Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Research & Development Project Report

Detection of Motion Faults and Robust control of an Omnidirectional Robot Using Machine Learning

Authors:

Chaitanya Hebbal

chaitanya.hebbal@smail.inf.h-brs.de

Matrikel Nr: 9027159

Youssef Mahmoud

yousef_mahmoud@live.com

Matrikel Nr: 9024421

Supervisors:

Prof. Dr. Paul Plöger

paul.ploeger@h-brs.de

M.Sc. Anastassia Küstenmacher

anastassia.kuestenmacher@h-brs.de

A report submitted in fulfilment of the completion for the R& D Project

in

Bonn-Rhein-Sieg University of Applied Sciences

Department of Computer Science

January 2017

Declaration of Authorship

This R&D project report titled, 'Detection of Motion Faults and Robust control of an Omnidirectional Robot Using Machine Learning' and the work presented has been completed in cooperation of Youssef Mahmoud & Chaitanya Hebbal. We as contributors confirm that:

- This work was done wholly or mainly while in candidature for a master degree at this University.
- Where any part of this report has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where we have consulted the published work of others, this is always clearly attributed.
- Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this report is entirely our own work.
- We have acknowledged all main sources of help.
- The project work was completely done in cooperation by both authors. However, report writing was split up by both authors as follows: The Chapters 1, 5, & 6 were authored by Youssef Mahmoud. The chapters 2, 3, & 4 were authored by Chaitanya Hebbal

Signatures: _____

Date: _____

Bonn-Rhein-Sieg University of Applied Sciences

Abstract

Department of Computer Science

R &D Project Report

Detection of Motion Faults and Robust control of an Omnidirectional Robot using Machine learning

by Chaitanya Hebbal, Youssef Mahmoud

Autonomous mobile robots are becoming more relevant in applications where human life can be at risk. Applications such as space exploration, search and rescue, fire fighting put human life in extreme danger. With the increase of dependence on these autonomous robots, the more complicated the tasks that are to be executed become, and a bigger number of software and hardware components are involved. With that being said, the occurence of faults is inevitable.

The ability of detecting faults and identifying them is an integral part in making any mobile platform more autonomous. Many efforts have been made previously to propose solutions on how to tackle this problem, however, a complete solution is not yet available. This research and development project aims to build a strong foundation for detection of motion faults and robust control of an omnidirectional robot. Since omnidirectional robots have the advantage of high maneuverability and mobility, they are of high interest to current research.

This project proposes the accurate kinematic and dynamic modelling of a four wheeled omnidirectional platform in order to use them for fault detection, robust control, and simulation purposes. The accurate development of these models is crucial to the process of making the mobile platform more autonomous since these models can predict the correct behaviour of the platform. A kinematic and dynamic model with very high accuracy have been developed and tested for different scenarios using the Kuka YouBot and a velocity controller was successfuly implemented on the dynamic model.

Acknowledgements

We would like to thank our supervisors Prof. Paul Ploeger and M.Sc. Anastassia Kuestenmacher for their continued support. They have given us plenty of their time and guidance in order to steer us in the right direction. We would also like to thank the Bonn-Rhein-Sieg Univerisy of Applied Sciences RoboCup team for their effort and guidance while using the youBot. We would also like to thank Alex, Santosh and Boris for helping us in gathering the data. Finally, we would like to thank our families and friends for their support throughout the project.

Contents

Chapter 1

Introduction

1.1 Motivation

Autonomous wheeled mobile robots are being extensively used in domains such as service robots, logistics, industrial, agriculture and healthcare. The reason being that, wheeled mobile robots have the ability to freely move around in their predefined workspace in order to perform the specified task. This property of wheeled mobile robot makes it suitable for variety of applications in both structured and unstructured environments.

Sidek [1] defines Wheeled Mobile Robot(WMR) as a wheeled vehicle that can move autonomously without any assistance from human operators. These are equipped with a set of motorized actuators and an array of sensors that aid the robot in carrying out its task. Most of the modern Wheeled mobile robots make use of mecanum wheels as they provide a high-level of manoeuvrability. Mecanum wheels have 3-degress of freedom(DOF): one, in the direction of wheel rotation, second, in the direction of roller rotation and third, the rotational slip about vertical axis passing through point of contact[2]. Mecanum wheels have the following advantages: compact design, simple to control, low speed, and low pushing force when moving diagonally [3].

With the increasing demand for wheeled mobile robots, the need to make them more reliable becomes more important now than ever. Wheeled mobile robots play an important role in a variety of applications by providing mission critical services. Applications such as search and rescue, planetary exploration, and nuclear waste disposal are applications where wheeled mobile robots can replace humans as it is safer and also in many cases impossible for humans to intervene. With that being said, the correct behaviour of the robot is of great importance, as even the smallest of faults can lead to catastrophic failures.

In order to maintain high level of quality of service in such dynamic environments,there is a need for wheeled mobile robots to be equipped with intelligent mechanisms, capable of locating faults in a timely manner, so that corrective measures can be taken to prevent faults from causing severe damage. Fault diagnosis is one of the key techniques for the development of such intelligent mechanism for wheeled mobile robots in order to provide a robust service.

[4]Fault diagnosis includes three steps:

1. Fault detection: This is the basic step, in which the presence of fault or malfunction in a system is detected.
2. Fault isolation: This step is used to determine the location of a faulty component.
3. Fault identification: This step is used to determine the type, shape and size of fault.

Gao et al,[4] broadly classify fault diagnosis techniques into following categories: Model-based, Signal-based, Knowledge-based, Hybrid, Active fault diagnosis techniques respectively. The interested reader can go through the survey papers [4] [5] for more information on different fault-diagnosis approaches.

From the literature review we found that for Wheeled mobile robots, model-based diagnosis methods have been successfully applied to real systems. In addition, the model-based methods offer performance assessment of wheeled mobile robots on a system level, rather than on device or a component level. Also, from the control perspective, model-based control approaches are the most widely used techniques to control a wheeled mobile robot [6][1].Hence, in our research and development project we make use of model-based approach to control design and fault-diagnosis.

1.2 Problem Statement

To build kinematic and dynamic model of omni-directional mobile robot with mecanum wheels and validate it. The validated model is to be used for advanced controller design for precision control, and also to develop a robust on-board fault diagnosis system.

1.3 Organization of report

The report is structured as follows:

Chapter 2 discusses the state of art in model-based diagnosis and control design.

Chapter 3 provides a detailed derivation of kinematic model and dynamic model of the omni-directional robot with mecanum wheels. We also derive the state space representation of the dynamic model.

Chapter 4 provides a detailed approach to residual generation by means of structural analysis.

Chapter 5 describes the experimental setup that was used to validate the obtained mathematical models. We also discuss the findings of our experiments.

Chapter 6 provides a detailed conclusion of our work. We also discuss the problems encountered, limitations and also future work that will be taken up as part of the master thesis.

Chapter 2

State-Of-The-Art

This chapter will give a brief overview of the state-of-the-art of some of the topics within the scope of this project. As mentioned in the problem statement in chapter 1, our goal is to build a kinematic and dynamic model of a four wheeled omnidirectional robot in order to use them for model based control, fault detection and other applications.

2.1 Modelling

Model based approaches require some kind of model to be built such as a kinematic or dynamic model. Each model gives insight to some characteristics of the system. Since the goal of the project is to be as accurate as possible when deriving a model, the literature gathered in this section provide guidelines on how to do so.

Seeing as our targeted platform is a four wheeled mecanum omni-directional robot, we examined papers that used similar approaches. Agullo [7] Describes a mecanum wheel as a wheel that has three degrees of freedom. These wheels have the advantage of great maneuverability because of the ability to move at a fixed angle relative to the robot's platform. The authors describe some characteristics of a platform with four mecanum wheels attached by deriving the kinematic model. This model takes into consideration the geometry of the platform and the wheel in order to build relations between the platform's transversal and longitudinal velocities and the wheel's individual velocity. Some of the important geometric parameters include the length, width and physical charactersitics of the mecanum wheel such as its radius, roller angle etc. that will be further explained in detail in chapter 3. The same authors also develop a dynamic model in [8], in which their aim is to get more insight on the behaviour of the omnidirectional platform. The authors develop equations of motion of the platform by considering

inertias and momentum. A model determining the driving torques of the wheels is developed using Appell equations of motion. In classical mechanics, also known as newton mechanics, the study of motion can be formulated in different ways. Some such ways include Newton's equations of motion, Langrangian mechanics and Appell's equations of motion. Appell's equations do not differ from the other derivations of motion; however, they provide a more convenient method of derivation when taking into consideration non holonomic motion.

Looking into the research by Indiveri [9], a general model is derived that can be used for an omnidirectional platform with N wheels. The aim of this paper is to be able to describe the kinematic relations of an omnidirectional platform where the wheel arrangments do not cause any singularities. Moreover, the paper strives to decouple the commanded angular and transversal and longitudinal velocities. The model can then be used for different applications such as velocity control or trajectory tracking controller design.

Viboonchaicheep [10] describes a kinematic and dynamic model is derived for a four wheeled omnidirectional platform. The authors use the same methods mentioned in the previous researches for developing the kinematic model. However, in regards to the dynamic model, the kinetic and friction forces of the platform are considered. In order to, to derive the driving torques for the wheels, the Lagrangian method is used. The dynamic model is then transformed into state space form so that it can be used for velocity and position control of the platform.

2.2 Control

As mentioned in the previous section, many of the papers reviewed use the developed model for some form of control, that being velocity or position control. In the aforementioned paper by Viboonchaicheep [10], The developed dynamic model was then used for control purposes. To control the velocity of the platform, the authors use a centralized solution where all four wheels of the robot are controlled using a single controller. The controller works by trying to eliminate the error in the output of the system through altering the input to the system. In this specific research, the authors add an integral gain to the system to try to eliminate the error and use the pole assignment method in order to force the system to behave as they wish. To control position, a simple PD controller is used on top of the velocity controller, making the system have a cascaded control system with the ability to control each phenomena independently.

Another method of controlling the four wheeled platform is by applying a decentralized solution, such as in [9]. By using simple PID solutions, accurate control of the system can be achieved by deploying a controller to each wheel individually.

Different forms of control can also be studied such as fault tolerant control where the controller can still navigate the robotic platform correctly after compensating for an occuring fault. Papers such as [11], [12], and [13] give more insight into this topic regarding omnidirectional platforms.

2.3 Fault

We use model-based approach to fault detection and diagnosis. The origin of model-based approach to fault diagnosis started with work of Beard [14]. The idea was to replace hardware redundancy methods by analytical redundancy methods. Although hardware redundancy is reliable,it is not cost-effective and occupies more space. In contrast, analytical redundancy is cost-effective as it is directly obtained from the model of the system and it does not occupy any space on the system since it is not a physical entity. But analytical redundancy methods suffer some drawbacks like modelling errors, the sensitivity of the models to noise, and their complexity increases with complexity of the system dynamics and control structure[4].

[15] [16] The goal of model-based approaches is to detect faults as early as possible and issue a timely warning. Model-based approaches make use of mathematical models of the system to perform fault detection and diagnosis. The Model of the system can be obtained by either applying physical principles or through system identification techniques. The approach works as follows: the mathematical model of the system is run in parallel to the real system and is supplied with same inputs as the ones given to the system. In the nominal situation, the system output and the model estimates are consistent with each other. In faulty situations, the estimate from the model and system output are inconsistent. The quantity that represents the inconsistencies between the observed values from the system and the estimated values from the mathematical model is called residual. Residuals are computed from the observable/known variable of the system and in a nominal situation the residual should be either zero or close to zero. Thus, residuals can be used for fault detection, isolation and identification.

The general schematic of analytical redundancy based approaches is as shown in Figure:2.1

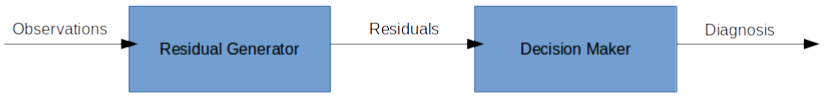

FIGURE 2.1: Schematic of analytical redundancy based FDI approaches (chow and willsky 1984)

It consists of two blocks:

1. Residual Generator : Its function is to generate residuals.
2. Decision Maker : Its function is to analyse residuals to arrive at a diagnostic decision that being, the detection and isolation of the fault.

The Best known residual generation techniques in literature are: Parity equations, Diagnostic observers and Kalman filters. Limit checking of individual plant variables is the most widely used decision making/diagnostic approach in the literature. The survey by Gertler [17] provides a detailed explanation of these techniques.

Below are some relevant publications making use of this approach:

[18] address the problem of detecting actuator faults in service robots in indoor environments. A model-based approach is proposed for detecting and identifying faults. The idea is to calculate an error-bound between estimated and measured robot states. The model of the system is used to estimate robot's next state using the current state and current values as the input. If the estimation error exceeds the error-bound, it indicates

that a fault has occurred in the system. To prevent false alarms, model uncertainties are factored in the bound calculations by considering the uncertainty bounds.

Also, Hoang [19] uses a model-based approach to address the problem of fault detection in a wheeled mobile robot. The proposed approach makes use of non-linear observer for fault isolation. The non-linear observer design is based on the dynamic model of the mobile robot. The proposed approach involves a two-step procedure. The first step is fault detection, wherein a fault is detected if the residual exceeds certain threshold. The second step is to activate the three observers to isolate the following three types of fault:Right wheel fault, left wheel fault, other dynamic faults.

Obtaining accurate models describing faulty behaviour is a difficult and expensive process because it needs either experimental data from a faulty process or it needs profound knowledge of the process. These are not easily available, more so in the early stages of development. To design a reliable and robust diagnosis system,it is important to analyse the level of knowledge needed to identify the potential faults to be diagnosed and also ways to isolate them.Structural analysis as a tool, enables us to gather this kind of information [20].

Among different methods of residual generation only few are applicable to non-linear systems as well. Amidst them, structural analysis technique provides a feasible solution to residual generation in non-linear systems[16].

In light of its numerous advantages, structural analysis techniques have been successfully applied for residual generation in literature, for applications like ship propulsion benchmark, unmanned ground vehicle, fixed-wing aircraft motion[21][22][23][24][25][26]. In addition, there are readily available tools to perform structural analysis. Some of them are given below:

- SATOOL : [27] This is a matlab-based software tool for structural analysis of system structure at a high-level of specification. It is a rapid prototyping tool with an easy to use GUI(Graphical User Interface).The developers of the tool have also provided a detailed user manual [28].
- SaTool : This is a python version of the SATOOL, and is developed in-house by Deebul Nair of Autonomous systems program, BRSU. It is still under active development and supports some of the important features needed to perform structural analysis. We made use of this tool for generating residuals [29].

Some relevant publications that make use of this approach in the domain of wheeled mobile robot is given below:

[16] also addresses the problem of actuator fault diagnosis in a 4-wheeled skid steering mobile robot by means of model-based approach. A kinematic model of the robot is developed. This kinematic model is then used to derive the structural model of the system. By applying structural analysis procedure on the structural model, a set of parity equations is obtained. These parity equations can be used as residual generators which can then be used for fault detection and diagnosis.

Fourlas [30] makes use of the fault detection approach developed in [16]. In addition to fault detection,the authors propose fault accommodation framework for handling the problem of left or right set of tires becoming flat, of skid steering mobile robot during operation. The proposed accommodation method is based on a Recursive Least Squares(RLS) approximation. The proposed approach functions in the following matter, Whenever the left of right set of tire become flat the faulty wheel radius is approximated by means of Recursive Least Squares and a new control input is generated to compensate for fault.

In light of numerous advantages offered by structural analysis approach for residual generation and the availability of automated tools to apply it, we decided to follow the same approach for our project. The detailed description of applying structural analysis on omni-directional mobile robots is delved into in Chapter 4.

Chapter 3

Modelling of Physical System

In this chapter we derive the kinematic and dynamic models of the omni-directional mobile robot with mecanum wheels. We also provide the state-space representation for the derived models that will be used for control design in Chapter 4.

3.1 Kinematic Model

Figure 3.1 represents a 4-mecanum wheel based omni-directional mobile platform with frames attached. Frame $\{I\}$ denotes an inertial frame and Body Frame $\{B\}$ is attached to the center of mass of the mobile platform. The translational and rotational velocities of the platform *i.e.* v_x, v_y and ω are expressed in terms of the body frame. Let the rotational velocity of the wheels be denoted by $\dot{\theta}_i$ where $i = (1, 2, 3, 4)$. Let the translational velocity of the wheels be denoted as V_{iW} and let V_{ir} denote the tangential velocity of free roller touching the ground surface, where $i = (1, 2, 3, 4)$.

