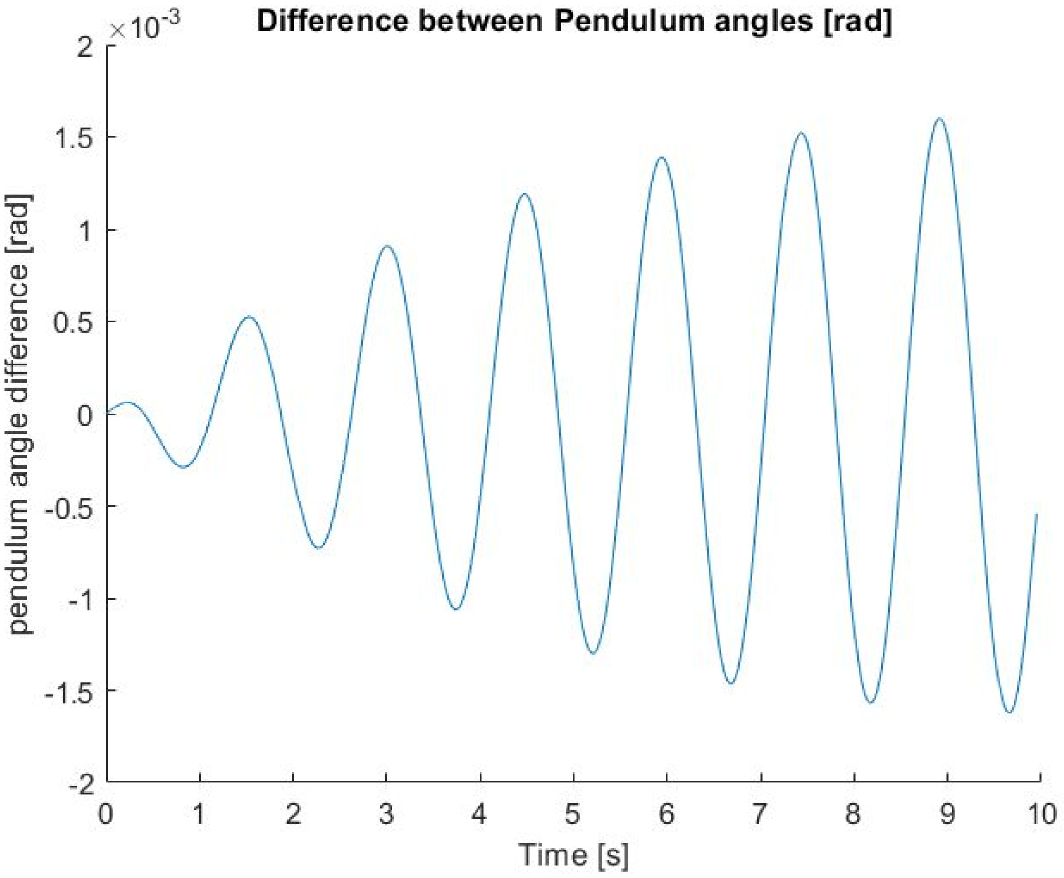


Figure 3.14: *Difference between State-space and Non-linear models in Rexygen*

At this point, it makes sense to compare equivalent models from Rexygen and Simulink. Firstly, the non-linear (Figure 3.15) and then the state-space ones (Figure 3.17). The Rexygen models are lagging behind the Simulink models in both cases. The difference between the non-linear models gets greater over time both in lag and amplitudes while for state-space models the amplitude difference is consistently negligible.

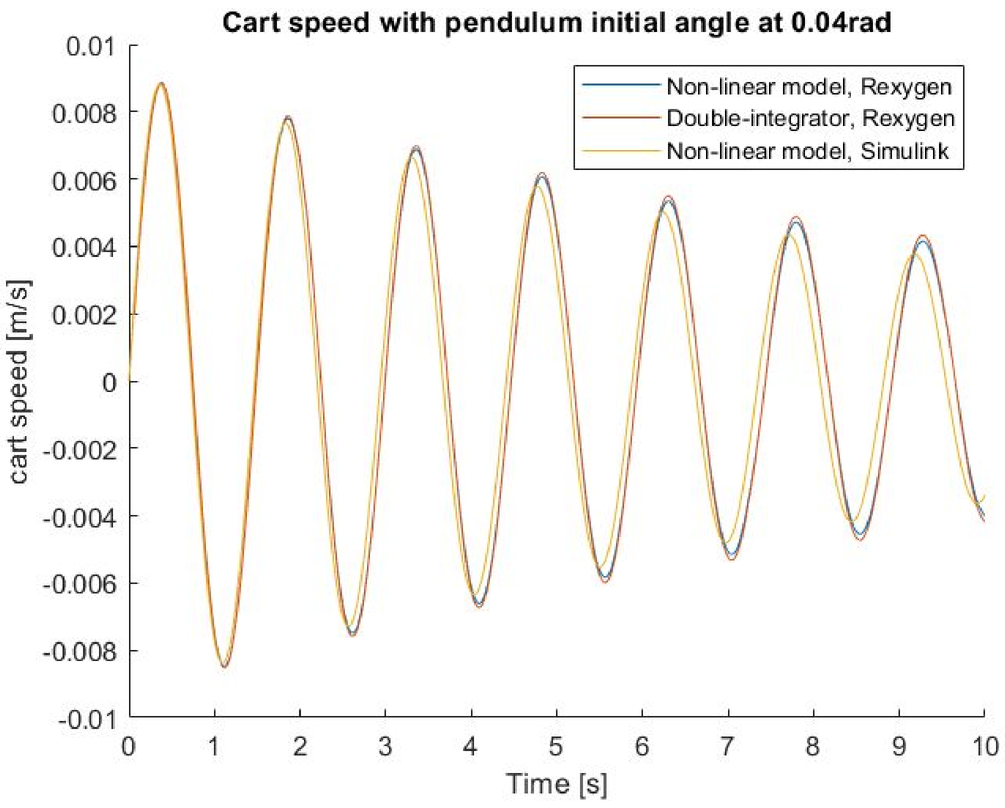


Figure 3.15: *Comparison of Simulink and Rexygen simulation results for non-linear model simulations*

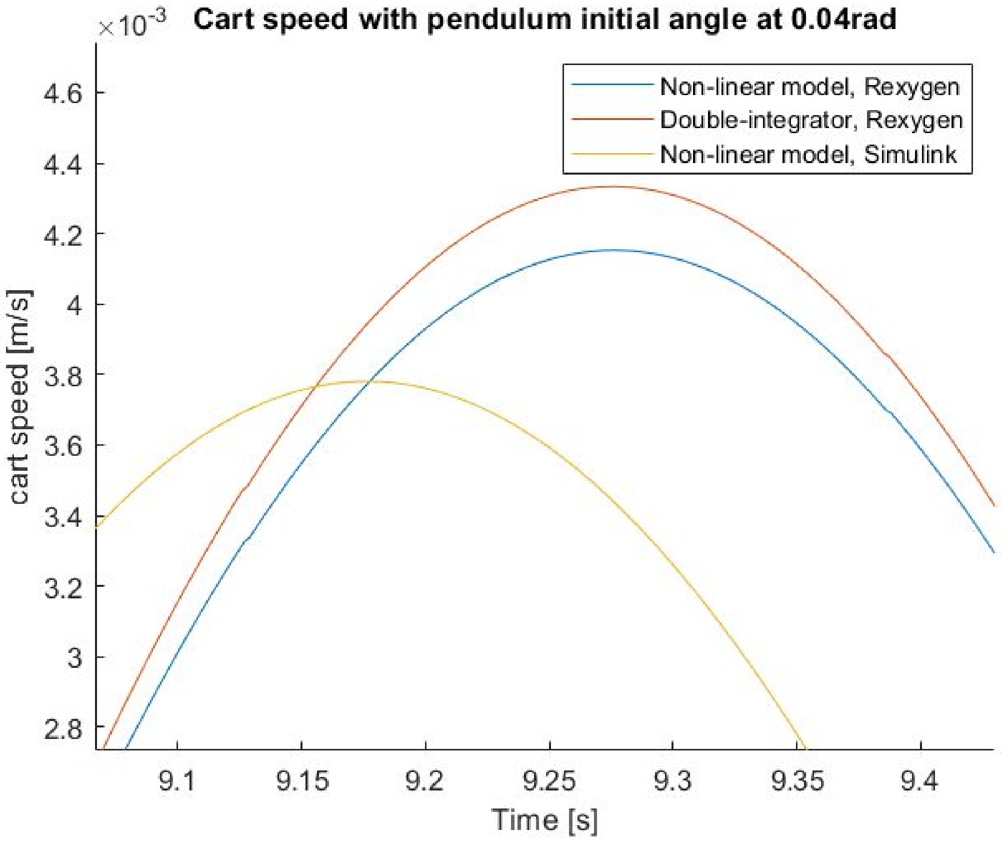


Figure 3.16: *Detail of plot from* Figure 3.15

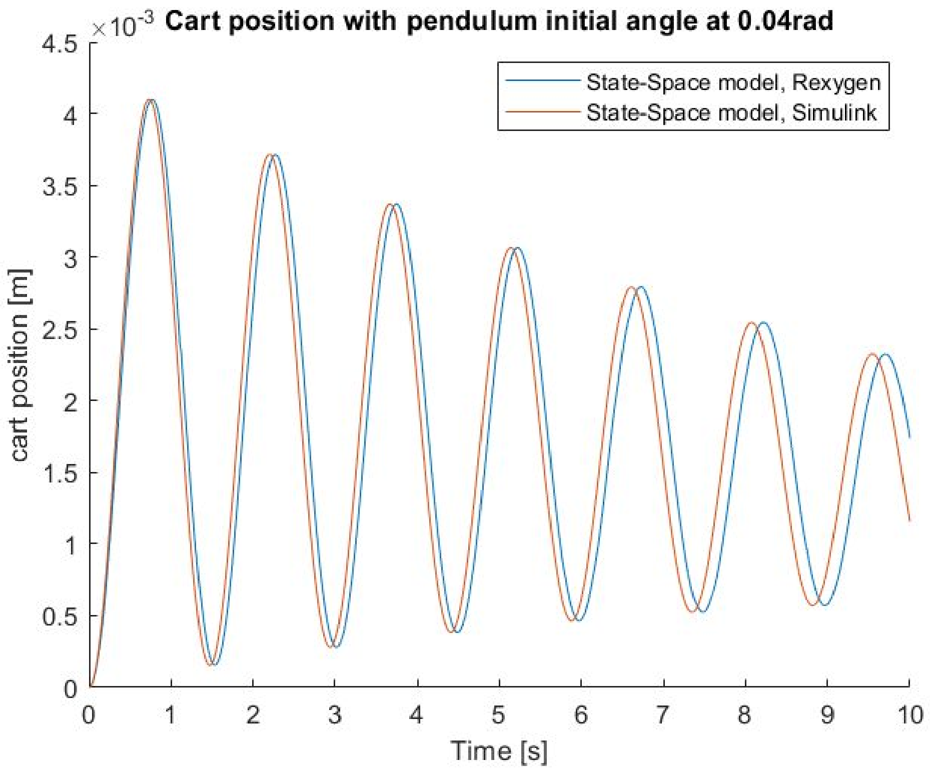


Figure 3.17: *Comparison of Simulink and Rexygen simulation results for State-space model simulations*