[12]Wheel velocity can be expressed in terms of velocity components of wheel and roller velocities in the body frame $\{B\}$ as,

$$\begin{aligned} V_{1X} &= V_{1W} + V_{1r}\cos\alpha \quad V_{1Y} = V_{1r}\sin\alpha \\ V_{2X} &= V_{2W} + V_{2r}\cos\alpha \quad V_{2Y} = -V_{2r}\sin\alpha \\ V_{3X} &= V_{3W} + V_{3r}\cos\alpha \quad V_{3Y} = -V_{3r}\sin\alpha \\ V_{4X} &= V_{4W} + V_{4r}\cos\alpha \quad V_{4Y} = V_{4r}\sin\alpha \end{aligned} \tag{3.1}$$

where α is the offset angle of roller of mecanum wheel. In case of youbot platform $\alpha = 45°$.

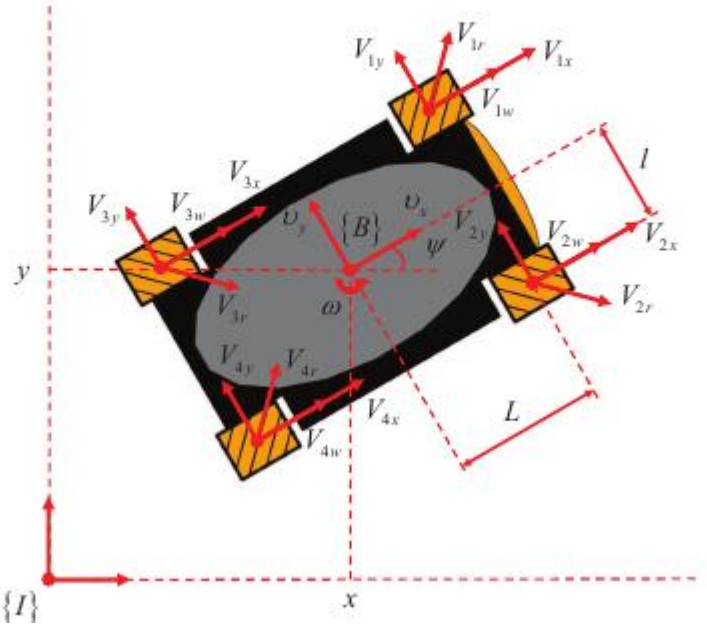

FIGURE 3.1: Omni-directional mobile platform with 4-mecanum wheels[12]

Since the wheels are rigidly attached to the platform, we can also express wheel velocities in terms of platform velocities v_x, v_y and ω

$$\begin{aligned} V_{1X} &= v_x - l\omega \quad V_{1Y} = v_y + L\omega \\ V_{2X} &= v_x + l\omega \quad V_{2Y} = v_y + L\omega \\ V_{3X} &= v_x - l\omega \quad V_{3Y} = v_y - L\omega \\ V_{4X} &= v_x + l\omega \quad V_{4Y} = v_y - L\omega \end{aligned} \tag{3.2}$$

where l and L denote the lateral and longitudinal distance of wheels from the center of mass respectively. Substituting equation 3.1 in 3.2 we get,

$$\begin{aligned} V_{1W} + V_{1r}\cos\alpha &= v_x - l\omega \quad & V_{1r}\sin\alpha &= v_y + L\omega \\ V_{2W} + V_{2r}\cos\alpha &= v_x + l\omega \quad & -V_{2r}\sin\alpha &= v_y + L\omega \\ V_{3W} + V_{3r}\cos\alpha &= v_x - l\omega \quad & -V_{3r}\sin\alpha &= v_y - L\omega \\ V_{4W} + V_{4r}\cos\alpha &= v_x + l\omega \quad & V_{4r}\sin\alpha &= v_y - L\omega \end{aligned} \tag{3.3}$$

We know that Velocity of the wheel is given by equation 3.4 , where R_W is the radius of the wheel and θ_i is i^{th} wheel rotation velocity.

$$V_{iW} = R_W \cdot \dot{\theta}_i \tag{3.4}$$

Substituting equation 3.4 in 3.3 we get,

$$\begin{aligned} R_W \cdot \dot{\theta_1} + V_{1r}\cos\alpha = v_x - l\omega \quad & V_{1r}\sin\alpha = v_y + L\omega \\ R_W \cdot \dot{\theta_2} + V_{2r}\cos\alpha = v_x + l\omega \quad & -V_{2r}\sin\alpha = v_y + L\omega \\ R_W \cdot \dot{\theta_3} + V_{3r}\cos\alpha = v_x - l\omega \quad & -V_{3r}\sin\alpha = v_y - L\omega \\ R_W \cdot \dot{\theta_4} + V_{4r}\cos\alpha = v_x + l\omega \quad & V_{4r}\sin\alpha = v_y - L\omega \end{aligned} \tag{3.5}$$

Re-arranging terms in equation 3.5

$$\begin{aligned} V_{1r} = (v_x - l\omega - R_W \cdot \dot{\theta_1})/\cos\alpha \quad & V_{1r} = (v_y + L\omega)/\sin\alpha \\ V_{2r} = (v_x + l\omega - R_W \cdot \dot{\theta_2})/\cos\alpha \quad & V_{2r} = -(v_y + L\omega)/\sin\alpha \\ V_{3r} = (v_x - l\omega - R_W \cdot \dot{\theta_3})/\cos\alpha \quad & V_{3r} = -(v_y - L\omega)/\sin\alpha \\ V_{4r} = (v_x + l\omega - R_W \cdot \dot{\theta_4})/\cos\alpha \quad & V_{4r} = (v_y - L\omega)/\sin\alpha \end{aligned} \tag{3.6}$$

Equating in terms of V_{ir} and $\alpha = 45°$ we get,

$$\begin{aligned} R_W \cdot \dot{\theta_1} &= v_x - v_y - (l + L)\omega \\ R_W \cdot \dot{\theta_2} &= v_x + v_y + (l + L)\omega \\ R_W \cdot \dot{\theta_3} &= v_x + v_y - (l + L)\omega \\ R_W \cdot \dot{\theta_4} &= v_x - v_y + (l + L)\omega \end{aligned} \tag{3.7}$$

Expressing equation 3.7 in matrix form we get,

$$\begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \\ \dot{\theta_3} \\ \dot{\theta_4} \end{bmatrix} = \frac{1}{R_W} \begin{bmatrix} 1 & -1 & -(l+L) \\ 1 & 1 & (l+L) \\ 1 & 1 & -(l+L) \\ 1 & -1 & (l+L) \end{bmatrix} \cdot \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \tag{3.8}$$

Equation 3.8 represents the inverse kinematics of omni-directional mobile robots with mecanum wheels. The above equation can be expressed as 3.9

$$V_W = T \cdot V_O \tag{3.9}$$

where $V_W = \begin{bmatrix} \dot{\theta}_1 \dot{\theta}_2 \dot{\theta}_3 \dot{\theta}_4 \end{bmatrix}^T$ and $V_O = \begin{bmatrix} v_x v_y \omega \end{bmatrix}^T$. T is the transformation matrix given by,

$$\begin{bmatrix} 1 & -1 & -(l+L) \\ 1 & 1 & (l+L) \\ 1 & 1 & -(l+L) \\ 1 & -1 & (l+L) \end{bmatrix}$$

[10]The forward kinematic equation can be obtained by taking the pseudo-inverse of the the transformation matrix T using the equation 3.10 , where $T^+ = (T^T \cdot T)^{-1} T^T$

$$V_O = T^+ \cdot V_W + (I - T^+ \cdot T)\omega \tag{3.10}$$

The forward kinematic equation thus obtained is given below

$$\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = \frac{R_W}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ \frac{-1}{(l+L)} & \frac{1}{(l+L)} & \frac{-1}{(l+L)} & \frac{1}{(l+L)} \end{bmatrix} \cdot \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \end{bmatrix} \tag{3.11}$$

From equation 3.11, we can express v_x, v_y and ω in terms of $\dot{\theta}_i$

$$v_x = \frac{R_W}{4}(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3 + \dot{\theta}_4) \tag{3.12}$$

$$v_y = \frac{R_W}{4}(-\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3 - \dot{\theta}_4) \tag{3.13}$$

$$\omega = \frac{R_W}{4(l+L)}(-\dot{\theta}_1 + \dot{\theta}_2 - \dot{\theta}_3 + \dot{\theta}_4) \tag{3.14}$$

3.2 Dynamic Model

[10] The kinetic energy of vehicle is given by the equation 3.15, where "m" is the mass of the vehicle,"J_z" is the Moment of inertia of the vehicle and "J_w" is the moment of inertia of the wheel.

$$K = \frac{1}{2}m(v_x^2 + v_y^2) + \frac{1}{2}J_z\omega_z^2 + \frac{1}{2}J_w(\dot{\theta_1}^2 + \dot{\theta_2}^2 + \dot{\theta_3}^2 + \dot{\theta_4}^2) \tag{3.15}$$

The loss of energy due to friction is given by equation 3.16, where D_θ is the friction co-efficient.

$$D = \frac{1}{2} D_\theta (\dot{\theta_1}^2 + \dot{\theta_2}^2 + \dot{\theta_3}^2 + \dot{\theta_4}^2) \tag{3.16}$$

We derive the dynamic model of the vehicle using the lagragian formalism. The lagragian is given by equation 3.17, where K denotes kinetic energy and U denotes the potential energy of the body.

$$L = K - U \tag{3.17}$$

We assume that the robot moves on a plane, hence the potential energy is zero. In this case, the lagrangian function is equal to Kinetic energy. We also take into account the friction forces. Since friction forces cannot be expressed in terms of potential energy, we model the friction forces in terms of rayleigh potentials.

In our case , Lagrange L is same as equation 3.15,since $U = 0$ and Rayleigh potential R is same as equation 3.16 , that is,

$$R = \frac{1}{2} D_\theta (\dot{\theta_1}^2 + \dot{\theta_2}^2 + \dot{\theta_3}^2 + \dot{\theta_4}^2)$$
$$L = \frac{1}{2} m(v_x^2 + v_y^2) + \frac{1}{2} J_z \omega_z^2 + \frac{1}{2} J_w (\dot{\theta_1}^2 + \dot{\theta_2}^2 + \dot{\theta_3}^2 + \dot{\theta_4}^2)$$

We make use of the modified lagrangian function to derive the dynamic equations of motion of the robot. The modified lagrangian function is given by the equation 3.18 where Q_i denotes generalized forces and q_i denotes the generalized co-ordinates.

$$Q_i = \frac{d}{dt}(\frac{\partial L}{\partial \dot{q_i}}) - \frac{\partial L}{\partial q_i} + \frac{\partial R}{\partial \dot{q_i}} \tag{3.18}$$

Substituting equations (3.12 - 3.14) in 3.15 and applying the lagrangian function given by equation 3.18 we get,

$$\tau = M\ddot{\theta} + D_\theta \dot{\theta} \tag{3.19}$$

where, $\theta = \begin{bmatrix} \theta_1 \, \theta_2 \, \theta_3 \, \theta_4 \end{bmatrix}^T$, $\tau = \begin{bmatrix} \tau_1 \, \tau_2 \, \tau_3 \, \tau_4 \end{bmatrix}^T$ and Matrix M is given by,

$$M = \begin{bmatrix} A+B+J_w & -B & B & A-B \\ -B & A+B+J_w & A-B & B \\ B & A-B & A+B+J_w & -B \\ A-B & B & -B & A+B+J_w \end{bmatrix}$$

$$A = \frac{mR_W^2}{8}$$

$$B = \frac{J_z R_W^2}{16(L+l)^2}$$

3.2.1 State space representation

State space representation of the system is nothing but a mathematical model of the physical system that relates a set of input,output and state variables via a set of first-order differential equations. The term "State space" refers to a space whose axes are state variables and the state of a system can then be represented as a vector in that space. This approach provides a compact way to model and analyse physical systems.We make use of state space model for designing controller [31] [32].

State space model of the system is given by,

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

where $x \in \mathbb{R}^n$ denotes the state vector ,$u \in \mathbb{R}^p$ denotes the control vector $y \in \mathbb{R}^q$ denotes the output vector of the system, A is the state matrix with $\dim[A] = n \times n$, B is the Input matrix with $\dim[B] = n \times p$, C is the output matrix with $\dim[C] = q \times n$, D is the Feed-forward/feed-through matrix with $\dim[D] = q \times p$. We convert the dynamic model of the system given by equation 3.19 to state-space as follows:

let $x_1 = \theta$ and $x_2 = \dot{\theta} = \dot{x_1}$,then equation 3.19 can be re-written in terms of x_1, x_2 as,

$$\tau = M\dot{x_2} + D_\theta x_2$$

Then ,

$$\dot{x_1} = x_2$$
$$\dot{x_2} = \frac{(\tau - D_\theta x_2)}{M}$$

Expressing $\dot{x_1}, \dot{x_2}$ in terms of x_1, x_2

$$\dot{x_1} = 0 \cdot x_1 + 1 \cdot x_2 + 0 \cdot \tau \tag{3.20}$$

$$\dot{x_2} = 0 \cdot x_1 - \frac{D_\theta}{M} \cdot x_2 + \frac{1}{M} \cdot \tau \tag{3.21}$$

Expressing equations (3.20 - 3.21) in matrix form and substituting for x_1, x_2 we get,

$$\dot{x} = \begin{bmatrix} 0_{4\times4} & I_{4\times4} \\ 0_{4\times4} & -M^{-1}D_\theta \end{bmatrix} \cdot \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0_{4\times4} \\ M^{-1} \end{bmatrix} \cdot \tau \tag{3.22}$$

If we want to measure wheel position then $y = x_1$. Expressing y in terms of x_1, x_2 and τ we get,

$$y = 1 \cdot x_1 + 0 \cdot x_2 + 0 \cdot \tau$$

Substituting for x_1, x_2, we can write above equation in matrix form as,

$$y = \begin{bmatrix} I_{4\times4} & 0_{4\times4} \end{bmatrix} \cdot \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0_{4\times4} \end{bmatrix} \cdot \tau \tag{3.23}$$

If we want to measure wheel speeds then $y = x_2$. Expressing y in terms of x_1, x_2 and τ we get,

$$y = 1 \cdot x_1 + 0 \cdot x_2 + 0 \cdot \tau$$

Substituting for x_1, x_2, we can write above equation in matrix form as,

$$y = \begin{bmatrix} 0_{4\times4} & I_{4\times4} \end{bmatrix} \cdot \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0_{4\times4} \end{bmatrix} \cdot \tau \tag{3.24}$$

Equations (3.22 - 3.24) constitute the state space representation of the system. We are reproducing these equations below for easy readability.

$$\dot{x} = \begin{bmatrix} 0_{4\times4} & I_{4\times4} \\ 0_{4\times4} & -M^{-1}D_\theta \end{bmatrix} \cdot \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0_{4\times4} \\ M^{-1} \end{bmatrix} \cdot \tau$$

$$y = \begin{bmatrix} I_{4\times4} & 0_{4\times4} \end{bmatrix} \cdot \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0_{4\times4} \end{bmatrix} \cdot \tau$$

or

$$y = \begin{bmatrix} 0_{4\times4} & I_{4\times4} \end{bmatrix} \cdot \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0_{4\times4} \end{bmatrix} \cdot \tau$$

where $\dot{x} = \begin{bmatrix} \theta\, \dot{\theta} \end{bmatrix}^T$,$\theta = \begin{bmatrix} \theta_1\, \theta_2\, \theta_3\, \theta_4 \end{bmatrix}^T$,$\dot{\theta} = \begin{bmatrix} \dot{\theta_1}\, \dot{\theta_2}\, \dot{\theta_3}\, \dot{\theta_4} \end{bmatrix}^T$ and $\tau = \begin{bmatrix} \tau_1\, \tau_2\, \tau_3\, \tau_4 \end{bmatrix}^T$. $I_{4\times4}$ stands for identity matrix and $0_{4\times4}$ stands for zero matrix with dimensions 4×4.

Chapter 4

Structural Analysis And Control Design

In this chapter we derive the residuals needed for fault detection and diagnosis using structural analysis approach, using the dynamic model derived in Chapter 3. The reasons for choosing structural analysis technique is outlined in Chapter 2. We also design a low-level controller for omni-directional mobile robot using the dynamic model.