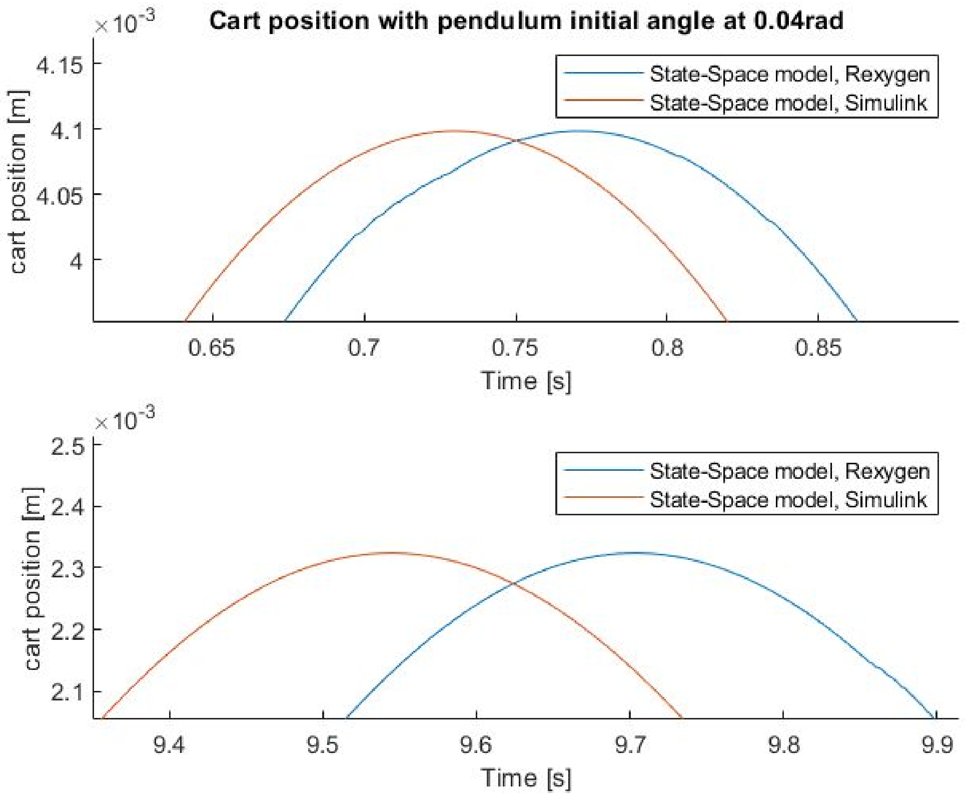


Figure 3.18: *Detail of* Figure 3.17 *plot at early and later stages of simulation, showing increasing phase difference over time*

Lastly, differences between Simulink models have been evaluated with various initial conditions (Figure 3.19). As expected, the further the initial conditions were from the linearization point, the greater the absolute difference was between the non-linear and State-space models, but relative difference is indistinguishable among various initial conditions (Figure 3.20).

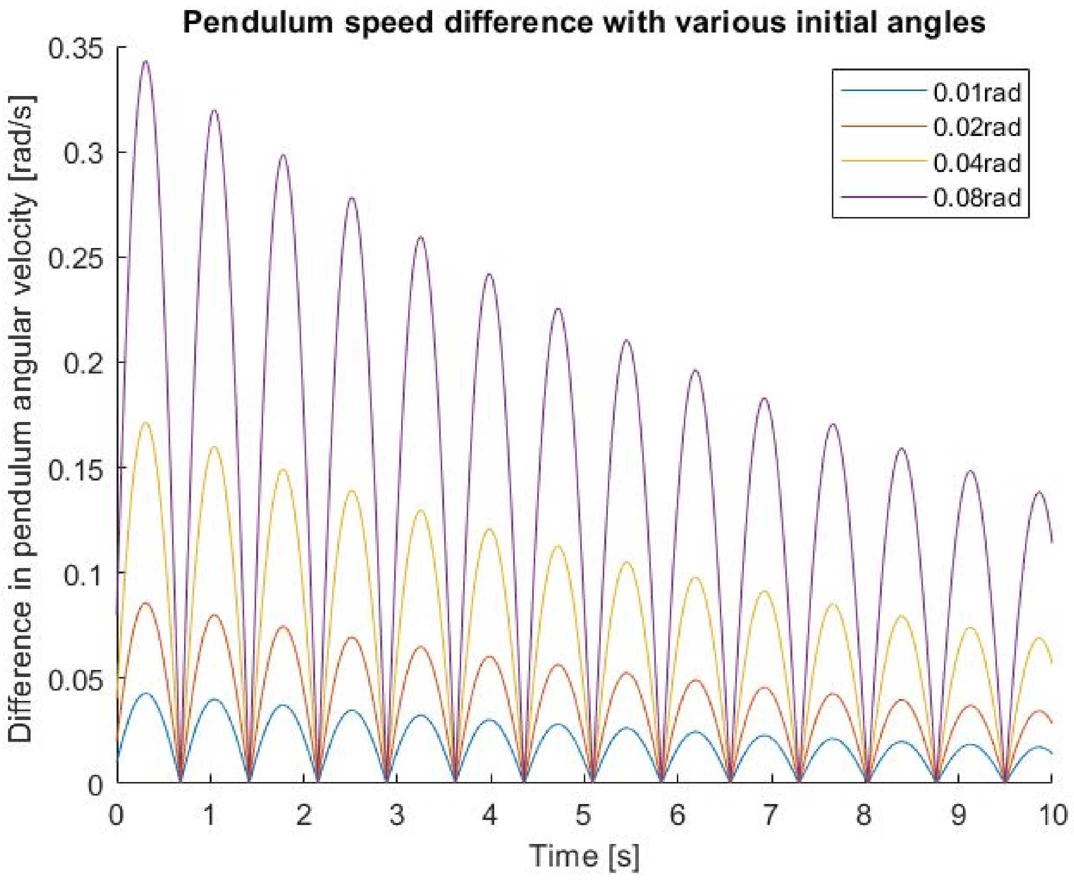


Figure 3.19: *Differences between non-linear and state-space models with various initial conditions*

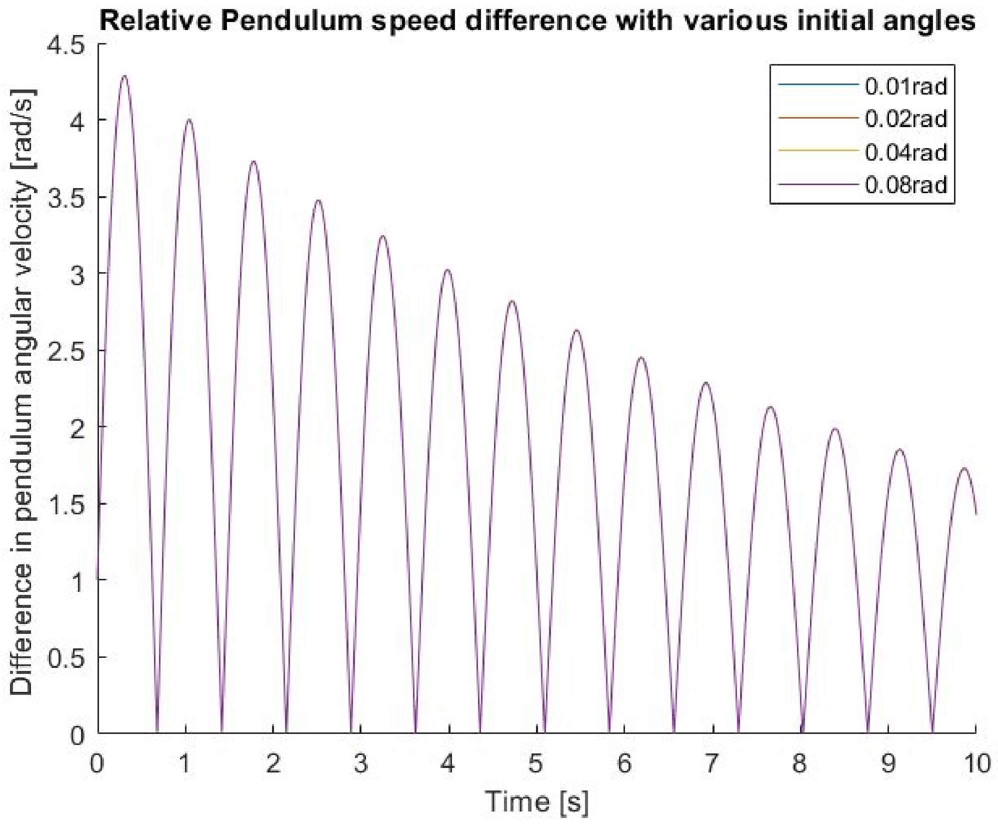


Figure 3.20: *Relative differences with various initial conditions*

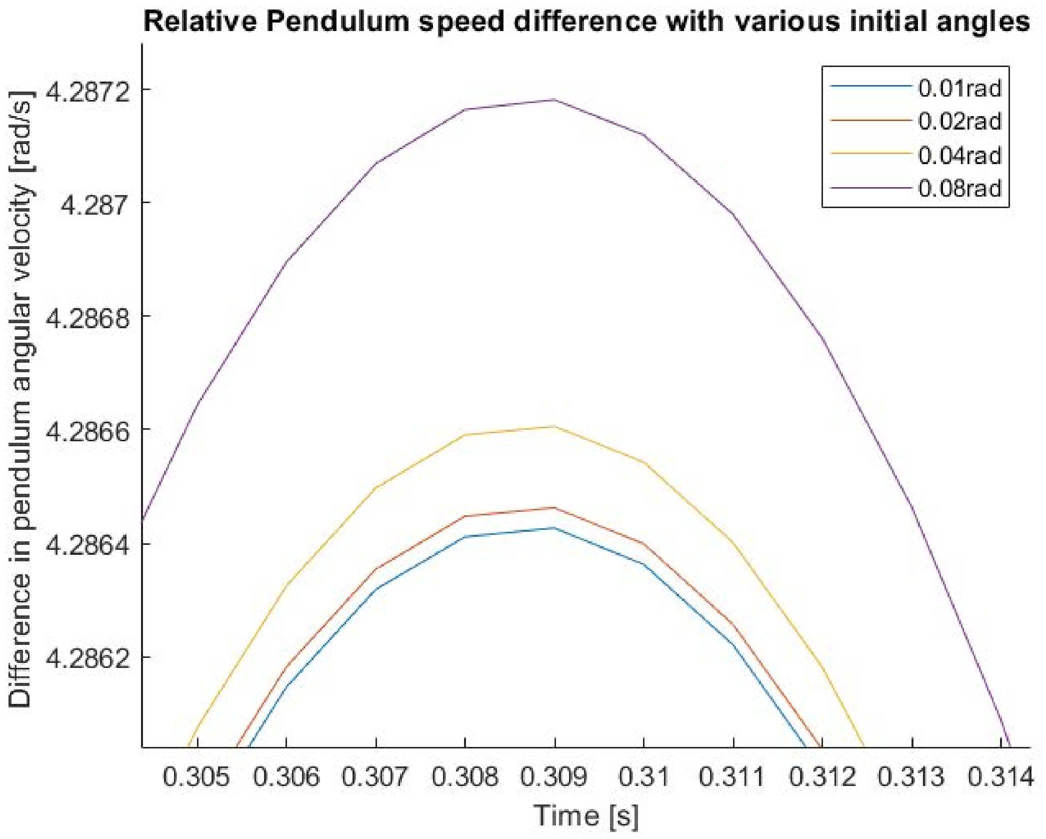


Figure 3.21: *Detail of* Figure 3.20

3.3 Control

3.3.1 PID controller design

A PID regulator was designed using Matlab PID Tuner app. Attempts were made to create a controller in PIDlab but were abandoned after several unsuccessful attempts to design a stable one (PIDlab was unable to draw stabilising regions for large model parameters). The regulator's output controls the force applied to the cart. The output of the model is the cart's position only. The controller was tested in Simulink (Figure 3.22) to prove its ability to control the system and later used in Rexygen to control the pendulum model.

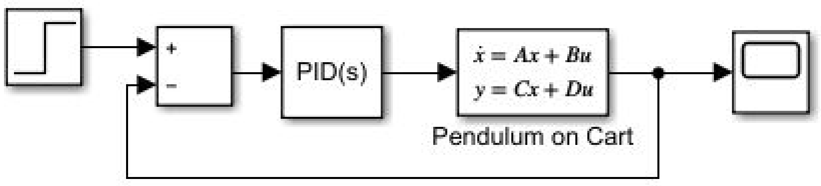


Figure 3.22: *Control scheme for built-in PID tuning*

3.3.2 Applying an input-shaping filter

To cancel swinging (Figure 3.23) of the load at the end of the crane movement, an input shaping filter was needed to be added to the control scheme. In this case, a *Zero vibration input shaper* block in Rexygen has been used [26]. There are two system-related parameters that needed to be set to work properly: the natural frequency of the system and the relative damping coefficient. When the filter is active, upon change of the user-selected set-point, it does not change the set-point for the PID regulator immediately but it does so in several steps (Figure 3.24). It is probably worth noting that the setpoint weighting factor for the derivative part has been set to 0 to lower the initial controller response.

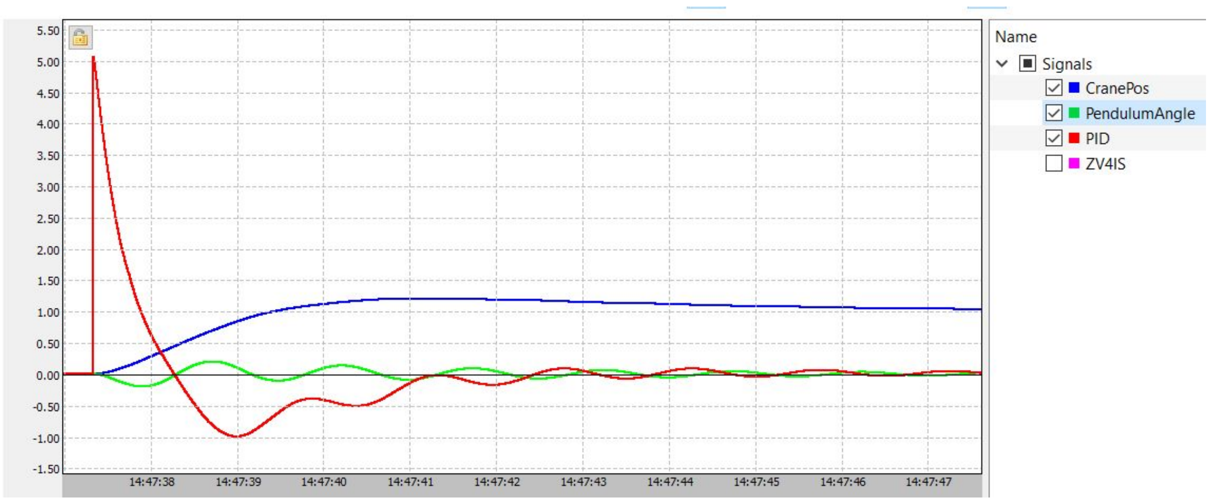


Figure 3.23: *System output without input-shaping filter*

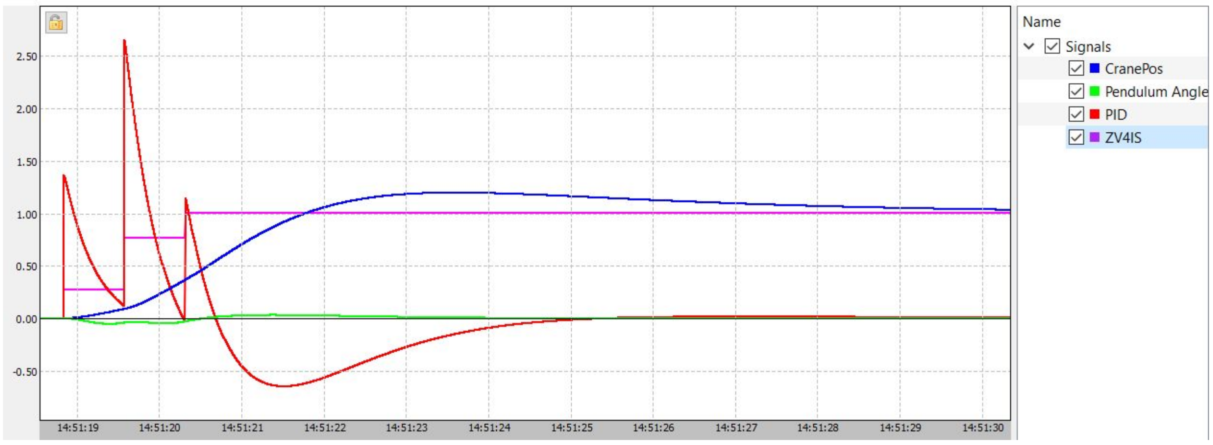


Figure 3.24: *System output with input-shaping filter*

3.4 Software-in-the-loop

The pendulum model and the controller were put in separate files to emulate a real control system where the controller is physically separated from the controlled system (Figure 3.25).

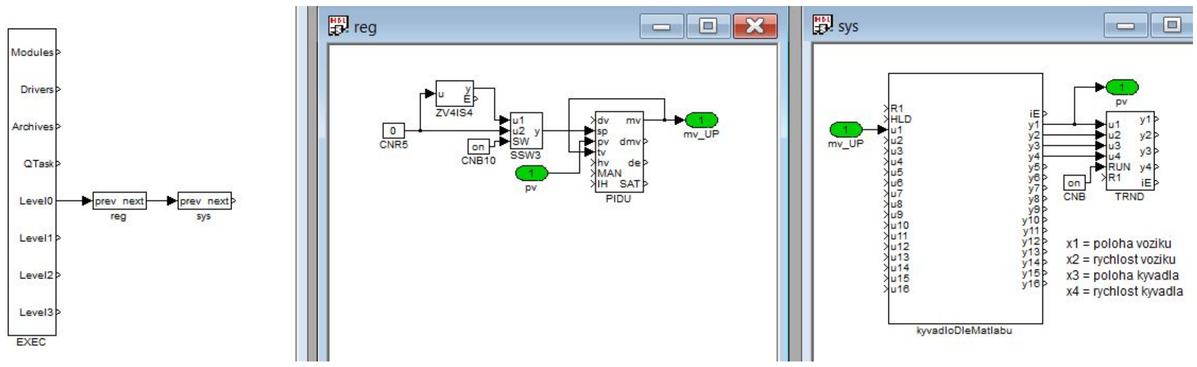


Figure 3.25: *Real-time simulation with the controller separated from the model, communicating via information signal*

3.5 3D Visualisation

3.5.1 Introduction

An important part of any HIL-based simulation is visualisation. Since this model is not based on a specific crane, any gantry crane model is usable. A model [3] for a 3D printed gantry crane has been chosen (Figure 3.26). Out of ten parts from the model, only 4 were used in the visualisation: the vertical support frame (Figure 3.27), the bridge girder, the trolley and wheels. Some of those models were customised to better fit the visualisation needs (instead of the original purpose to be a toy) and a simple cuboid was created to serve as a container. After these individual models were adjusted, they were put together to form an assembly (Figure 3.28). Based on the assembly, a rail model was made to fit the wheels and their spacing and added to the assembly as well. All of these steps have been taken using the Solidworks CAD environment. Adding minor changes (the rail model has been changed and a rope model has been added), the model was converted into a Rexygen HMI visualisation. This 3D visualisation uses Three.js rendering framework and has been tested by changing input constants (Figure 3.29) to make sure that everything works correctly. This visualisation operates in two modes: the first one has a fixed viewpoint in space; the other has a viewpoint fixed to the crane, so when the cran moves the viewpoint follows. The view (Figure 3.30) can be rotated, panned and zoomed.