4.1 Structural analysis

This section follows the steps outlined in the book by Blanke et al.[33].The definitions,notations used in this section directly follow from the book, so it is easy for the reader to follow. The reader is suggested to go through the book for more details on structural analysis.

Structural analysis is a powerful and easy to use tool for analysing the fault detectability and fault isolability properties of the system. It enables us to evaluate the system model with respect to fault detectability and fault isolability properties using graph-based tools. In this approach, we consider only the structural information contained in the structural model of the system.

Structural model is an abstraction of behavioural model of the system wherein it takes into account only the structure of the constraints and not the constraints themselves. Constraints are nothing but relations between a set of variables. In case of omni-directional mobile robot with mecanum wheels, the set of constraints that constitute the structural model is given below:

$$
\begin{aligned}
c_1 &: \dot{\theta_1} = \frac{d\theta_1}{dt} \\
c_2 &: \dot{\theta_1} = \frac{1}{K} \cdot \left[p_1 - \frac{mR^2\dot{\theta_4}}{8} + \frac{R^2 J_z \left(\dot{\theta_2} - \dot{\theta_3} + \dot{\theta_4} \right)}{16\left(l+L\right)^2} \right] \\
c_3 &: \dot{\theta_2} = \frac{d\theta_2}{dt} \\
c_4 &: \dot{\theta_2} = \frac{1}{K} \cdot \left[p_2 - \frac{mR^2\dot{\theta_3}}{8} - \frac{R^2 J_z \left(-\dot{\theta_1} - \dot{\theta_3} + \dot{\theta_4} \right)}{16\left(l+L\right)^2} \right] \\
c_5 &: \dot{\theta_3} = \frac{d\theta_3}{dt} \\
c_6 &: \dot{\theta_3} = \frac{1}{K} \cdot \left[p_3 - \frac{mR^2\dot{\theta_2}}{8} + \frac{R^2 J_z \left(-\dot{\theta_1} + \dot{\theta_2} + \dot{\theta_4} \right)}{16\left(l+L\right)^2} \right] \\
c_7 &: \dot{\theta_4} = \frac{d\theta_4}{dt} \\
c_8 &: \dot{\theta_4} = \frac{1}{K} \cdot \left[p_4 - \frac{mR^2\dot{\theta_1}}{8} - \frac{R^2 J_z \left(-\dot{\theta_1} + \dot{\theta_2} - \dot{\theta_3} \right)}{16\left(l+L\right)^2} \right] \\
c_9 &: \dot{p_1} = \frac{dp_1}{dt} \\
c_{10} &: \dot{p_1} = -D_\theta \cdot \dot{\theta_1} \\
c_{11} &: \dot{p_2} = \frac{dp_2}{dt} \\
c_{12} &: \dot{p_2} = -D_\theta \cdot \dot{\theta_2} \\
c_{13} &: \dot{p_3} = \frac{dp_3}{dt} \\
c_{14} &: \dot{p_3} = -D_\theta \cdot \dot{\theta_3} \\
c_{15} &: \dot{p_4} = \frac{dp_4}{dt} \\
c_{16} &: \dot{p_4} = -D_\theta \cdot \dot{\theta_4} \\
&\textit{where},
\end{aligned}
$$

$$
K = \left(\frac{mR^2}{8} + \frac{J_z R^2}{16(l+L)^2} + J_w \right)
$$

The approach used to derive the above equations is outlined in Appendix A. The interested reader can go through the appendix for more details.

The structural model of the system is represented by its structure graph.Structure graph is a bi-partite graph that connects variables and constraints. For a system $S = (C, Z)$ its structure graph is given by

$$G = (C, Z, E)$$

where $E \subset C \times Z$ is the set of edges. An edge $(c_i, z_j) \in E$ iff, variable z_j appears in constraint c_i.

For the given system, the set of variables Z and set of constraints C are,

$$Z = \left\{\theta_1, \theta_2, \theta_3, \theta_4, \dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}, \dot{\theta_4}, p_1, p_2, p_3, p_4, \dot{p_1}, \dot{p_2}, \dot{p_3}, \dot{p_4}\right\}$$
$$C = \{c_1, c_2, ..., c_{16}\}$$

The structure graph is shown in figure: 4.1

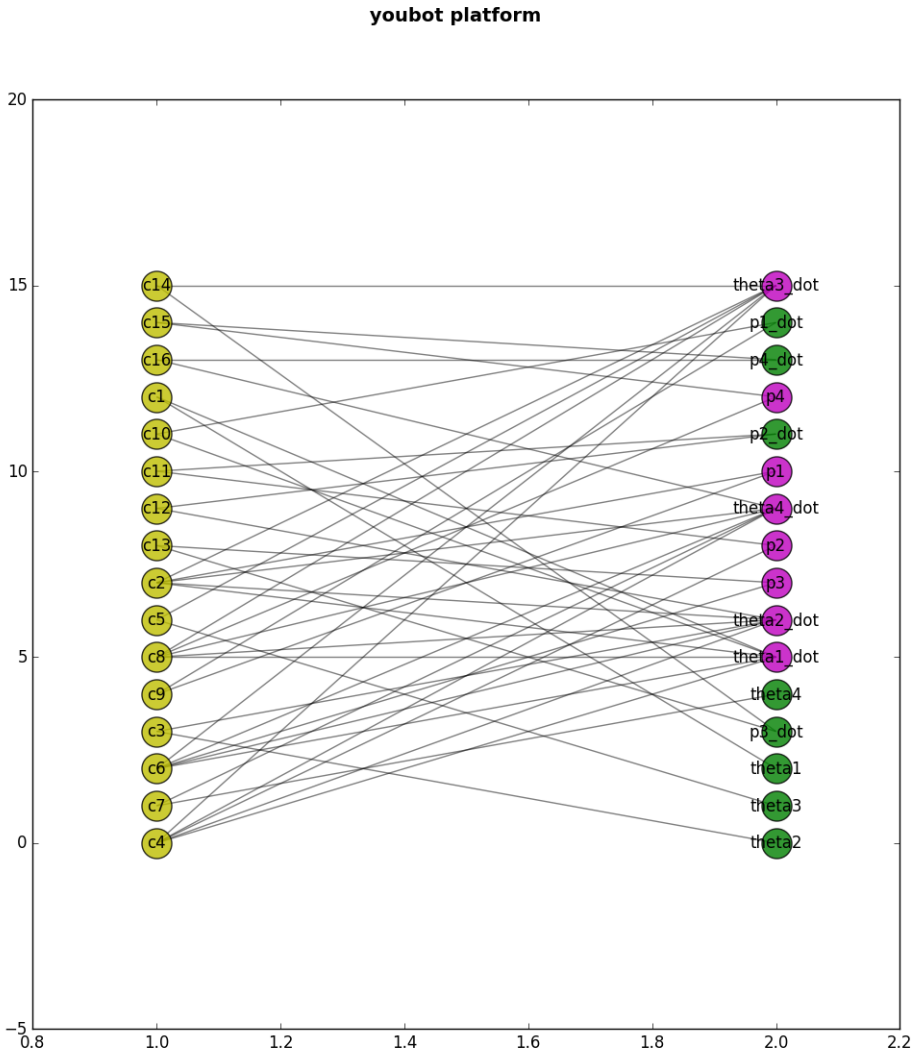

FIGURE 4.1: Structure graph of omni-directional mobile robot with mecanum wheels. This figure is generated using SaTool-python [1]

[1] https://github.com/deebuls/sa_tool_python.git

Structure graph can be represented in an algebraic manner by means of an Incidence matrix. The rows of this matrix are associated with constraints and columns with variables. A "1" in intersection of row c_i and column z_j indicates the existence of edge $(c_i, z_j) \in E$. The incidence matrix is as shown in Table:4.1.

TABLE 4.1: Incidence matrix

	Input				Output				Internal variables							
	p_1	p_2	p_3	p_4	$\dot{\theta_1}$	$\dot{\theta_2}$	$\dot{\theta_3}$	$\dot{\theta_4}$	θ_1	θ_2	θ_3	θ_4	$\dot{p_1}$	$\dot{p_2}$	$\dot{p_3}$	$\dot{p_4}$
c_1					1				1							
c_2	1				1	1	1	1								
c_3						1				1						
c_4		1			1	1	1	1								
c_5							1				1					
c_6			1		1	1	1	1								
c_7								1				1				
c_8				1	1	1	1	1								
c_9	1												1			
c_{10}					1								1			
c_{11}		1												1		
c_{12}						1								1		
c_{13}			1												1	
c_{14}							1								1	
c_{15}				1												1
c_{16}								1								1

The set of constraints and variables can be partitioned into sets of known/unknown variables and constraints.Let K, X, C_X, C_K denote the set of known variables, unknown variables, unknown constraints and known constraints respectively.

$$K = \left\{\dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}, \dot{\theta_4}, p_1, p_2, p_3, p_4\right\}$$
$$X = \{\theta_1, \theta_2, \theta_3, \theta_4, \dot{p_1}, \dot{p_2}, \dot{p_3}, \dot{p_4}\}$$
$$C_X = \{c_1, c_3, c_5, c_7, c_9, c_{10}, c_{11}, c_{12}, c_{13}, c_{14}, c_{15}, c_{16}\}$$
$$C_K = \{c_2, c_4, c_6, c_8\}$$

Then the incidence matrix for the given system can be re-written as in table: 4.2.

TABLE 4.2: Incidence matrix in terms of known and unknown variables/constraints

	Known variables								Unknown variables							
	p_1	p_2	p_3	p_4	$\dot{\theta_1}$	$\dot{\theta_2}$	$\dot{\theta_3}$	$\dot{\theta_4}$	θ_1	θ_2	θ_3	θ_4	$\dot{p_1}$	$\dot{p_2}$	$\dot{p_3}$	$\dot{p_4}$
c_2	1				1	1	1	1								
c_4		1			1	1	1	1								
c_6			1		1	1	1	1								
c_8				1	1	1	1	1								
c_1					1				1							
c_3						1				1						
c_5							1				1					
c_7								1				1				
c_9	1												1			
c_{10}					1								1			
c_{11}		1												1		
c_{12}						1								1		
c_{13}			1												1	
c_{14}							1								1	
c_{15}				1												1
c_{16}								1								1

The next step in structural analysis is to find a way to calculate the unknown variables using the constraints and known variables. The method of determining the unknown variables is called matching.

Matching is a causal assignment which associates with every unknown variable, a constraint that can be used to determine the variable. An important point to note here is that unknown variables which do not appear in a matching cannot be calculated. Applying a matching algorithm to the structural model we get,

$$M = \{c_1, c_3, c_5, c_7, c_9, c_{12}, c_{14}, c_{16}\}$$
$$U = \{c_2, c_4, c_6, c_8, c_{10}, c_{11}, c_{13}, c_{15}\}$$

where M, U denote the set of matched and unmatched constraints.

Figure:4.2 represents a structure graph of omni-directional mobile robot with mecanum wheels with maximal matching. Yellow coloured nodes denote the constraints. Violet coloured nodes denote known variables, while green coloured nodes denote unknown variables. Blue lines denote the matching of unknown variable with a constraint.

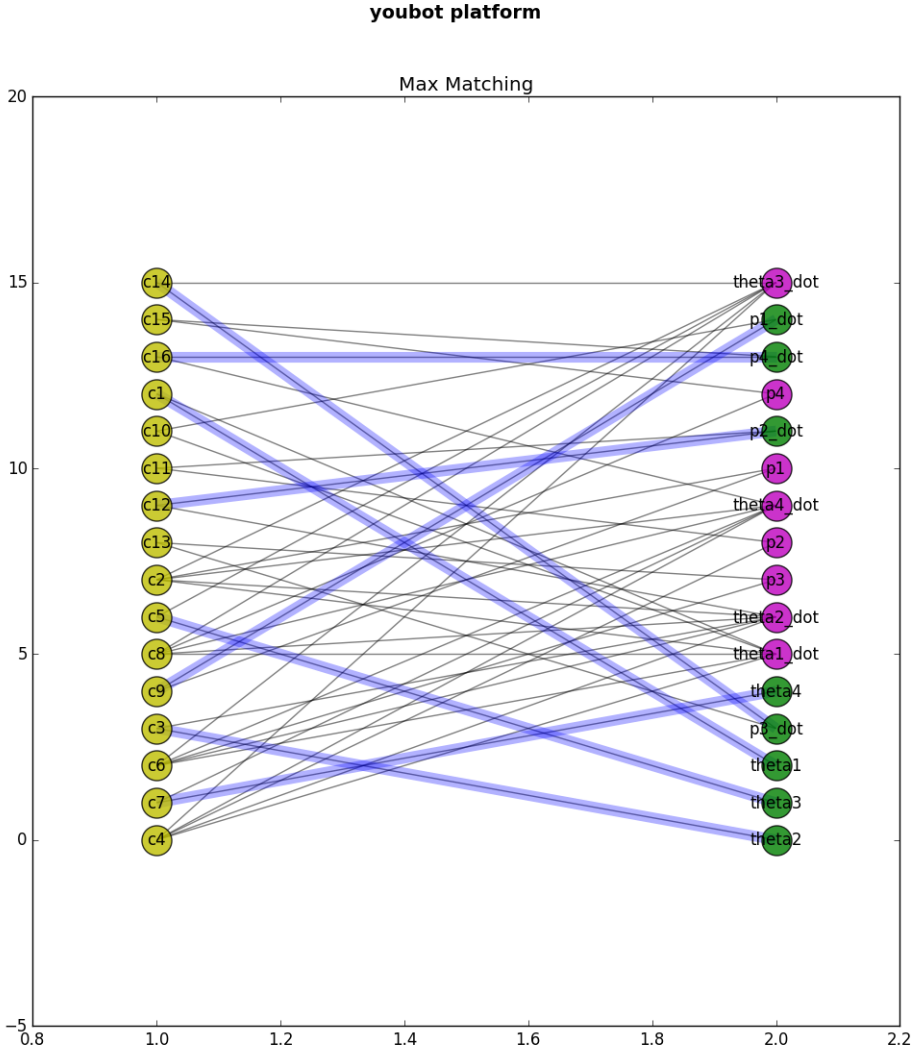

FIGURE 4.2: Structure graph of omni-directional mobile robot with mecanum wheels with maximal matching. This figure is generated using SaTool-python [2]

In the inicidence matrix, a matching is represented by selecting atmost "1" in each row and column and marking it by ① . Each ① represents an edge of matching. Table :4.3 represents an incidence matrix structural model with maximal matching.

[2]https://github.com/deebuls/sa_tool_python.git

TABLE 4.3: Incidence matrix with maximal matching

	Known variables								Unknown variables							
	p_1	p_2	p_3	p_4	$\dot{\theta_1}$	$\dot{\theta_2}$	$\dot{\theta_3}$	$\dot{\theta_4}$	θ_1	θ_2	θ_3	θ_4	$\dot{p_1}$	$\dot{p_2}$	$\dot{p_3}$	$\dot{p_4}$
c_2	1				1	1	1	1								
c_4		1			1	1	1	1								
c_6			1		1	1	1	1								
c_8				1	1	1	1	1								
c_1					1				①							
c_3						1				①						
c_5							1				①					
c_7								1				①				
c_9	1												①			
c_{10}					1								1			
c_{11}		1												1		
c_{12}						1								①		
c_{13}			1												1	
c_{14}							1								①	
c_{15}				1												1
c_{16}								1								①

Since we found a maximal matching for the structure graph, we can now add orientation to the structure graph as follows:

1. Matched constraint : The edges adjacent to matched constraint are oriented using the following rules:
   - From non-matched(Input) variable towards the constraint.
   - From constraint towards the matched(unknown) variable.
2. Unmatched constraint : All variables are used as inputs, hence all edges are directed towards the constraint.

Figure:4.3 represents an oriented structure graph . Yellow coloured nodes denote the matched constraints, while red coloured nodes denote unmatched constraints. Violet coloured nodes denote known variables, while green coloured nodes denote unknown variables. Blue lines denote the matching of unknown variable with a constraint and thick black lines denote the orientation.