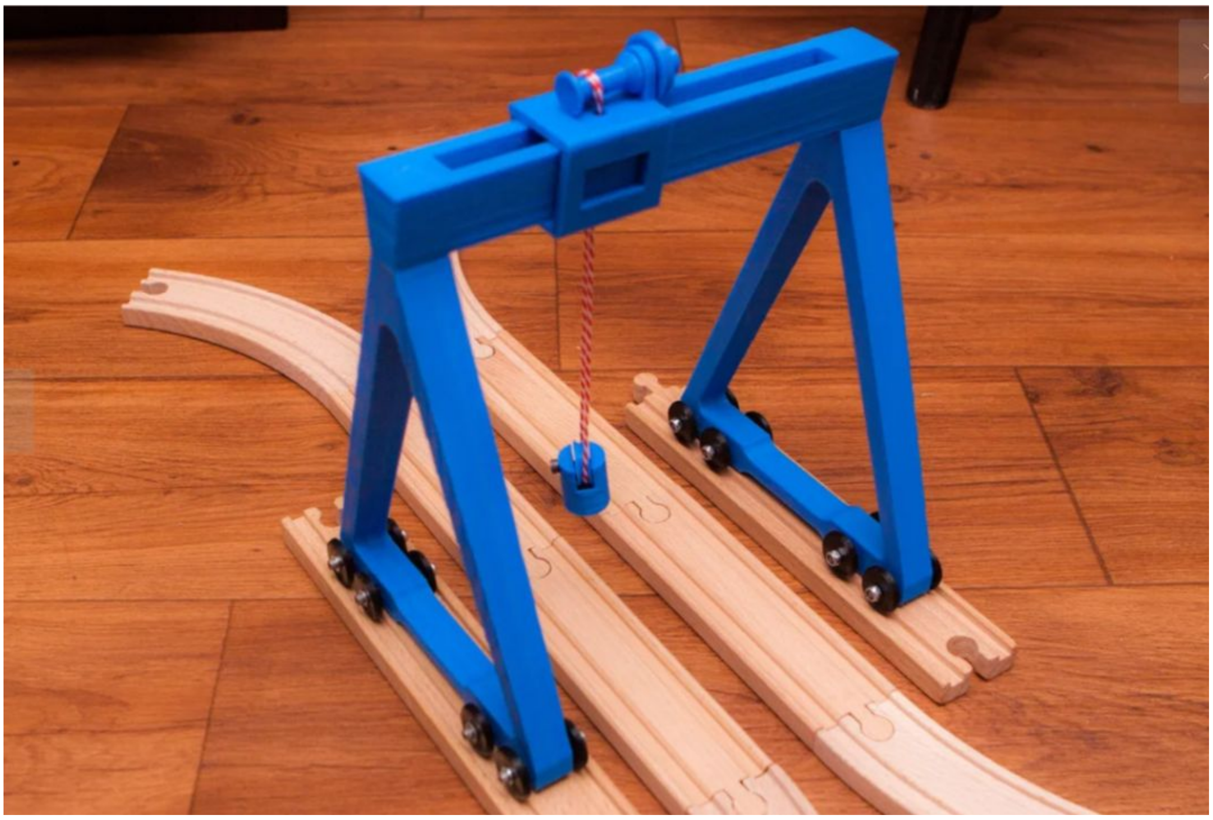


Figure 3.26: *3D printed crane model that the visualisation is based on*

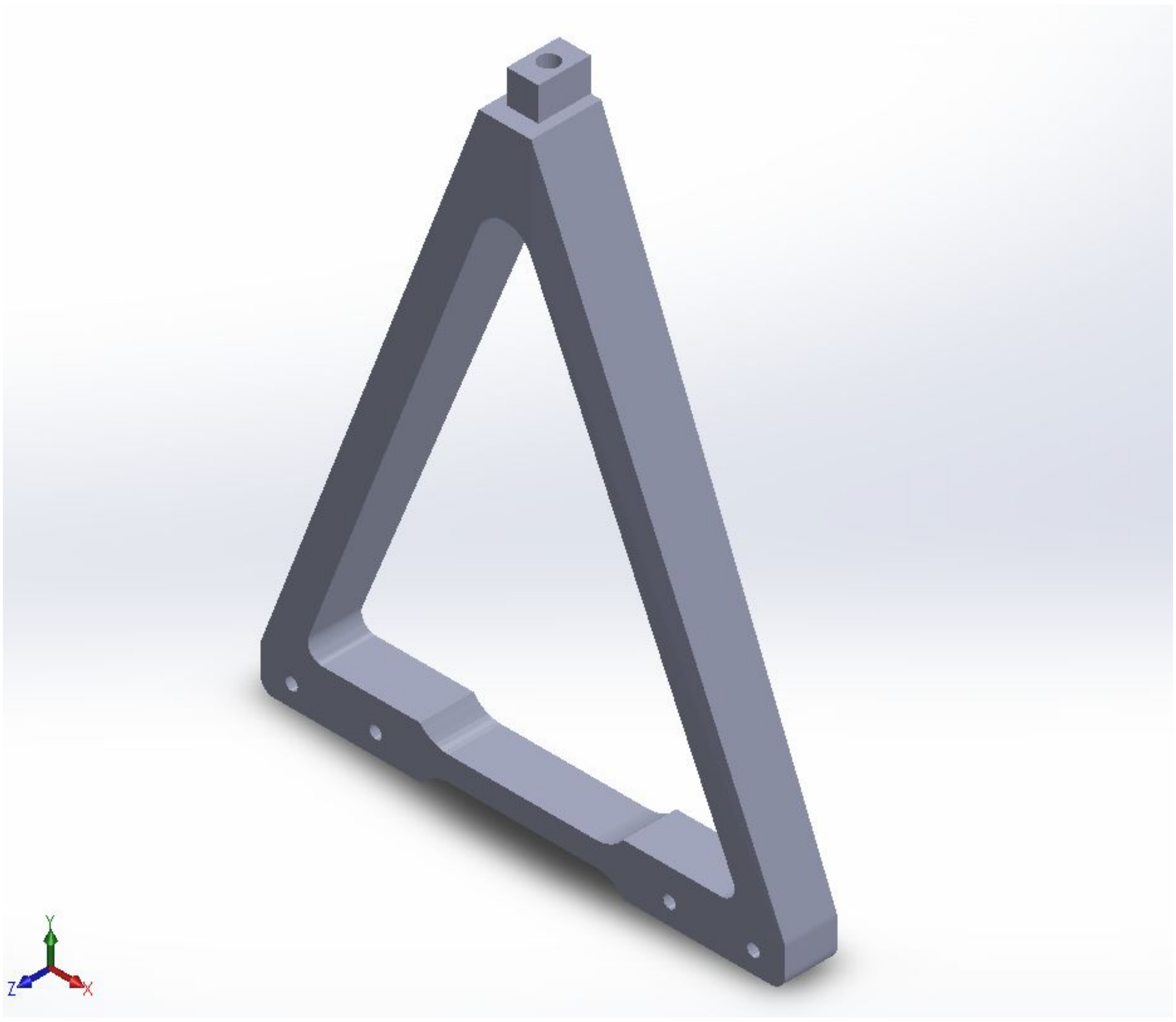


Figure 3.27: *Single part view in Solidworks*

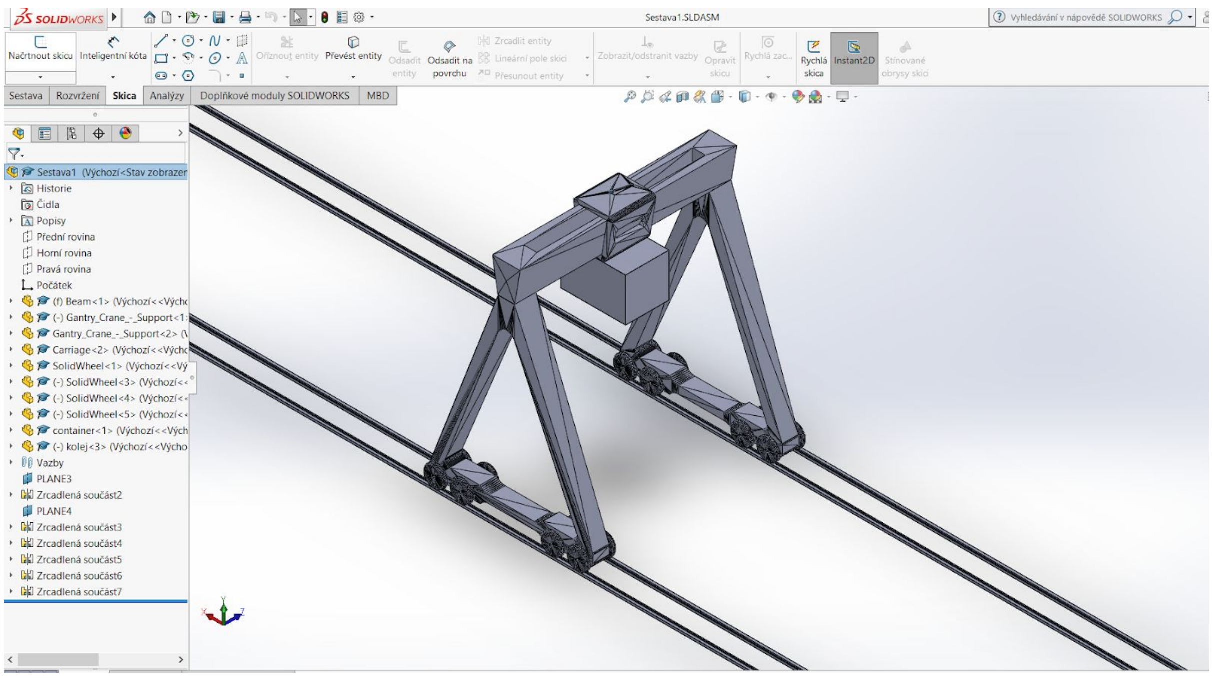


Figure 3.28: *Parts assembly in Solidworks*

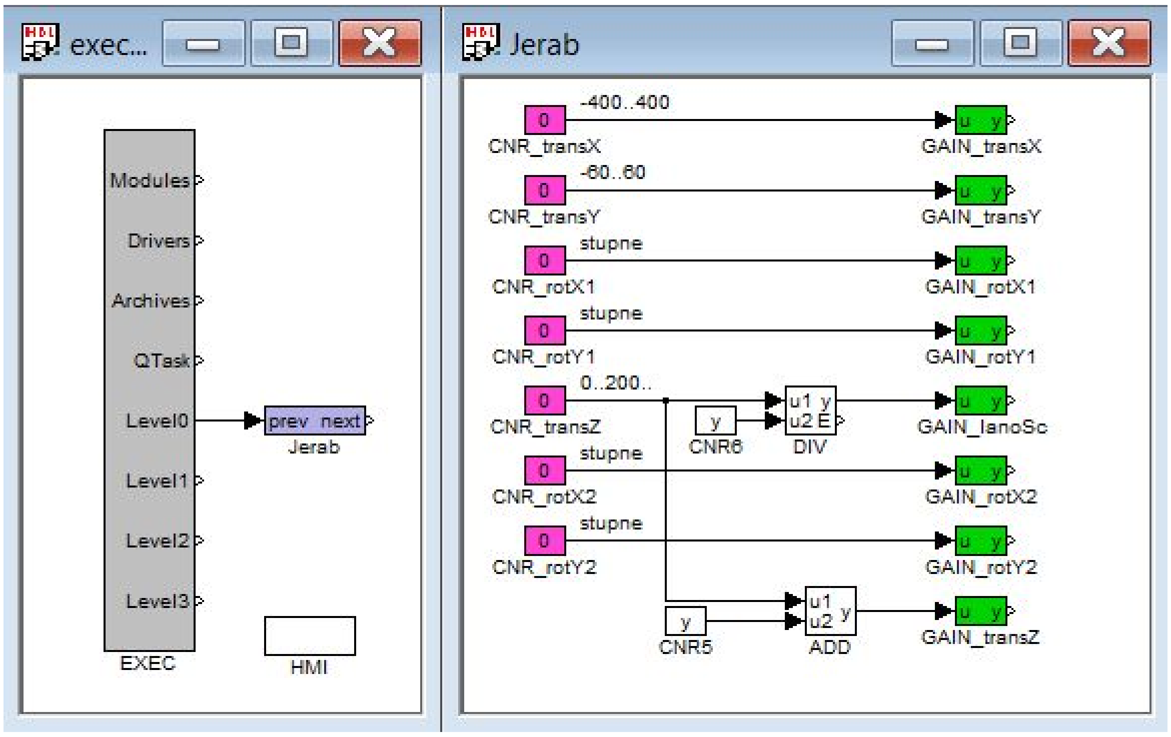


Figure 3.29: *Control variables for the visualisation*

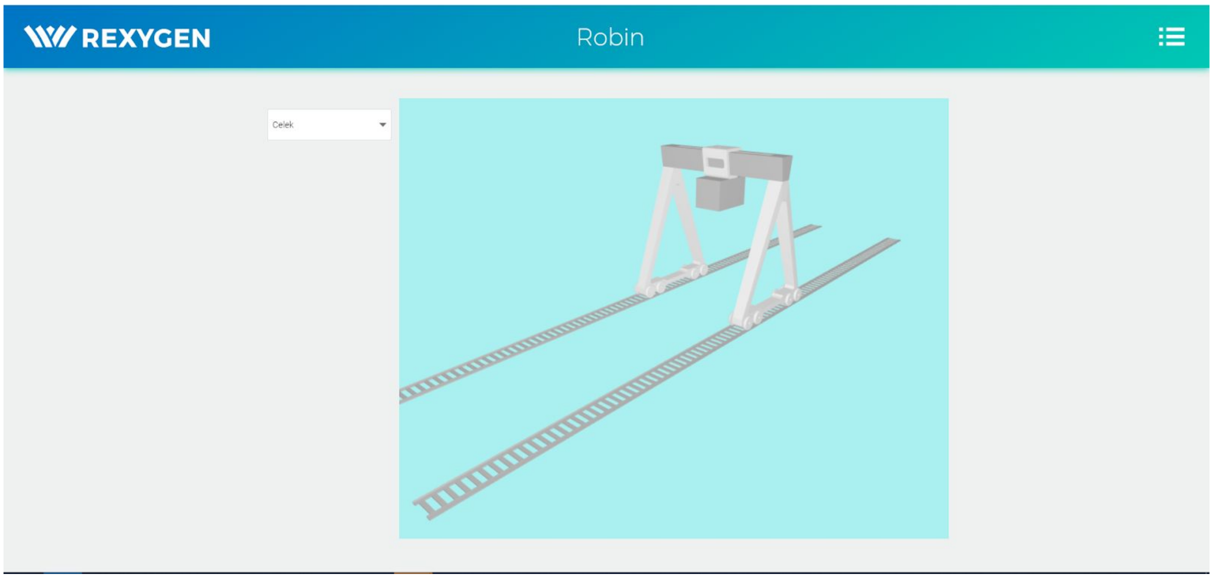
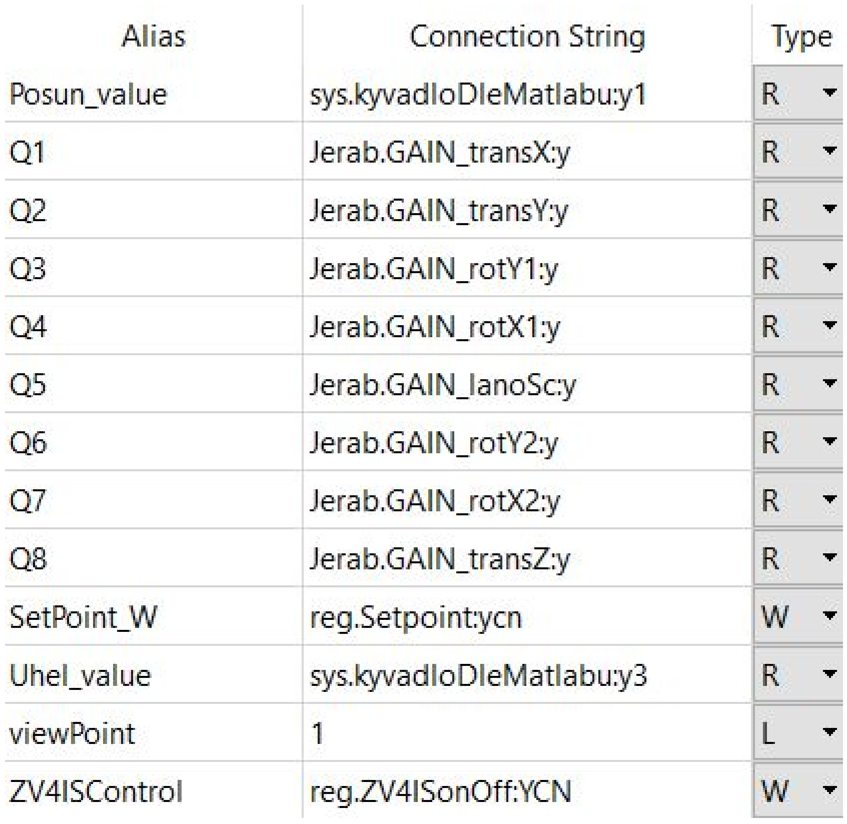


Figure 3.30: *Basic 3D visualisation in the internet browser*

3.5.2 Connection to the model

To make the visualisation move according to the changes in the process values, it was necessary to set up a proper configuration. This consisted of two parts: separating those variables into a separate task in Rexygen Studio; then connecting the visualisation to the separate task in Rexygen HMI Designer (Figure 3.31).



Alias	Connection String	Type
Posun_value	sys.kyvadloDleMatlabu:y1	R
Q1	Jerab.GAIN_transX:y	R
Q2	Jerab.GAIN_transY:y	R
Q3	Jerab.GAIN_rotY1:y	R
Q4	Jerab.GAIN_rotX1:y	R
Q5	Jerab.GAIN_lanoSc:y	R
Q6	Jerab.GAIN_rotY2:y	R
Q7	Jerab.GAIN_rotX2:y	R
Q8	Jerab.GAIN_transZ:y	R
SetPoint_W	reg.Setpoint:ycn	W
Uhel_value	sys.kyvadloDleMatlabu:y3	R
viewPoint	1	L
ZV4ISControl	reg.ZV4ISonOff:YCN	W

Figure 3.31: *HMI elements connected to process variables*

3.5.3 Adding control elements

Additional elements had to be included in the visualisation, both input and output ones. The inputs consist of a setpoint textbox (which works with numbers only, obviously) and a button to turn on and off the input shaping filter. The output display shows the values of the crane position and the pendulum angle variables, with 4 decimals (Figure 3.32). Figure 3.33 shows the canvas in Rexygen HMI designer.

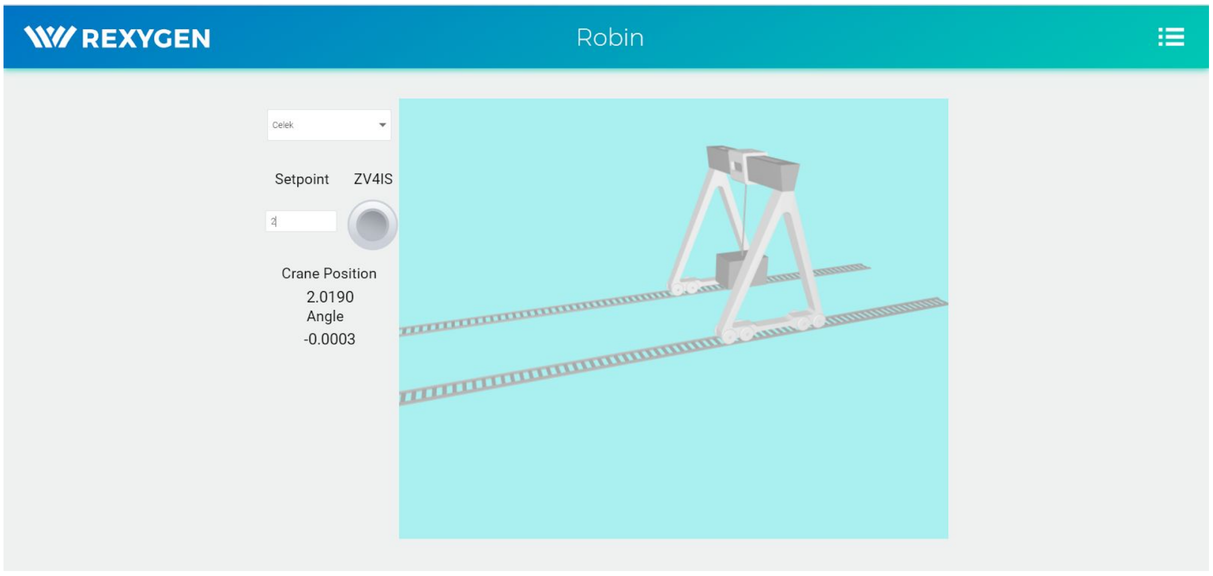


Figure 3.32: *Visualisation with first HMI elements interacting with process variables*