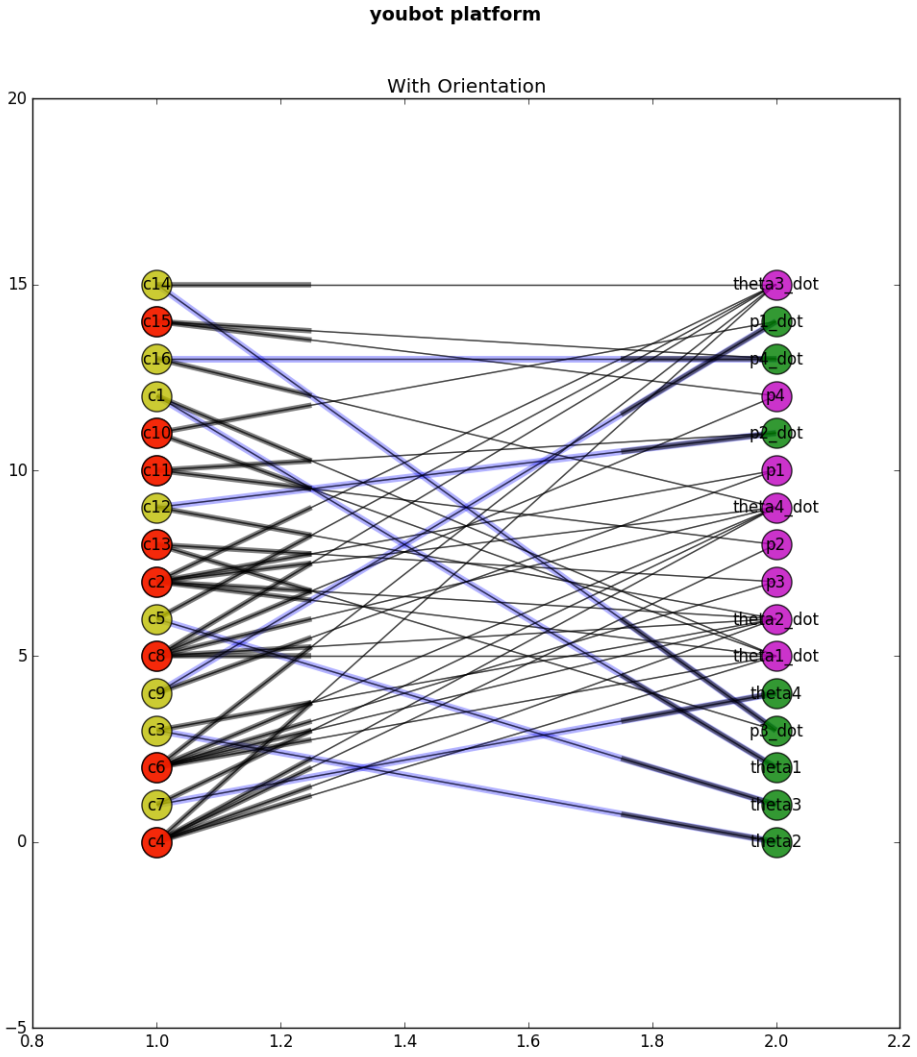

FIGURE 4.3: Oriented Structure graph of omni-directional mobile robot with mecanum wheels. This figure is generated using SaTool-python [3]

After matching has been found for a given system model, the next logical step is to determine the analytical redundancy relations, also called as parity relations. Parity relations are obtained by substituting all unknown variables in the unmatched constraints. Backtracking along an alternated chains in the matching will eliminate the unknown variables. Finally, each unmatched constraint c_i will give one parity relation r_i to be used for diagnosis. Parity relations obtained are given below:

$c_1\left(\dot{\theta_1}\right) \rightarrow \theta_1$
$c_2\left(p_1, \dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}, \dot{\theta_4}\right) \rightarrow ZERO$
$c_3\left(\dot{\theta_2}\right) \rightarrow \theta_2$

$c_4\left(p_2, \dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}, \dot{\theta_4}\right) \rightarrow ZERO$
$c_5\left(\dot{\theta_3}\right) \rightarrow \theta_3$
$c_6\left(p_3, \dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}, \dot{\theta_4}\right) \rightarrow ZERO$
$c_7\left(\dot{\theta_4}\right) \rightarrow \theta_4$
$c_8\left(p_4, \dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}, \dot{\theta_4}\right) \rightarrow ZERO$
$c_9\left(p_1\right) \rightarrow \dot{p_1}$
$c_{10}\left(\dot{\theta_1}, \dot{p_1}\right) \Longrightarrow c_{10}\left(\dot{\theta_1}, c_9\left(p_1\right)\right) \rightarrow ZERO$
$c_{11}\left(p_2, \dot{p_2}\right) \Longrightarrow c_{11}\left(p_2, c_{12}\left(\dot{\theta_2}\right)\right) \rightarrow ZERO$
$c_{12}\left(\dot{\theta_2}\right) \rightarrow \dot{p_2}$
$c_{13}\left(p_3, \dot{p_3}\right) \Longrightarrow c_{13}\left(p_3, c_{14}\left(\dot{\theta_3}\right)\right) \rightarrow ZERO$
$c_{14}\left(\dot{\theta_3}\right) \rightarrow \dot{p_3}$
$c_{15}\left(p_4, \dot{p_4}\right) \Longrightarrow c_{15}\left(p_4, c_{16}\left(\dot{\theta_4}\right)\right) \rightarrow ZERO$
$c_{16}\left(\dot{\theta_4}\right) \rightarrow \dot{p_4}$

Signature matrix or dependability matrix represents a boolean mapping of constraints onto residuals. Signature matrix for the above obtained parity relations is given below:

$$\begin{pmatrix} \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \\ r_7 \\ r_8 \end{pmatrix} \leftarrow \begin{pmatrix} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{16} \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

The signature matrix can be used to analyse the structural detectability and isolability properties of the given system.In structural analysis, a fault is defined as the violation of constraint. A fault is structurally detectable if and only if it has a non-zero boolean signature in some residual.

For the given system,constraints other than c_1, c_3, c_5, c_9 have non-zero boolean value in the signature matrix given above,the violation of constraints other than c_1, c_3, c_5, c_9 are structurally detectable.

A fault is structurally isolable if and only if it has a unique signature in the residual vector.

Considering the signature matrix given above, only four columns are independent:c_2, c_4, c_6, c_8, hence only these constraints are structurally isolable.

TABLE 4.4: Constraints structurally detectable and isolable

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}
1	n	i	n	i	n	i	n	i	d	d	d	d	d	d	d	d

Table:4.4 summarizes the structural detectability and isolability properties for the omni-directional mobile robot with mecanum wheels. d and i denote structural detectability and isolability, n denotes a constraint that cannot fail.

4.2 Control Design

In this section, the low level controller that will be implemented on the dynamic model derived in chapter 3 will be explained. As mentioned previously in the state of the art chapter 2, there are two ways that the controller can be approached. One way would be to have a centralized solution where all four wheels of the omnidirectional platform would be commanded under one controller. The other way would be by using a decentralized solution, which is more common among low level controllers. The idea of a decentralized controller is to individually control each wheel of the omnidirectional platform. Since implementing a decentralized solution is simpler and as efficient as a centralized solution regarding the use case of this research, it was chosen as our method of control.

Also, after deriving the dynamic model of the system in the previous chapter, it was obvious that the system is linear and therefore, we looked into linear controllers. The most common method of implementing a decentralized controller is by using a PID controller on each wheel of the omnidirectional platform.

A proportional-integral-differential controller is a linear feedback controller which works by calculating the error value between a reference value and the current output values of a system. For our case, if we look at the state-space representation of our system, equation 3.22, we could notice that the input to our system is τ, which is the torque input to each wheel. To implement a controller on the current system, the input should be changed to a controlled input.

The ideal equation for a PID controller is as follows:

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de}{dt} \tag{4.1}$$

where $u(t)$ is the controlled input, $e(t)$ is the error signal, which in our case will be $(\dot{\theta_r} - \dot{\theta_c})$ where $\dot{\theta_r}$ is the reference angular velocity and $\dot{\theta_r}$ is the current angular velocity produced by the system. k_p, k_i and k_d are the proportional, integral and derivative gain.

By replacing the input in the state space representation to equation 4.1, we get a controlled system. The proportional term in equation 4.1 corresponds to the change that will be produced in the controlled output based on the current value of the error. The integral term in the equation affects the output by accumulating the error over time. In other words, if a large error has been accumulating in the system, the reaction of the controller would be stronger to the current error. Finally, the derivative term is responsible for predicting the error that might occur in the system based on the gradient of the system over time.

When analyzing a system, some characteristics are of importance, such as rise time, overshoot, settling time, and steady state error. Rise time is the time taken for the system to reach the desired output level for the first time. Overshoot is the maximum output the system produces while trying to reach its steady state. Settling time is the time taken for the system to reach steady state. Last, steady state error is the difference between the reference output and the output of the system after reaching steady state. The table below shows the effects of the controller gain on the system:

TABLE 4.5: Effect of PID controller gains on the system [34]

Parameter	**Rise Time**	**Overshoot**	**Settling Time**	**Steady State Error**
Kp	Decrease	Increase	Small change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Small change	Decrease	Decrease	No effect

The selection of the gains of the PID controller can be made in several ways, such as, Ziegler-Nichol's, or relay feedback, or other methods. The selection process and characteristics of the system will be discussed later in the experimental evaluation chapter, chapter 5.

Chapter 5

Experimental Evaluation

In this chapter, the experiments performed for this project are explained. The aim of this research is to develop a mathematical model which correctly describes the behaviour of an omnidirectional platform, and then, implement a controller which is suitable for the developed model and that can simulate the behaviour of an actual controller.

In order to do so, we derived a kinematic and dynamic model as described in chapter 3. For these models to be validated, they should behave the same as an actual robot platform would behave when given the same input. Clearly, there would be some margin of error since the models assume an ideal environment scenario, while in reality, the environment plays a part in a dynamic system. However, if this error is within reasonable limit, we can consider the model accurate to a certain percentage or degree.

As for the controller that is to be implemented, it should behave similarly to an actual controlled omnidirectional platform. The controller should also be modifiable so that it can simulate the response of the system if its parameters were to change.

There are two main experiments which are performed. The first set of experiments were conducted to validate the kinematic and dynamic model of the physical system which was modeled. The second set of experiments were performed to evaluate a velocity controller which was implemented on the dynamic model and to compare it to the existing system.

This chapter is divided as follows: the first section will discuss the experimental setup, including detailed explanations of what hardware/softwre was used in the process of obtaining the results. The second section will discuss the results obtained in detail and how they were derived. The results of the experiments can be reproduced by following the steps of each experiment.

5.1 Experimental Setup

In this section, a detailed explanation will be given on the used omnidirectional platform "Kuka youBot", the softwares used in the experiments and how our system was set up, and the environment where we ran our experiments.

5.1.1 Omnidirectional Platform - youBot

The youBot is a omnidirectional platform consisting of 4 omnidirectional wheels developed by KUKA. It is developed mainly for research and educational purposes. The youBot allows the users to access nearly all levels of control of the available hardware, which was optimal for our use case.

FIGURE 5.1: youBot platform [1]

5.1.1.1 Technical Specifications

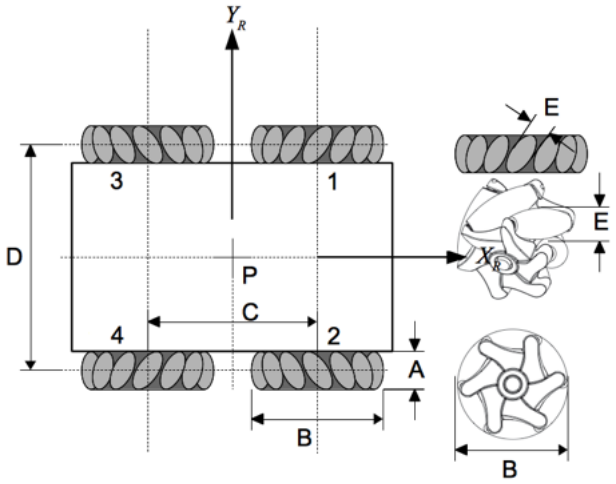

FIGURE 5.2: youBot base geometry [2]

[1] https://static.generation-robots.com/5379-thickbox_default/kuka-youBot-mobile-platform.jpg
[2] http://www.youBot-store.com/wiki/images/4/40/MeasurementData.png

The youBot is made up of four omnidirectional wheels with the angle $\gamma = 45$. The dimensions of the platform which are shown in Fig. 5.2 are as follows:

- A = 74.87 mm
- B = 100 mm
- C = 471 mm
- D = 300.46 mm
- E = 28 mm

The motors that run the wheels are brushless DC motors from the company Maxon Motors of type EC 45 flat. There are four motor controller boards, 1 for each wheel, from the TMCM series by Trinamic. Some important parameters regarding the motors are given:

- Nominal Voltage = 24 V
- Nominal Current = 2.32 A
- Nominal Torque = 82.7 mNm
- Terminal Inuctance = 0.573 mH
- Terminal Resistance = 0.978 Ω
- Rated Speed = 5250 RPM
- Encoder counts per Revolution = 4000

The youBot is also equipped with an internal PC which is used to communicate and operate the platform. The specifications of the internal PC of the youBot are given below:

- CPU: Intel Atom D510 Dual Core 1.66 GHz
- RAM: 2 GB
- Hard Drive: 32 GB SSD
- Ports: USB, VGA, LAN
- DC Input: 12 V

5.1.1.2 youBot Driver

The youBot Driver is the software provided by the youBot Store for operating the mobile platform. The drivers are used to provide the necessary information from the motors and sensors on the youBot.

In abstract terms, the drivers are one layer above the firmware of the system, which communicate directly with the hardware. The firmware runs the youBot motors, which is structured in a way that can allow for different control modes, such as position, velocity or current control. From the firmware manual provided by TMCM, the motors have the following cascaded controller structure:

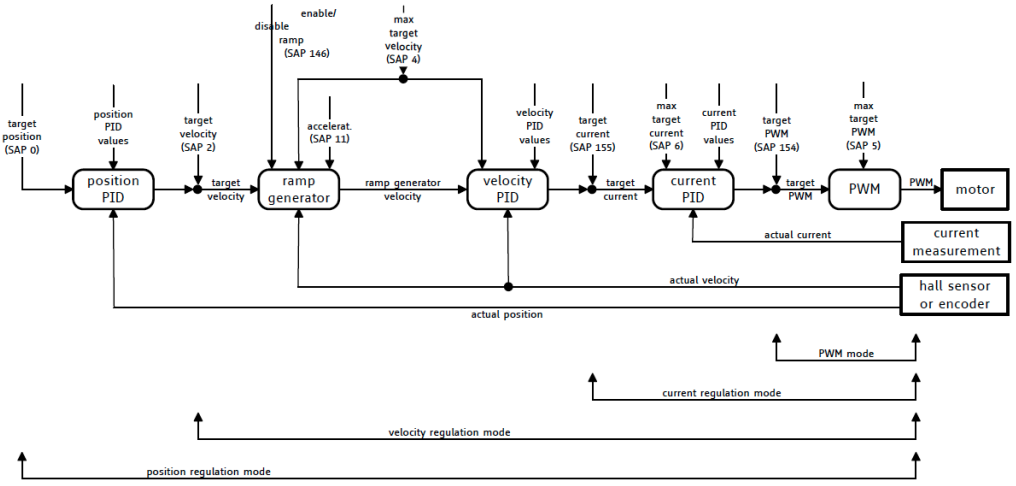

FIGURE 5.3: Motor firmware control structure [35]

The drivers(youBot API) then allow the user to access the firmware functionalities. The youBot API is open source and is written in C++. It has the advantage that it is framework independent. The aforementioned structure is useful for our experiments, since we want to test the youBot's behaviour on different control points.

5.1.2 Software

Our experiments required the validation of derived mathematical models and evaluation of a velocity controller, which meant that some comparison was to be made between the actual robot and the derived models, and the controller characteristics were to be examined and tested. In order to do so, some framework's and software's were used to facilitate this process.

For operating the youBot and logging the data, Robot Operating System (ROS) was used as the underlying framework. For validation purposes and evaluation, Matlab and Simulink were used.

5.1.2.1 Robot Operating System - ROS

ROS is a framework that is used widely among different robotic platforms and serves the purpose of simplifying the process of writing robot software. It consists of different tools and libraries that helps doing so.

Since the provided drivers use publishers/subscribers to send and receive messages, for example when we needed to send current commands to the wheels, new publisher and subscriber messages were created and used.

5.1.2.2 Matlab & Simulink

Matlab & simulink are powerful softwares that can be used for system design and analysis as well as many more applications. For our project, it was used for simulating the dynamic model and plotting its behaviour under different control scenarios.