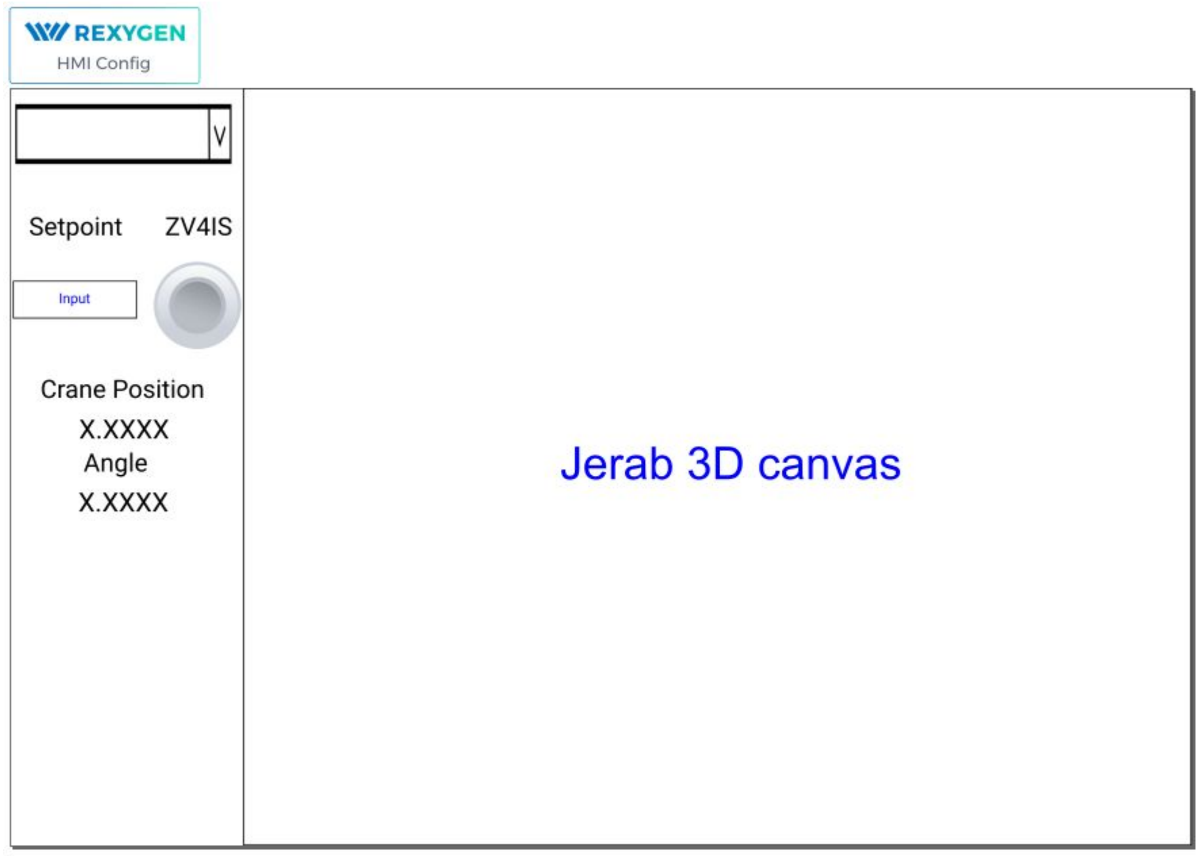


Figure 3.33: *Creation of HMI canvas*

3.6 Second crane axis

The last step related to this model was adding the movement of the cart on the crane. This was done by reusing the same pendulum-on-a-cart model with different parameters (namely a smaller weight and a smaller damping factor of the cart). The original model and this model are not connected, hence the axes are independent and the resulting effect is not exactly realistic. Nevertheless, for such small angles that this model employs, the difference is negligible and not noticeable in the 3D visualisation (Figure 3.34).

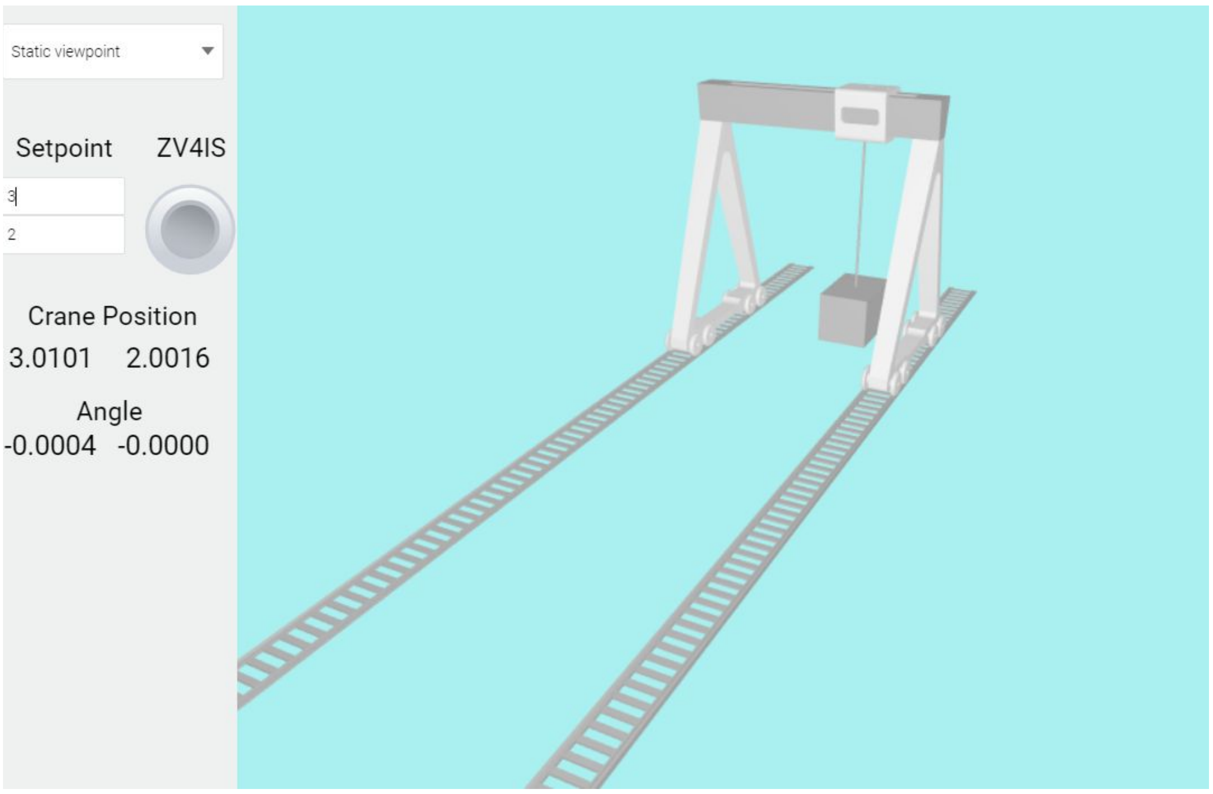


Figure 3.34: *Trolley movement visualisation*

3.7 Potential improvements

Naturally, this model is a simplification of a crane, meaning several real-world dynamics are omitted. There are three features that a proper model should have but aren't necessary for the oscillation cancellation using an input-shaping filter which was one of the main aims of this thesis.

The first feature would be considering the load as a spherical pendulum. Since the load is typically swinging in a range of mere degrees, using two independent axes is not much of an issue though. The second feature would be considering rope flexibility. The simplest way to do that would be a double-pendulum model. Such models' behaviour would be closer to that of a load swinging on a rope. However, going beyond that would drastically increase the complexity of the model while increasing accuracy negligibly. The last feature that is missing and would increase the simulation quality would be an electromotor simulation, both for the crane and the trolley movement and also for the rope winding.

It is clear that corners have been cut in the model design part of this thesis. However, the platform (Raspberry Pi 3) can barely run the model with a 3D visualisation as it is.

Chapter 4

HIL simulation

Once the model was verified, the next step was to run the separate tasks each on their own computer. A pair of Raspberry Pis running Rexygen core has been used for this part of the thesis. This phase had its own set of problems and challenges, the notable ones are mentioned in this chapter.

4.1 Platform description

As stated in the thesis instructions, the HIL simulator needs to be Rexygen-based. Being generally available, Raspberry Pi with Monarco HAT extension have been used. The device is capable of running the model at a 1 ms period with the 3D visualisation at a stable 20 frames per second. The Monarco HAT extension adds a digital and an analog voltage-controlled I/O and an RS-485 port. All mentioned parts of the extension have been utilised.

4.2 Exploring various information transfer options

It was necessary to make the two devices communicate with each other. The controller needs to send commands to the motors and receive position data. Furthermore, other variables of the system need to be set by the control system which needs to be remotely accessible. Several types of communication have been used for various purposes and are described in this subchapter.

4.2.1 Network

It is necessary to get the source code to the simulator and run it. This has been done via the TCP/IP protocol handled by Rexygen. Another use of this type of communication is accessing the control system remotely once the HIL simulation is running. The user can access the system from a web browser, see current values and their trends, the 3D visualisation and set the setpoints, the weight of the load, required length of the rope and change the parameters of the input shaping filter.

4.2.2 Analog and digital I/O

To control the motors and receive emulated position sensor signals, the analog I/O of the Monarco platform has been used. These inputs and outputs work with variable voltage in the range of 0-10 V. Besides that, digital signals (capable of sending boolean zeros and ones) have been used to indicate the direction of the motor control signal and to reset the model.

4.2.3 RS-485 (Modbus)

Since the Monarco HAT extension does not offer enough analog channels for all variables, an alternative had to be found. The RS-485 feature of the Monarco with Modbus drivers included in Rexygen seemed like a feasible solution. The setup was unexpectedly easy, though the complexity of the model (including the 3D visualisation) required changes in the Monarco configuration.

4.3 Changes to the model

The model described above was working in controlled conditions. To make it match the demands coming along with the HIL simulations, adjustments had to be made to every aspect of the model and several features were added. This subchapter explains all the changes made in order to satisfy the needs of the HIL simulation.

4.3.1 Standalone system

Transition to the HIL simulation brought several requirements for model changes that could be classified into 4 categories: model dynamics, communication, optimization and visualisation.

In terms of model dynamics, the main change was the usage of differential-equations based models instead of a state-space one. Hardstops have also been introduced so that the crane and trolley will not go too far in any direction and a small bounce has been added when the crane bumps into the end of tracks. The length of rope is now being changed by a first-order model simulating a motor to introduce a continuous length change.

Communication with the control unit has been altered significantly, as the variables going via RS-485 interface were interpolated into 12-bit numbers and the control and sensor signals have been adjusted to fit the available voltage of the analog interface.

Since the computational power of the selected platform is limited and needs to accommodate both the system model and the visualisation, slight changes have been made to make the load lighter. For example, the differential equations of the cart-and-pendulum model that each originally had their own separate block have been integrated into one block.

Several new variables have been introduced to the model, meaning they needed to be added to the visualisation.

4.3.2 Standalone controller

No structural changes have been made to the PID controller, but features handling the variables and accessibility by a remote workstation were added. The controller has been re-tuned for a new range of variables and a dead-zone was introduced to limit the impact the signal noise has on control. Nyquist diagram and responses for various parameters are shown in Figures 4.1 and 4.2.

The PID controller now receives position data via analog inputs and sends the commands over analog outputs, with direction indication via digital output. Other variables are transferred using Modbus protocol.

Variables like designated rope length, load weight and input-shaping filter parameters can be adjusted with the option of setting recommended parameters for a given rope length.

All of these new features were fitted in the HMI, along with a trend graph of several system variables (Figure 4.3, 4.4).

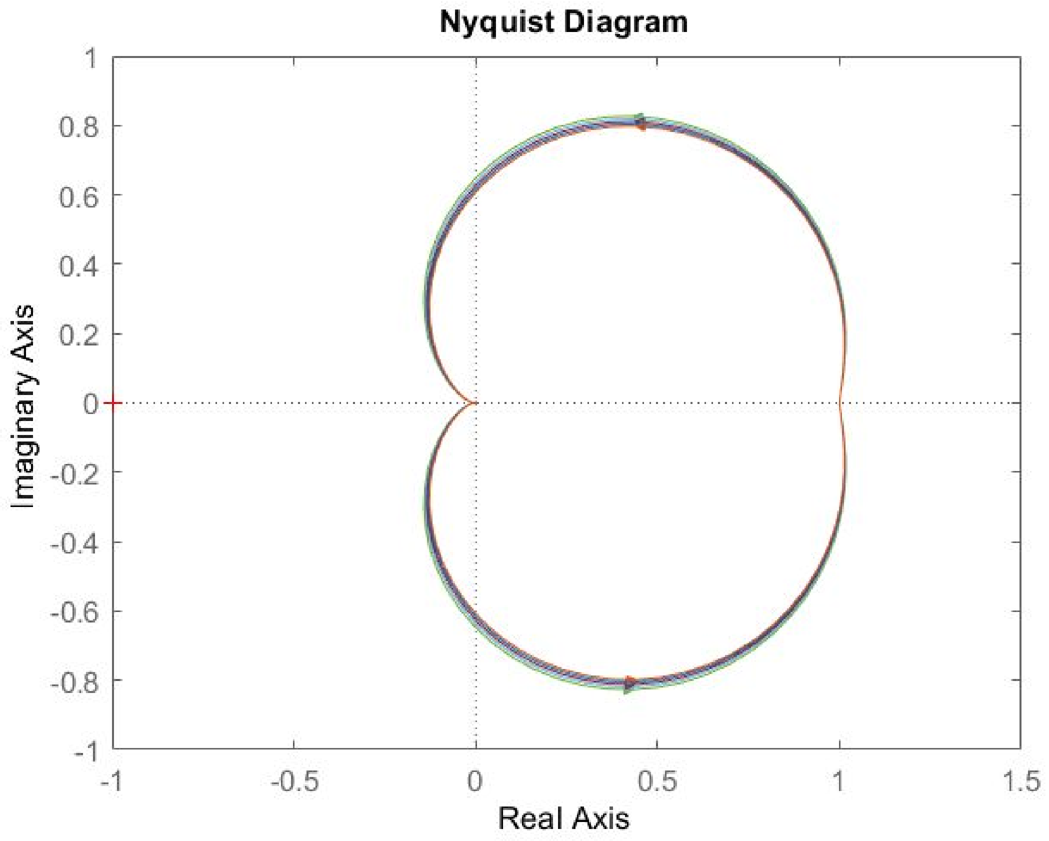


Figure 4.1: *Nyquist diagram of closed-loop system with samples of parameters from an acceptable range. In all cases, the system is stable according to the Nyquist criterion.*

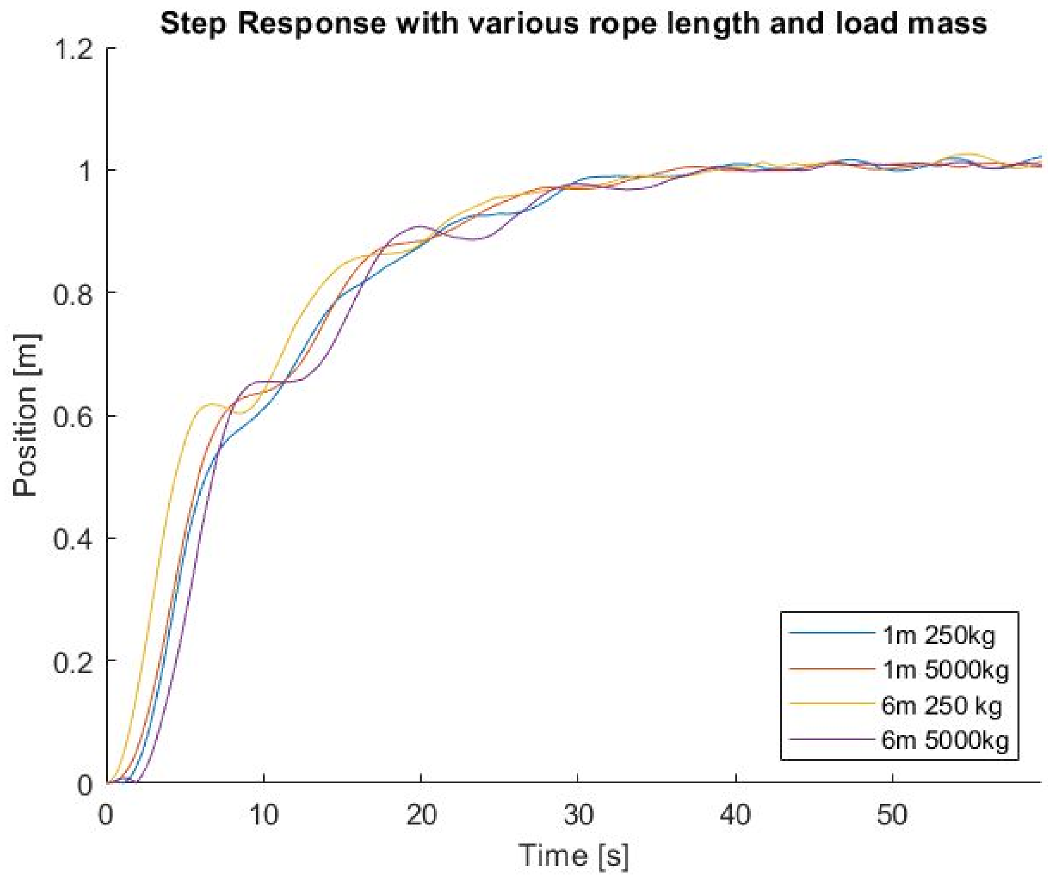


Figure 4.2: *Step response of a system with various parameters, further proving robustness of the controller*

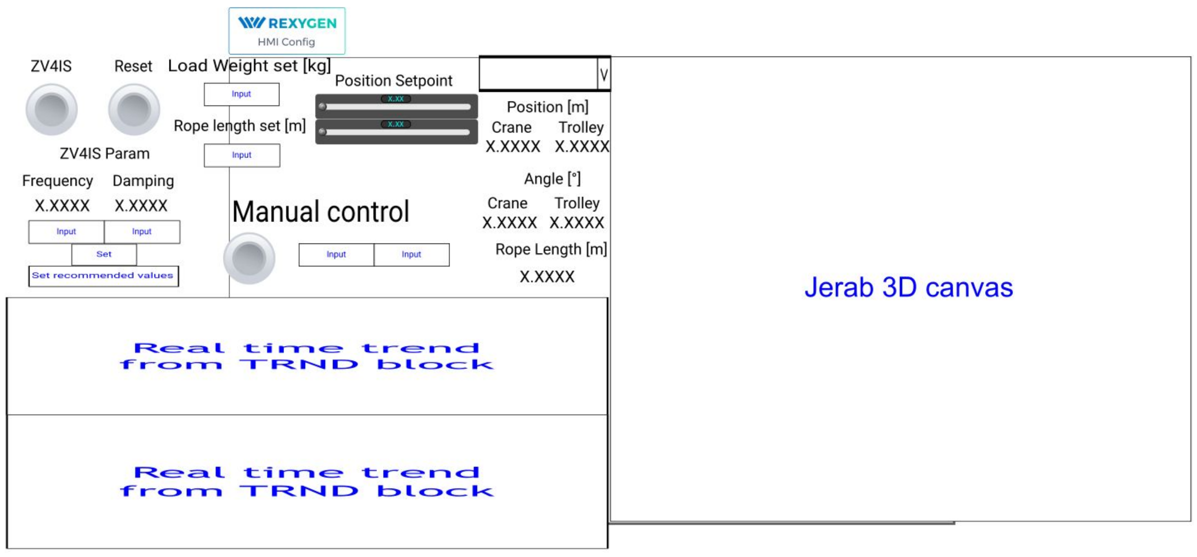


Figure 4.3: *Creation of web page HMI canvas*

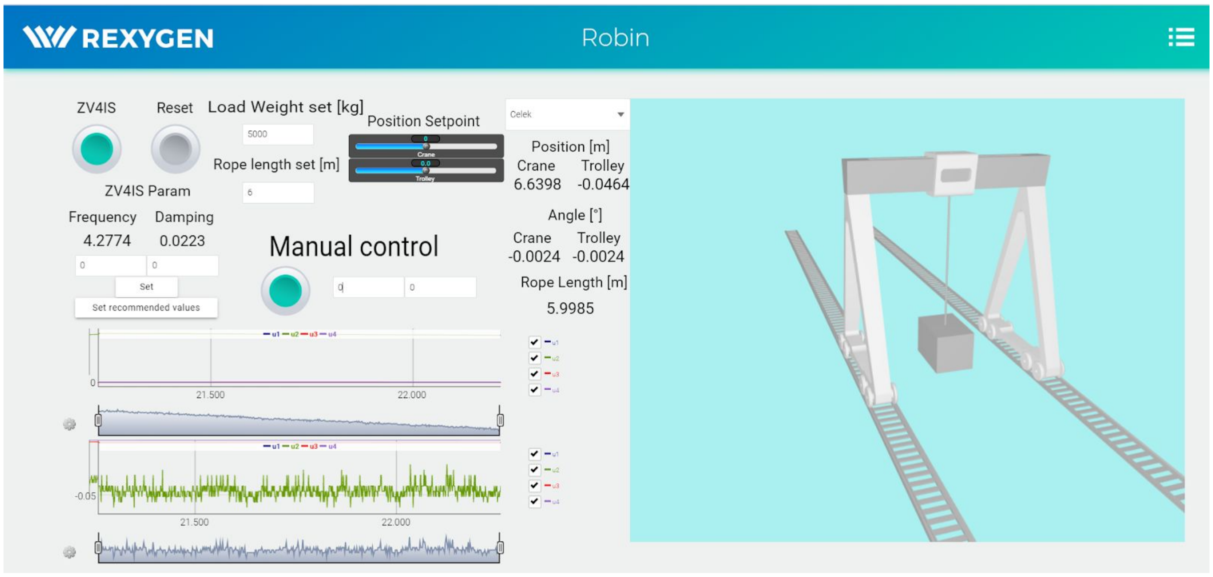


Figure 4.4: *Near-final version of the workstation visualisation*

4.4 HIL simulations

With all problems solved and goals achieved, the HIL simulator is working and is the main result of this thesis. Hardware-wise, the simulator consists of two Raspberry Pi computers with Monarco HAT extensions, connected to each other via analogue wires and a RS-485 cable. The computer simulating the crane has a display connected to it, rendering the non-interactive visualisation in real-time. This computer communicates only with the other computer which simulates the control system. The visualisation of the control system is accessible from any modern internet browser and is interactive. The user can adjust setpoints and some parameters for the controller, as well as some variables for the system. All of this is done remotely online without a direct cable connection to the system or the controller. Such simulators could be used in control theory courses.