From the derived kinematic (3.11) and dynamic (3.19) models from chapter 4, the following parameters were used and stored in matlab:

- R = 0.0475m (radius of wheel)
- m = 23.58 Kg (mass of youBot)
- L + l = 0.385
- J_w = 1.35 $Kgmm^2$ (moment of inertia of wheel)
- J_z = 5.5 $Kgmm^2$ (moment of inertia of youBot around Z axis)
- gear ratio = 0.0387
- torque constant = 0.0335

Some of the parameters above were measured such as the mass of the youBot. Other parameters were from the youBot drivers provided and the youBot's specifications webpage[3]. The parameters were used in creating the state space model in matlab and simulink.

[3]http://www.youBot-store.com/wiki/index.php/youBot_Detailed_Specifications

Since we wanted to test the youBot under different control set points, there were two basic models that were used in the validation and control evaluation experiments. The first is an open loop model of the system. This is where we try to validate the developed dynamic model by comparing its output with the output of the actual robot. The dynamic model takes in 4 inputs which are commanded torques for each wheel and produces 4 velocities to each wheel.

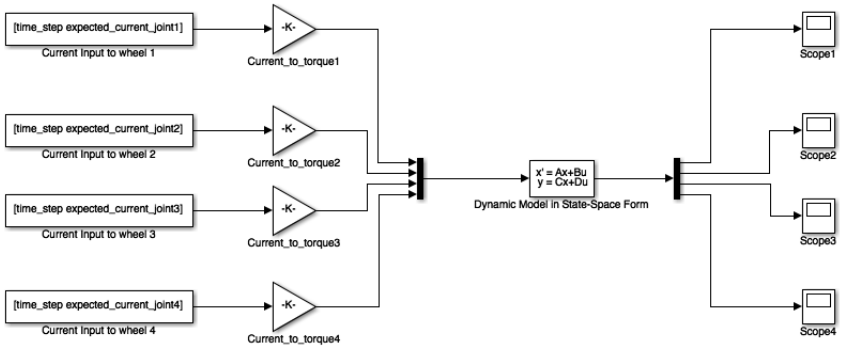

FIGURE 5.4: Open loop dynamic model on simulink

From Fig.5.4 we can see that the input to the model is current, since the lowest level of control allowed by the youBot drivers is current control as shown in Fig.5.3. However, a conversion from current to torque can be made to adapt for this model which is equivalent to: "torque constant/gear ratio". This conversion was also found in the driver files of the youBot. The output of the model tested for different inputs will be shown in the results section of this chapter.

The second model is the controlled version of the open loop model where a PID controller is applied to each wheel. This model was developed in order to evaluate different controller behaviours of the youBot using the derived dynamic model. In other words, if we wanted to see how the youBot would behave using different control parameters, we could simulate the behaviour first before deploying the parameters to the actual platform.

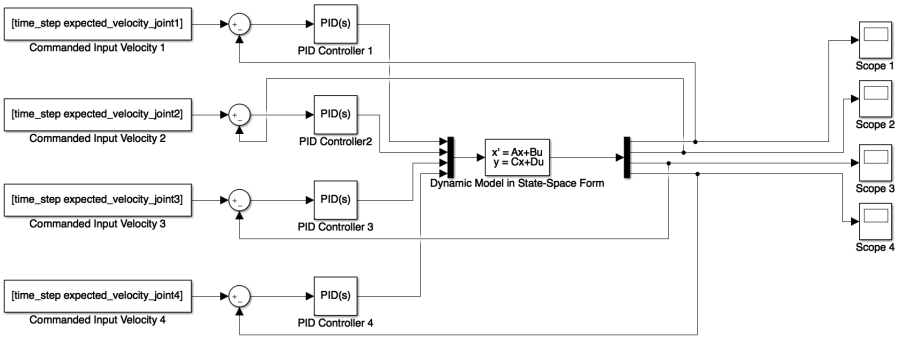

FIGURE 5.5: Closed loop controlled model on simulink

From Fig. 5.5 we could see that the difference between this model and the open loop model is that now the input to the system is a controlled input from each PID controller.

5.1.3 Environment setup

Since our experiments require running the youBot under different control conditions, a safe environment is needed where the youBot can be monitored and where accidents can be avoided. For that, the AICISS (Accelerating the Innovation Cycle in Service Robotics) lab at our university, Bonn-Rhein-Sieg University of Applied Sciences, was an optimal choice for conducting our experiments.

5.1.3.1 AICISS Lab

The AICISS lab is a lab located at our university which has the purpose of monitoring and logging the behaviour of the youBot for long periods of time. The mobile platform has an area of 4m x 7m to move around in. Since the robot can run for long periods of time, a direct input of power is always connected to the youBot during operation to insure a constant power supply. There are 3 cameras that are mounted on the ceiling of the lab which are used to provide position and orientation information of the youBot via markers attached to it. Both the cameras and the robot are connected to a central PC where the user can command the robot and log information from experiments.

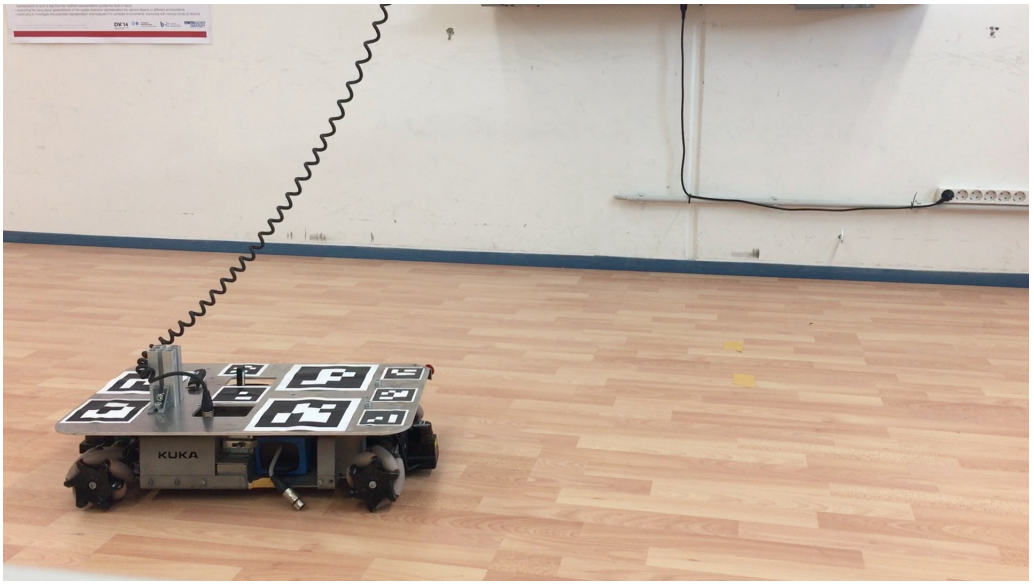

FIGURE 5.6: Kuka youBot with markers attached in the AICISS lab

5.2 Results

In this section of the chapter, the results obtained from testing are shown and explained in detail. As mentioned previously, there are mainly two sets of experiments which were conducted. The first set was to validate the kinematic and dynamic model that was developed in chapter 3, and the second set of experiments were performed to evaluate the controller implemented on the dynamic model. The results shown in this section were produced by using matlab. More results can be found in the appendix B section of this report.

5.2.1 Kinematic Model Validation

In our first experiment, we wanted to validate if our derived kinematic model matches the youBot. Since the youBot's driver generates individual wheel speeds based on commanded X, Y and angular velocities, we wanted to compare between the actual youBot's generated velocities and the kinematic model output to see if our kinematic model generates similar values. In order to do so, we ran the youBot under its default mode which was velocity control mode in random motion, gathered the inputs and passed them to the kinematic model. The output of the kinematic model which converted the commanded X, Y and angular velocities to individual wheel velocities was then plotted using matlab and shown below in Fig. 5.7. The Kinematic model which was dervied in chapter 3 is also given for clarification.

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \end{bmatrix} = \frac{1}{R_W} \begin{bmatrix} 1 & -1 & -(l+L) \\ 1 & 1 & (l+L) \\ 1 & 1 & -(l+L) \\ 1 & -1 & (l+L) \end{bmatrix} \cdot \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \tag{5.1}$$

As seen from Fig. 5.7, the plot shows that the velocity values match exactly for wheels 2 and 4. However, for wheels 1 and 3, it was found that the values are exact opposites. After the revision of the kinematic model and investigation of the youBot's driver files, it was found that there was a negative sign for both wheels 1 and 3 in the driver files since the authors of the drivers were trying to follow a structure that suits the controllers.

In order for us to compensate for this, we also added a negative sign to wheels 1 and 3. The corrected values are shown in the plot Fig. 5.8. To show this on the plot clearly, the commanded velocities by the drivers were incremented by 15 rad/s. Evidently, the kinematic model matches exactly the youBot and there is no need for further validation.

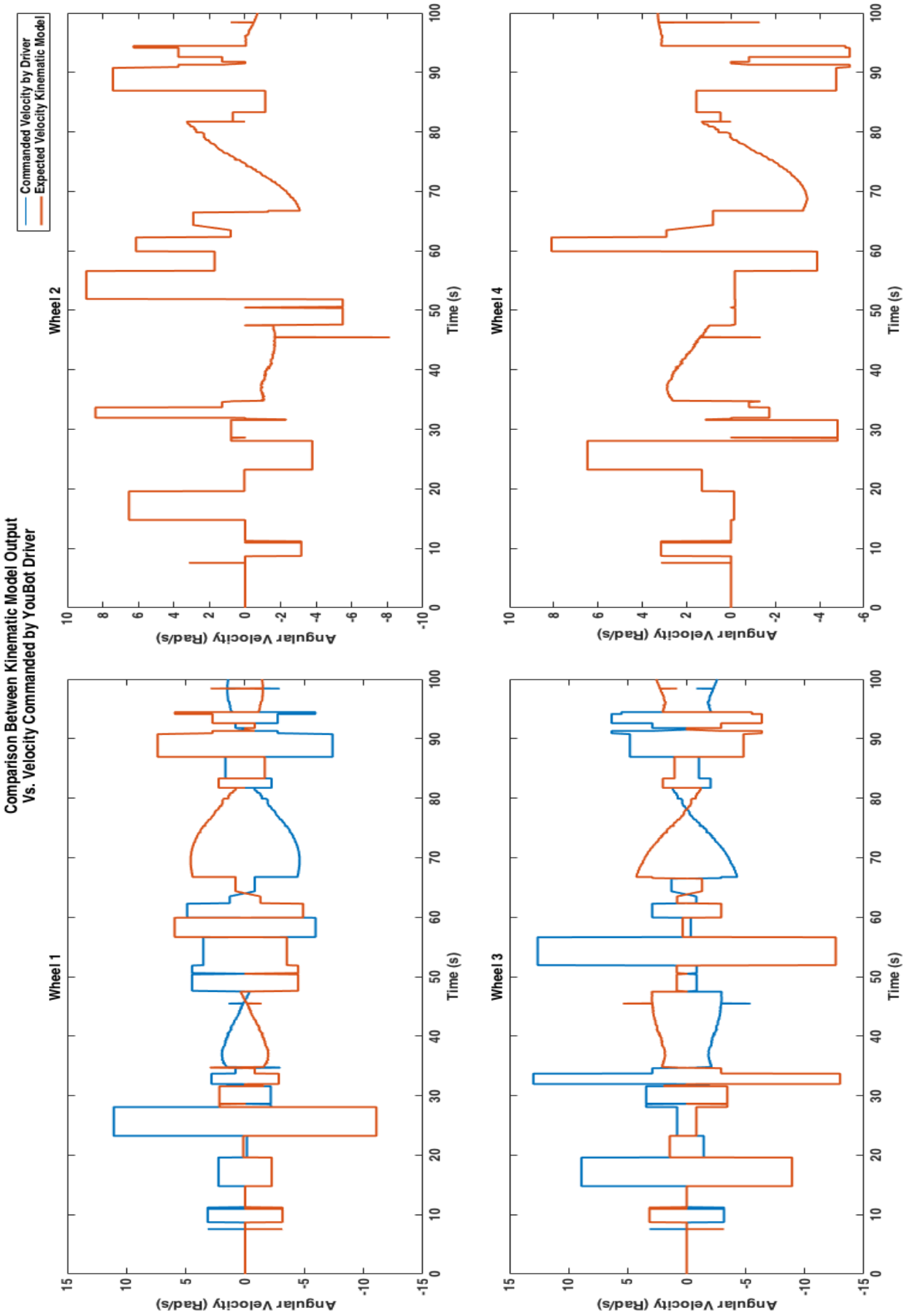

FIGURE 5.7: Comparison between kinematic model velocity prediction vs. commanded velocity by youBot drivers to each wheel.

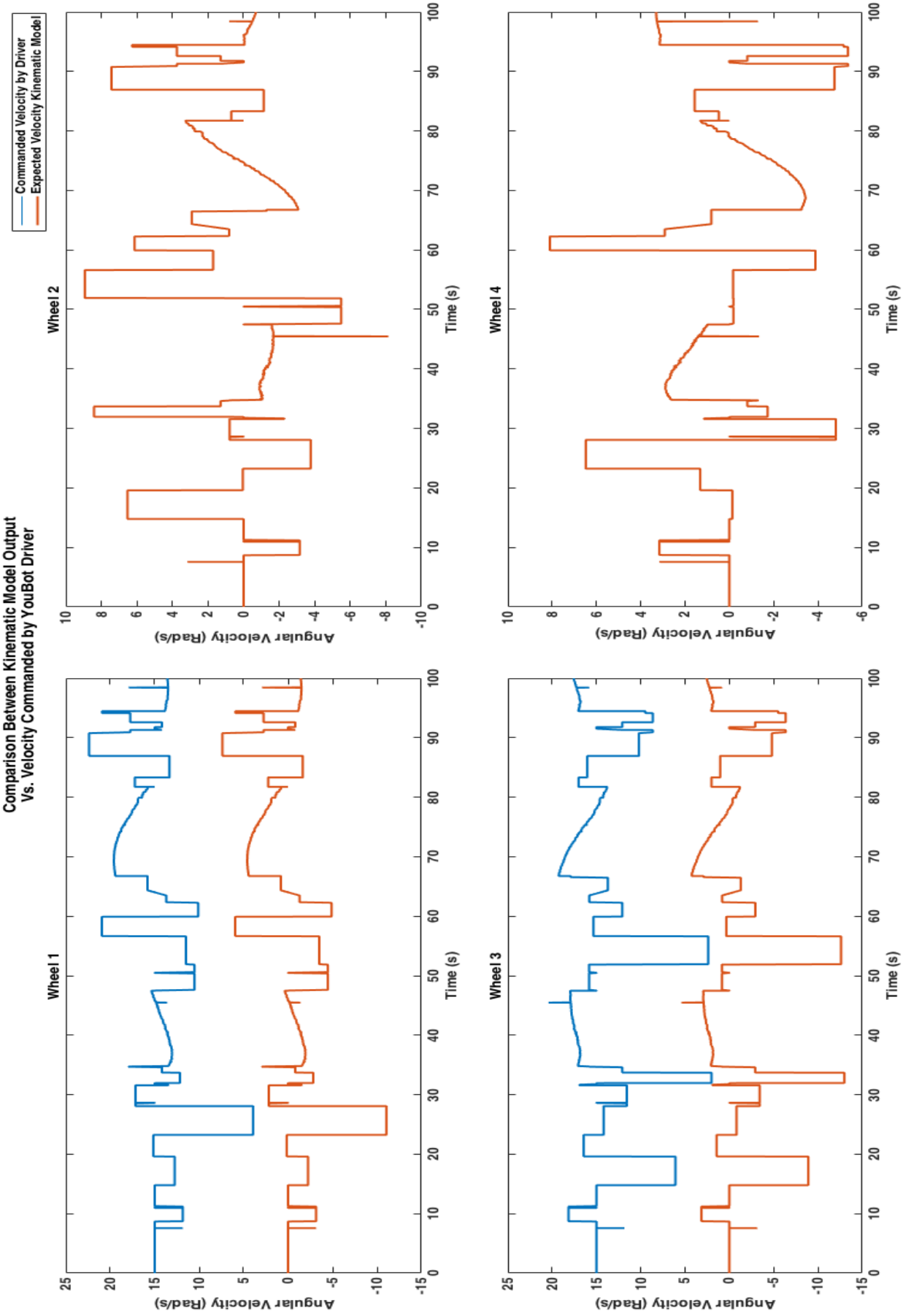

FIGURE 5.8: Comparison between kinematic model velocity prediction vs. commanded velocity by youBot drivers to each wheel after correction.

5.2.2 Dynamic Model Validation

Our next step was to validate the derived dynamic model. In order to do so, some necessary tests and experiments were conducted. Since our dynamic model is an open loop model of the system, meaning that the input was electric current to the motors (the control mode of the drivers were modified to suit this part of our experiments).

To begin, the response of the system was of interest to us so that we could see how our model behaves. Using the parameters given in the experimental setup section, the first test that was conducted was to see the step response of the system. Our system takes in four inputs (see Fig. 5.4) which are electric currents, and produces individual wheel velocities in (rad/s).

Since there are four inputs to the system and four wheels, each input would have an effect on all four wheels if the robot was on the ground (i.e. presence of friction on all wheels). However, since we wanted to see the response of the system without taking into consideration the surface the robot would be on yet, the value of the wheel's viscous friction coefficient was set to zero, therefore having no effect on any of the wheels.

The step response of all four wheels to their corresponding inputs are all identical to the shown Fig. 5.4. As seen in the plot, the system seems to be over damped with no overshoot or oscilation. The rise time, settling time and steady state are shown in the plot. The full plot showing the effect of all inputs on all wheels can be found in the appendix B setion of this report.

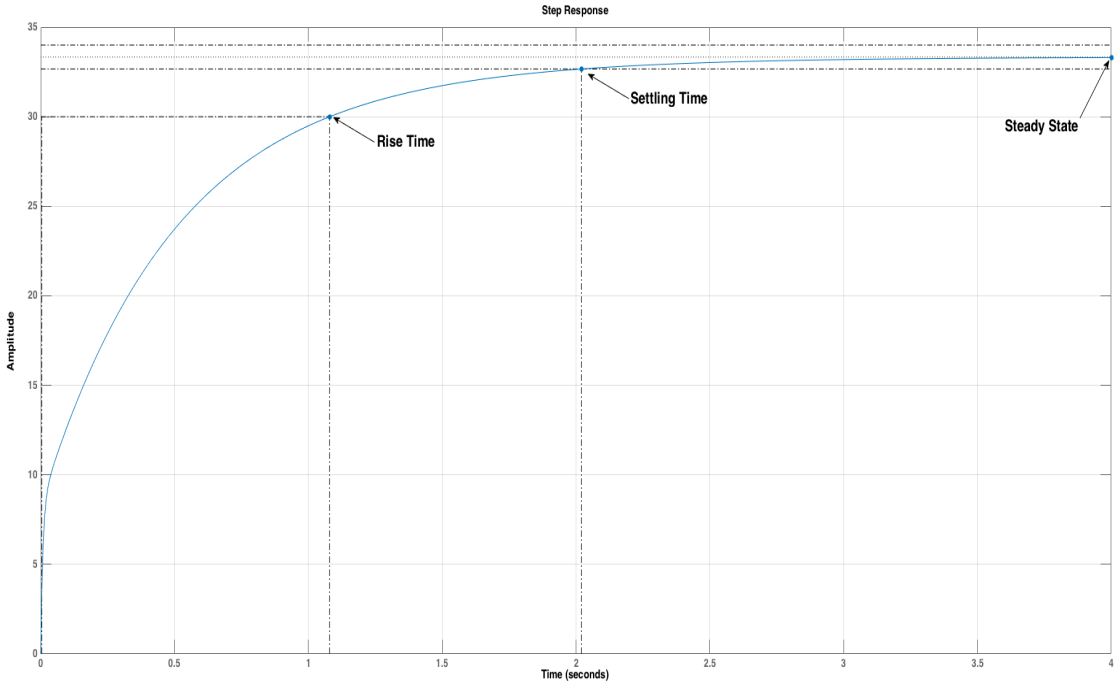

FIGURE 5.9: Open loop response of dynamic model

Next, we wanted to find the most appropriate coefficient of wheel's viscous friction to apply to our model. Since all our tests will be conducted in the AICISS lab, we are trying to find one value which corresponds to the friction between the labs surface and the omnidirectional wheels.

To find the wanted value, we conducted a simple test using the youBot where we commanded it to move forward. The data of that test was logged and then, using matlab, the value of the coefficient of friction was set between values from 0.01 to 0.045 with an increment of 0.005. Afterwards, the input of the test was passed to our dynamic model on simulink and the output was compared to that of the actual test.

To select the best value for the coefficient of friction, we compared the outputs of both the youBot and our developed model and calculated the root mean square error between them. The values were then plotted in Fig. 5.10 and as seen, the best value for the coefficient of friction was 0.035. The plot in Fig. 5.11 shows both the output of our developed model and the output of the youBot. As demonstrated, the data from the youBot is noisy, and, in the instance between the seconds 70-80, there is a small external force acting on the robot opposite to its direction. This is because we were trying to slow down the youBot since it was running in open loop control mode and there was no feedback to the robot. Therefore we had to ensure the saftey of the robot ourselves. However this, showed us that our model responds correctly to its input and with very low error in comparison to the actual youBot.

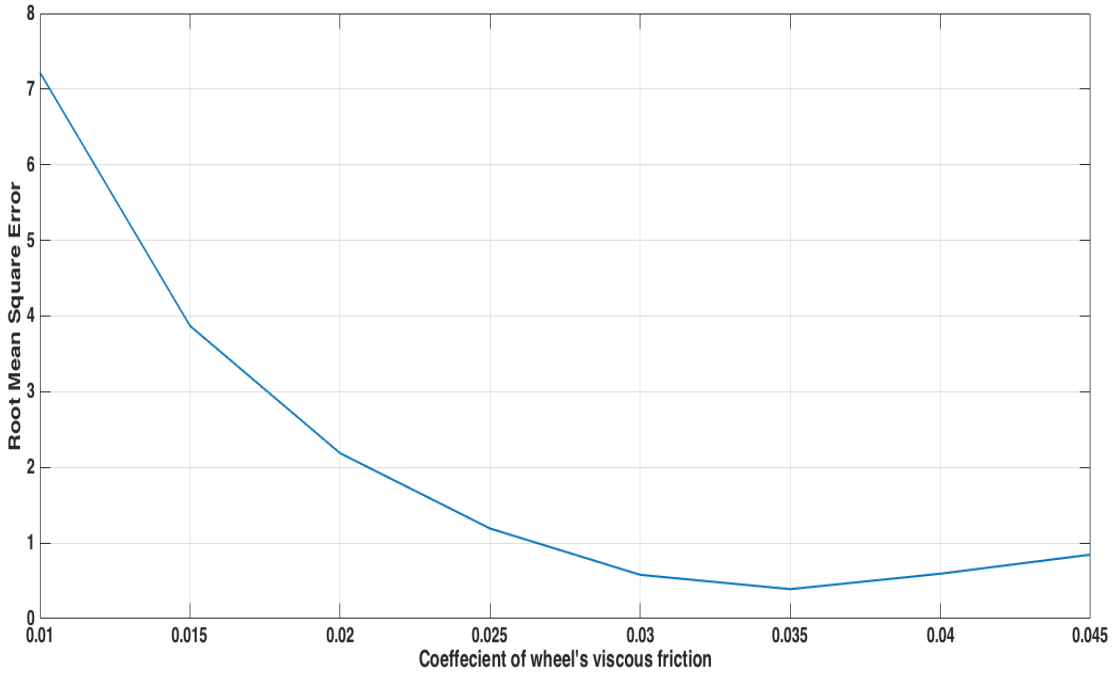

FIGURE 5.10: Root mean square error vs. coefficient of wheel's viscous friction

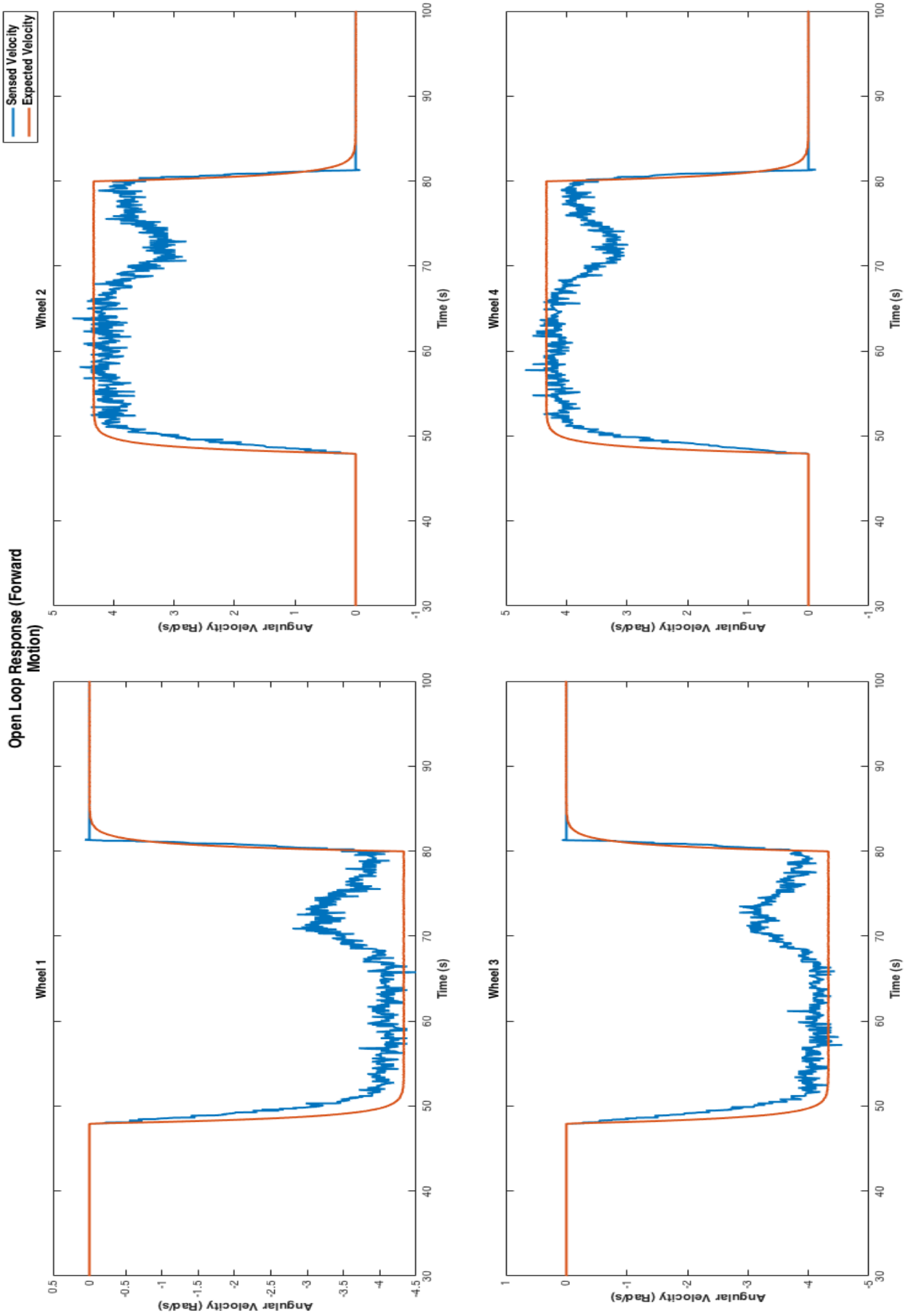

FIGURE 5.11: Open loop response of dynamic model: forward motion

To proceed with the validation testing, more motions needed to be tested in order to try to cover as much of the youBot's possible motion. Figures 5.12 to 5.14 show the different tests performed on the youBot and the comparison to the output of our dynamic model.

Fig. 5.12 shows a forward random motion. The dynamic model follows the peaks of the youBot's sensed velocity. However, we could notice that there was some error between both outputs.

Fig. 5.13 shows another test performed on the youBot and its comparision with the dynamic model output. In that test, the model also follows the peaks. Yet, we could notice that the youBot's sensed velocity takes more time to reach 0 rad/s from peaks.

Fig. 5.14 shows another test that we conducted where we tried to incorporate more motions. The test shows that there still is some big error between our model and the sensed velocity. This led us to further investigate and to find out why there was constantly an error.

After investigation, we found that the firmware that was running on our drivers was an older version than the one currently released by the youBot's developers. This led us to try to see if the current controllers were faulty in the first place. For that, we wanted to test if the youBot would behave normally once the wheels were left to rotate freely. We then commanded the youBot different motions and a fault was immediately recognized. Wheels 2, 3, and 4 were faulty in that when commanded by an input current, they would follow the correct behaviour until we stopped the commanded current. After that the wheels would go on in the same motion until we had to kill all the processes running on the youBot's PC. This behaviour is shown in Fig. 5.15 where it is clearly visible that the faulty wheels do not follow the correct trajectory.