4.5 Similarities to commercial HW

The final HIL simulation of this thesis resembles its industrial counterpart in several ways. The defining feature is its setup: a controlled system simulation (which would potentially work if replaced by a real system) communicates with the control unit via Modbus protocol and analogue signals controlling the motors and simulating the position sensors. The control unit can be accessed by any device (workstation) with a web browser, allowing the user to view the trends of variables and to set the setpoints and other variables of both the system and the controller. The control unit runs Rexygen software, which is compatible with some industrial PLCs.

4.6 Potential use

This simulation has two potential use cases. The more probable one would be using it as a practice simulation during one of the control theory courses at the University of West Bohemia. The students could control the crane using various controller synthesis methods, identify the system or make their own visualisation. The other use case could be a showcase of 3D visualisation capabilities of the Rexygen system and Monarco HAT display accessory.

Conclusion

The main focus and outcome of this thesis was a HIL gantry crane simulator as a result of the XIL process. Connecting the phases brought its own set of issues that were solved which was the main challenge of the thesis. As the automatic control education was not at all about connecting the individual courses in a meaningful way, there was an incentive to work on this kind of thesis. Another contribution was creating a 3D visualisation and running it on the system part of the HIL simulator and creating a workstation HMI for the controller hardware that is accessible via the internet from a web browser.

The first and second chapters of the thesis described various aspects of system engineering. Namely, model-based and knowledge-based system engineering (with a mention of knowledge management methods), then the XIL process with a brief explanation of each phase (MIL, SIL, PIL, HIL) and the current state of these phases in both industry and education. Then, the technologies used in this thesis were described, including the process of going through the XIL phases to create the final HIL simulator (Figure 5.1).

The third chapter focused on models derived for the gantry crane, their verification using simulations and controller synthesis. The process of creating a visualisation (both 2D and 3D) with an HMI interface was also displayed there.

The fourth chapter was all about making the model and the controller work in a HIL simulator setup. Three types of communication methods have been utilised, again, each with their own challenges like baud rate limitations of Modbus or accuracy of the analog I/O of the Monarco HAT. Features have been added for both the controlled system and the controller to make the simulation feel more industrial-grade like.

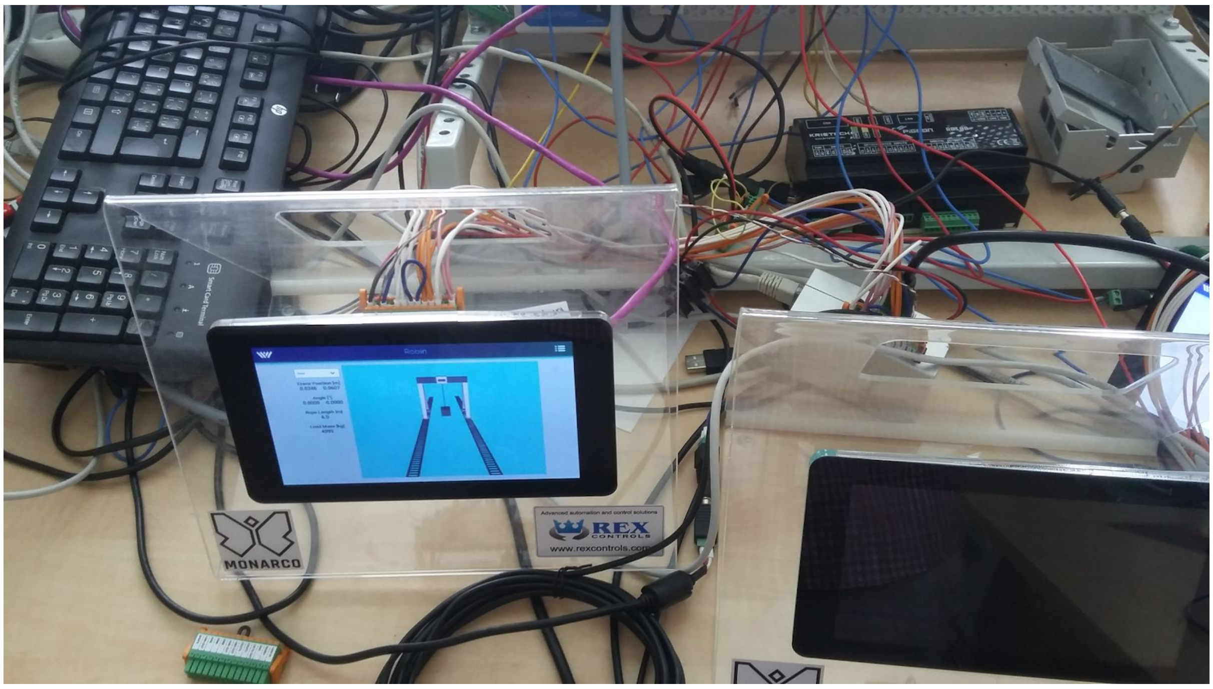


Figure 5.1: *Photo of running HIL simulator. Unit with the screen turned off runs the controller part.*

It is fair to say that no single part of this thesis went deep into detail. The model was simple (but sufficient), controller synthesis did not use any advanced method (which, if this simulation will be used in control theory classes, leaves an opportunity for the students to employ any method taught there) and the 3D model parts were re-used from a public library (though creating an assembly required understanding of a CAD software). As stated before, repeating content of individual courses with an overload of details that do not really contribute to practical skills was avoided to create something that was not specifically a part of the studies.

A possible follow-up work could create a physical component model based on SimScape and/or Modelia, export it as an FMI model and create a Rexygen-based HIL simulator [17] for such system.

Bibliography

[1] A. Wayne Wymore
https://en.wikipedia.org/wiki/A._Wayne_Wymore

[2] FMI Standard official webpages
https://fmi-standard.org/

[3] Gantry Crane model for 3D printing
https://www.thingiverse.com/thing:535197

[4] INCOSE official webpages
https://www.incose.org/about-systems-engineering/system-and-se-definition

[5] Inverted Pendulum: System Modeling
http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling

[6] KM World Magazine
https://www.kmworld.com/About/What_is_Knowledge_Management

[7] Knowledge-based engineering
https://en.wikipedia.org/wiki/Knowledge-based_engineering

[8] Knowledge management
https://en.wikipedia.org/wiki/Knowledge_management

[9] Lapping, Andy & Silberbauer, Amy. IBM Continuous Engineering Overview
https://youtu.be/BSUfD0INbSw

[10] Model-based systems engineering
https://en.wikipedia.org/wiki/Model-based_systems_engineering

[11] Modelon official webpages explaining FMI standard
https://www.modelon.com/functional-mock-up-interface-fmi/

[12] PIDlab official webpages
https://www.pidlab.com/cs/

[13] Software Education - MBSE Introduction
https://youtu.be/8d7WGUqJelQ

[14] V-Model
`https://en.wikipedia.org/wiki/V-Model`

[15] Venkatesh Mane - Difference between MIL SIL PIL HIL
`https://youtu.be/EZthOn4_0rw`

[16] Ästrom, K. J., Hägglund, T. „PID Controllers: Theory, Design, and Tuning“ 2nd Edition, Research Triangle Park : ISA-The Instrumentation, Systems and Automation Society, 1995.

[17] Cech, Martin & Königsmarková, Jana & Reitinger, Jan & Balda, Pavel. (2017). Novel tools for model-based control system design based on FMI/FMU standard with application in energetics. 416-421. 10.1109/PC.2017.7976250.

[18] Cech, Martin & Goubej, Martin & Sobota, Jaroslav & Visioli, Antonio. Model-based system engineering in control education using HIL simulators

[19] Estefan, Jeff. (2008). Survey of Model-Based Systems Engineering (MBSE) Methodologies. INCOSE MBSE Focus Group. 25.

[20] Friedenthal, Sanford & Griego, Regina & Sampson, Mark. (2009). INCOSE Model Based Systems Engineering (MBSE) Initiative.

[21] Goubej, Martin & Schlegel, Milos. (2010). Feature-based parametrization of input shaping filters with time delays. 10.3182/20100607-3-CZ-4010.00045.

[22] Hirschfeld, Karin. (2006) Retention and Fluctuation: Keeping staff - Losing staff

[23] Holla, Devaraja. “Knowledge-Based Engineering (KBE)” (2015).

[24] McInerney, Claire (2002). "Knowledge Management and the Dynamic Nature of Knowledge". Journal of the American Society for Information Science and Technology.

[25] Prasad, Brian & Rogers, Jeff. (2005). A Knowledge-Based System Engineering Process for Obtaining Engineering Design Solutions. ASME 2005 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Paper No. DETC2005-85561, pp. 477-488; 12 pages , Volume 3: 25th Computers and Information in Engineering Conference, Parts A and B. 3. 477-488. 10.1115/DETC2005-85561.

[26] Reitinger, Jan & Cech, Martin & Goubej, Martin. (2013). Advanced input shaping filter 3D virtual laboratory. Proceedings of the 2013 International Conference on Process Control, PC 2013. 528-533. 10.1109/PC.2013.6581465.

[27] Riedmaier, Stefan & Nesensohn, Jonas & Gutenkunst, Christian & Düser, Tobias & Schick, Bernhard & Abdellatif, Houssem. (2018). Validation of X-in-the-Loop Approaches for Virtual Homologation of Automated Driving Functions.

[28] Sobota, Jaroslav & Goubej, Martin & Königsmarková, Jana & Cech, Martin. (2019). Raspberry Pi-based HIL simulators for control education. IFAC-PapersOnLine. 52. 68-73. 10.1016/j.ifacol.2019.08.126.

Attachments

PDF Version of this thesis, models, figures and source codes and other attachments can be found on following link:

https://drive.google.com/drive/folders/1Z5ysCfBw_rXFMCiLMfNO_V-NaRf_v_zC?usp=sharing

If you have any questions or the link is not working anymore for some reason, feel free to contact me at: milanvosahlo@mail.com