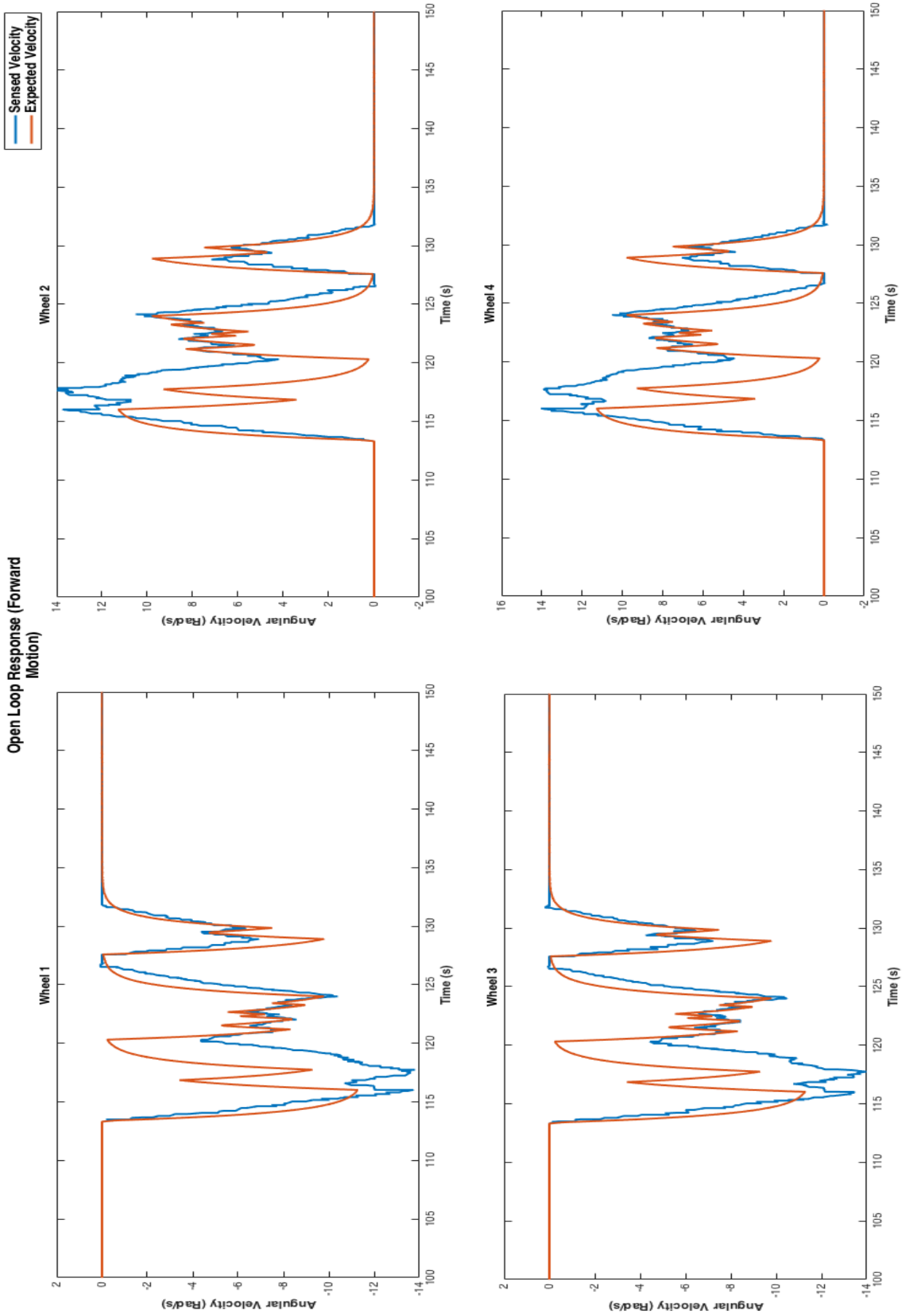

FIGURE 5.12: Open loop response of dynamic model: forward motion run 2

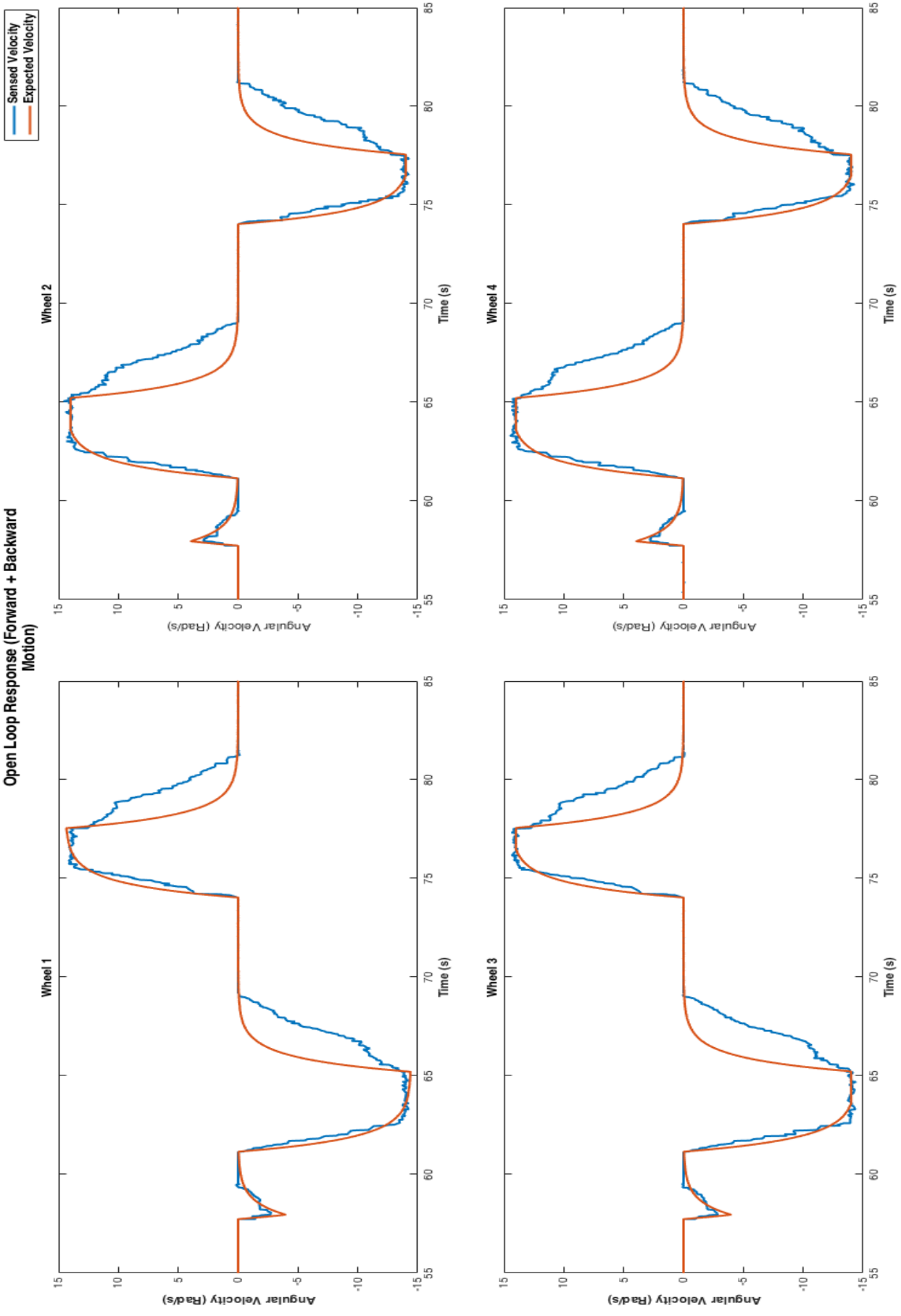

FIGURE 5.13: Open loop response of dynamic model: random forward + backward motion

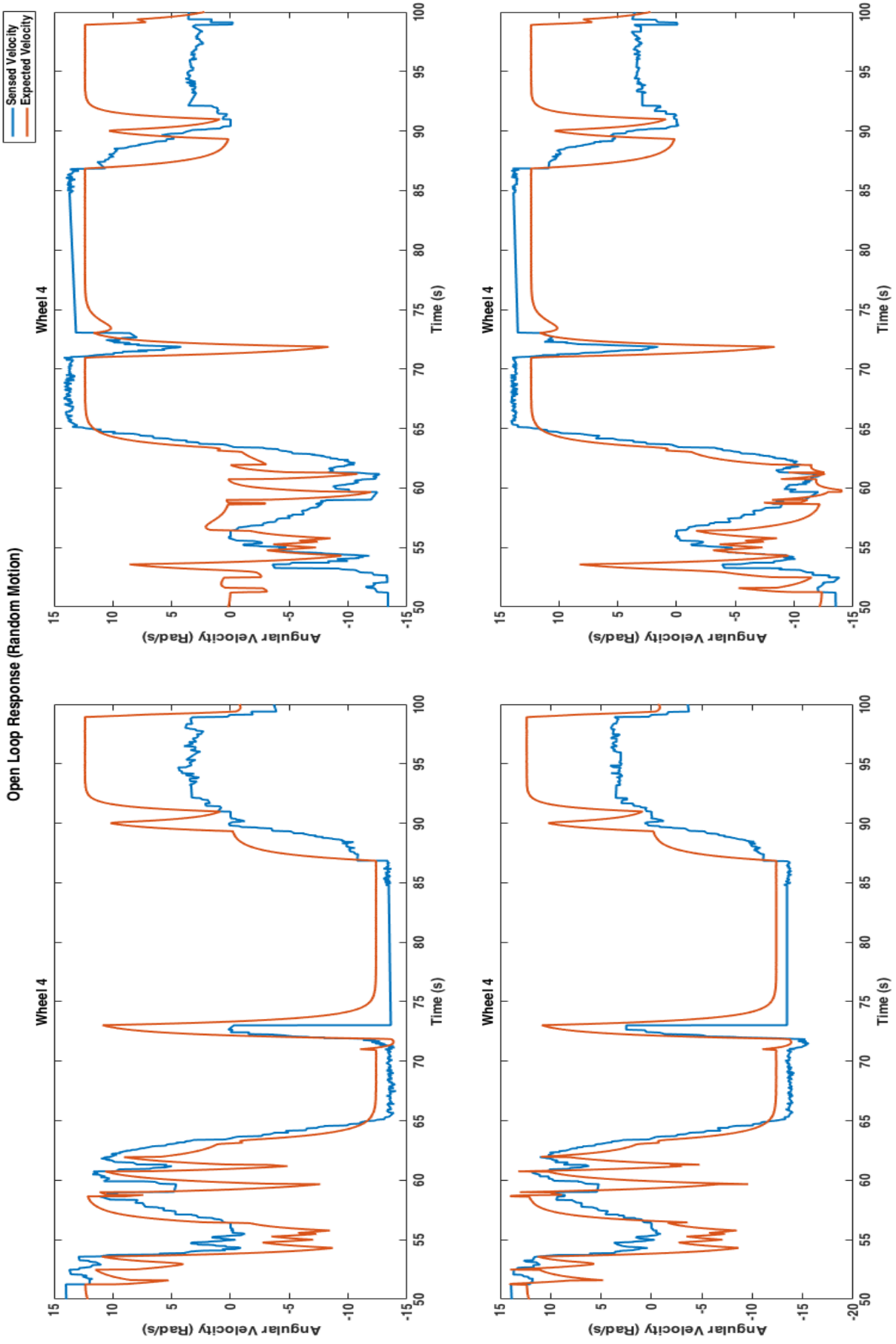

FIGURE 5.14: Open loop response of dynamic model: random motion

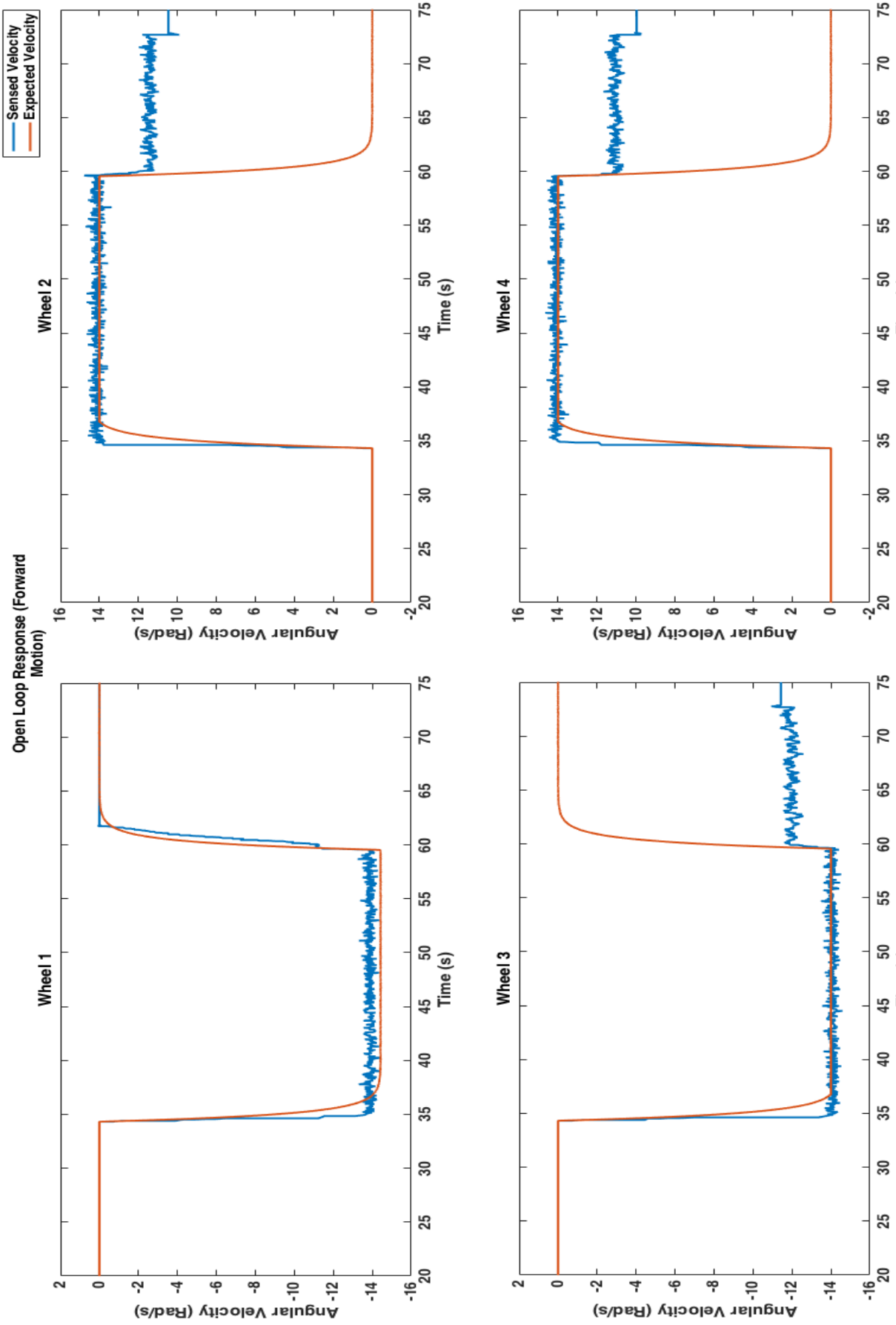

FIGURE 5.15: Open loop response of dynamic model: forward motion showing faulty behaviour

After finding out about this malfunction, it was clear that our model was working correctly, and as seen, we could detect faults at this early stage. Still, we still wanted to validate our model using a non-faulty robot platform. Fortunately, we had another youBot available, which was working correctly and had the latest firmware version running on its controllers. Figures 5.16 to 5.18 show the results of some of the tests ran on the working youBot, and their comparison with our dynamic model predictions.

Fig. 5.16 and 5.17 show the behaviour of the youBot when being commanded forwards and backward motions and its comparison with our dynamic model prediction. As seen in the plots, the dynamic model predictions are extremely similar to the youBot's behaviour. We could also notice that there is less noise in the readings from the youBot.

Fig. 5.18 shows a test where the youBot was commanded to move in random motion. The plot also shows the comparison with the dynamic model output. It is visible that the developed dynamic model can correctly predict the motion of the youBot given the same inputs.

From the previous tests and results, we concluded our experiments for validating both the kinematic and dynamic models. We could see from the conducted tests that the dynamic model could predict with a very high accuracy. The error between the model predictions and the youBot's output is less than 1% . The model also behaves symmetrically when rising to a peak and falling back to its previous state as seen in many of the plots which indicates its linearity. To deal with the noise, a linear or non-linear filter could be added to the data coming from the youBot's sensors in order to enhance readings. There are also many other ways which could be implemented to enhance the dynamic model which will be discussed in chapter 6.

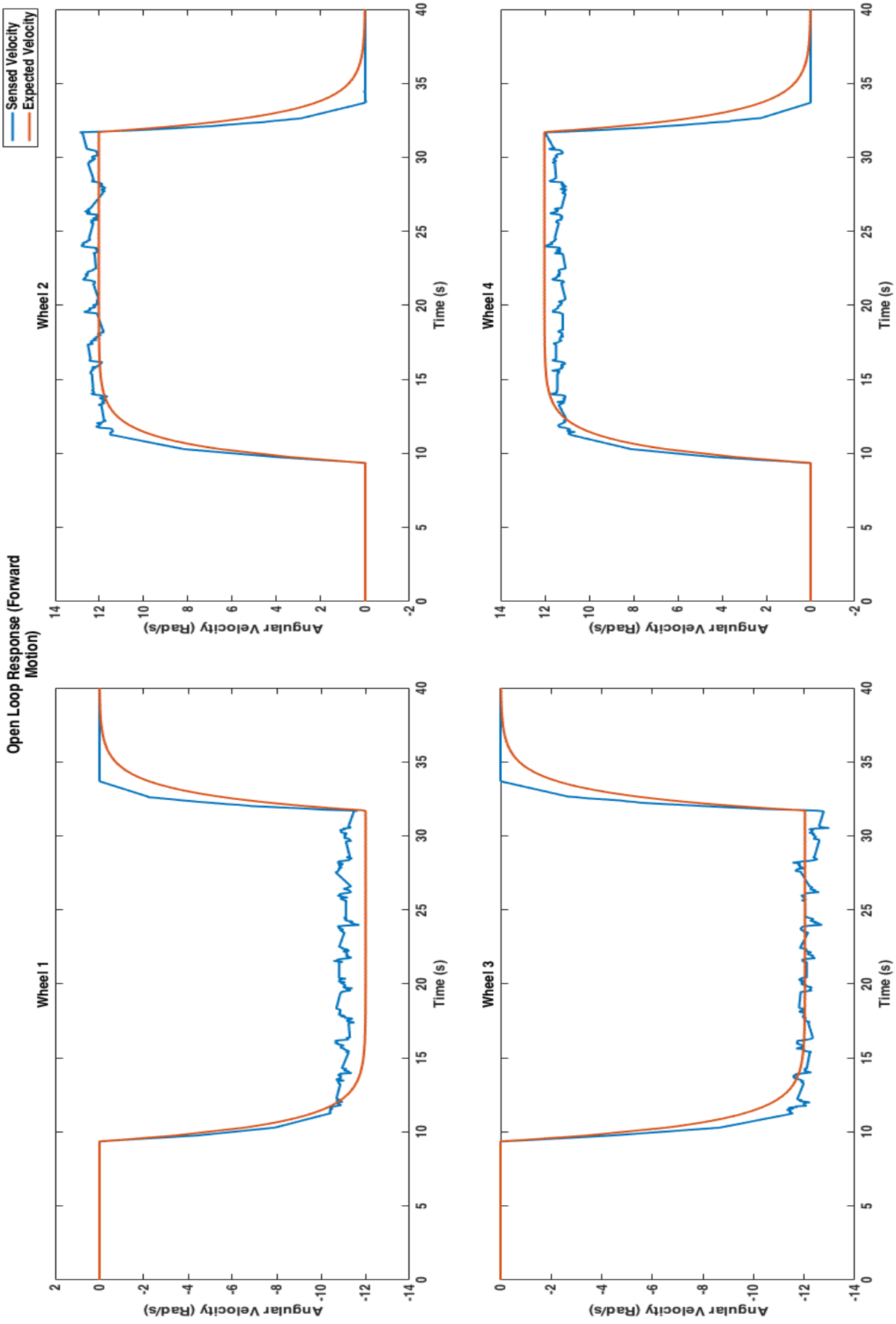

FIGURE 5.16: Open loop response of dynamic model: second youBot's behaviour forward motion

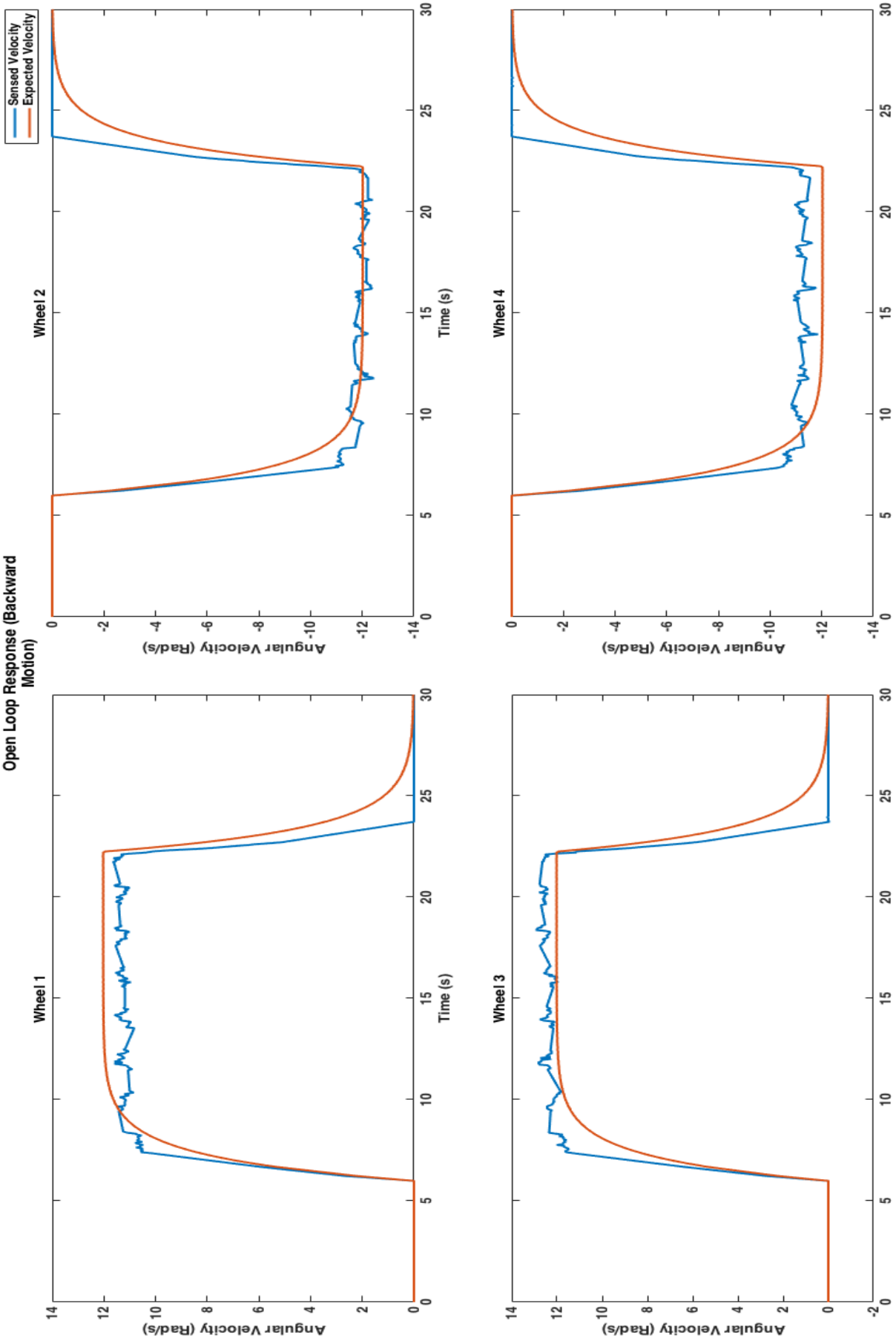

FIGURE 5.17: Open loop response of dynamic model: second youBot's behaviour backward motion

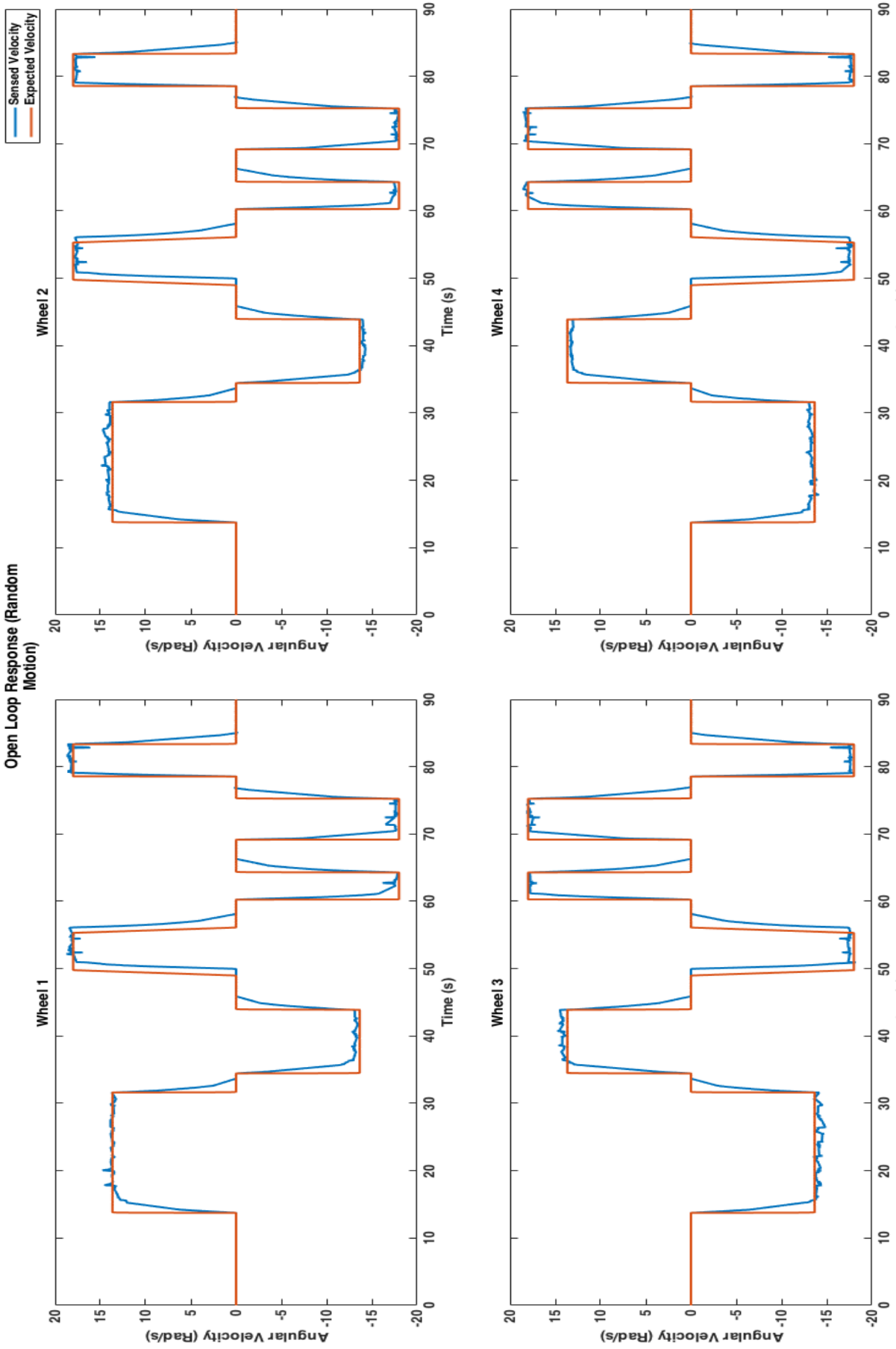

FIGURE 5.18: Open loop response of dynamic model: second youBot's behaviour random motion

5.2.3 Controller Evaluation

For evaluating the controller that was implemented on the dynamic model, the second model developed on simulink (Fig. 5.5) was used. As for the youBot, it was switched back to its default control mode which was velocity control. The aim of this experiment was to see if we can mimic the behaviour of the youBot using the same PID gains which were preset on the controllers.

The gains that were extracted from the youBot's drivers were as follows:

- P: 50
- I: 20
- D: 0

After running the youBot for several tests under velocity control mode, the data was gathered and then the input was also passed to our controlled dynamic model. The results of the comparison between the sensed velocity of the youBot, and the predicted velocity from the controlled dynamic model is shown in Fig. 5.19.

As seen from the plot, the controlled dynamic model predicts the motion of the youBot accurately. With that, we could see that we could mimic the behaviour of the robot and if needed, the controlled dynamic model could be used for simulating the behaviour of the youBot if the user wanted to change the controller parameters before deploying them. More results of testing can be found in the appendix B section of the report.

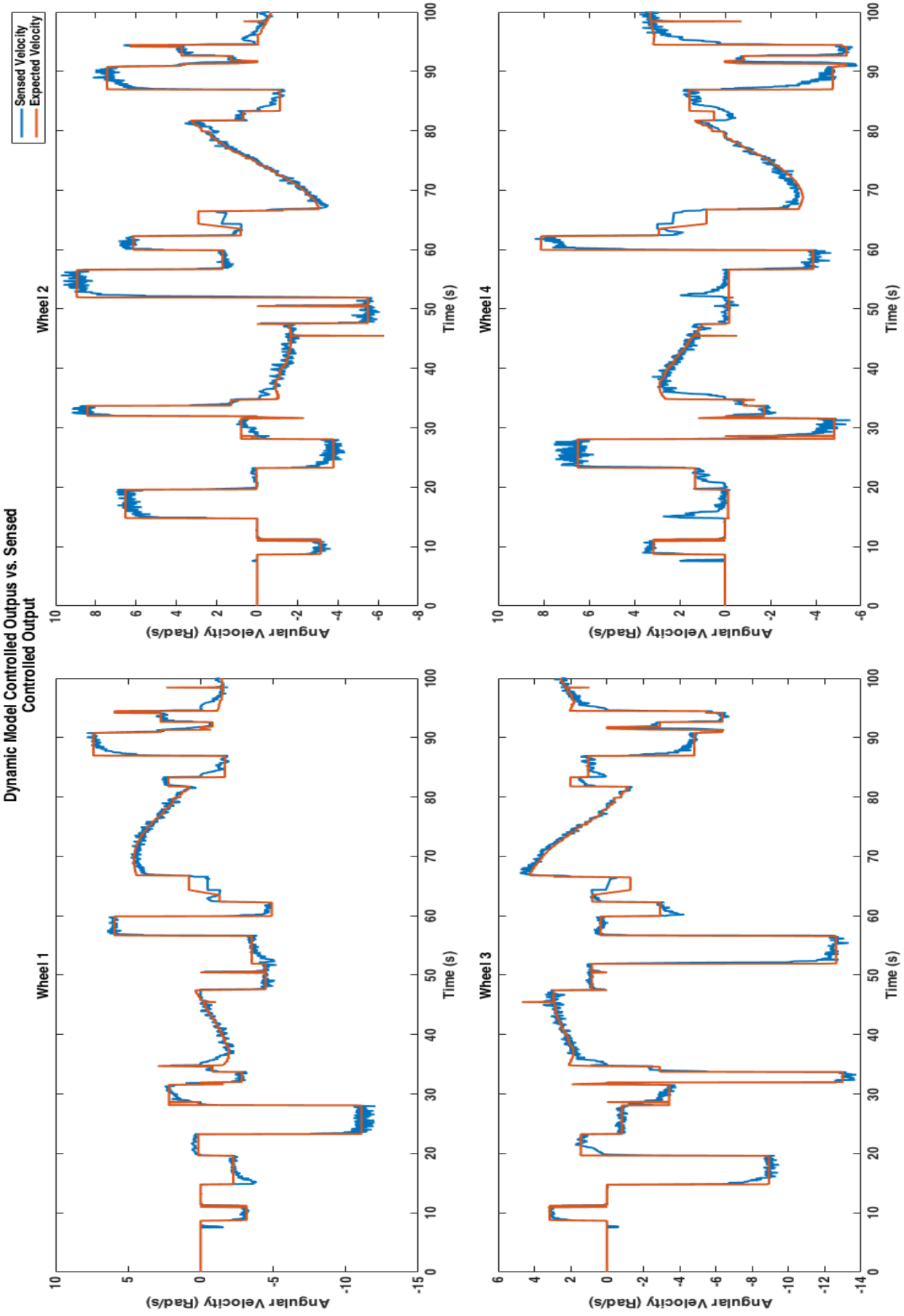

FIGURE 5.19: Comparision between the dynamic model controlled output and the youBot's controlled output

Chapter 6

Conclusion

This chapter will conclude the work presented throughout this report. A summary of the chapters will be given as well as a discussion regarding each topic presented. A section of the chapter will also be dedicated to discussing the future work that can be done based on the results of this report.

This report presented work regarding the development of a kinematic and a dynamic model which was used for the simulation of a four wheeled omnidirectional robot's behaviour. A PID controller was then implemented on the developed dynamic model and the model was also used for structural analysis which can be used for fault detection and diagnosis.

6.1 Summary

As discussed in the beginning of this report, the aim of the project is to develop a robust controller and to detect faults in motion of a four wheeled omnidirectional robot using any type of machine learning. At the beginning it seemed to be as a very complicated problem to tackle since there were many areas that needed to be studied in depth. However, after breaking the topic down to subtopics and organizing the tasks to be done, it was found to be feasible as long as if each subtopic was handled with high accuracy. To tackle the problem, the subtopics/tasks were organized as follows:

The first task was to understand the physics of an omnidirectional robot and to study the characteristics of the wheel. The second task was to develop a mathematical kinematic and dynamic model that can mimic the behaviour of an actual 4 wheeled omnidirectional robot. After that the developed model had to be validated with an actual robot which in our case, was the Kuka YouBot. The third task was to study ways of analyzing the

system with regards to faults and to find methods to detect and diagnose faults. The fourth task, which was done in parallel with the third task was to study and implement a suitable controller for the overall system and to also compare it with an actual robot.

As noticed, so far there is no mention of machine learning in these tasks. This is because after looking into research concerning machine learning and motion models, it was found that the best approach is to develop and implement everything mathematically first since the approaches already exist. And, this mathematical modelling process is exhausted or when there are behaviours that can not be explained using mathematical relations, machine learning can be used to fill in this gap and to improve the mathematical models.

After summarizing the chapters we will discuss how to improve what is already developed and what can be done next with regards to the scope of this project.

The first chapter of this report introduces the topic which is to be researched and discusses the importance of the work to be done and why it is relevant. A problem formulation is derived and the structure of the report is given.

The second chapter gives an overview of the background needed to be understood about the different aspects of this project. A review of the state of the art on modelling techniques, fault detection and isolation methods, and control methods is also given. This gave us guidlines to follow throughout this report.

The third chapter gives a detailed derivation of both the kinematic and dynamic model and walks the reader through the process. Both models describe the behaviour of a four wheeled omnidirectional robot in some manner. The kinematic model transforms the velocity of the omnidirectional platform from its center to the wheels or vice versa. The dynamic model takes into consideration the energies related to the motion of the platform such as kinetic and friction energies. Both models are used as a basis of our project and both were tested and validated later in the report.

The fourth chapter discusses the analysis of the developed models in terms of fault detection and diagnosis and discusses the relations between the constraints and the variables of the system. The chapter also discusses the control method proposed for the system and the basics of the PID theory.

The fifth chapter gives a detailed insight into the experimental evaluation performed to validate the kinematic and dynamic model and to also evaluate the controller that was implemented in chapter 4.

6.2 Limitations

From the work done for this project and from the experiments performed, some limitations were noticed.

From the developed dynamic model in chapter 3, it can be noticed that the model does not take into account the wheel slip that occurs in the omnidirectional wheel. This can sometimes affect the prediction of the model during complex maneuvers of the platform.

From the structural analysis of the system developed in chapter 4, we discerned that the relations derived hold true only when the platfom is behaving in steady state region. Although, since the platform is a dynamic system, false alarms could be raised from the developed relations.

From the experimental evaluation in chapter 5, some limitations were observed during experimentation. We could see in many of the runs that there is some mismatching between the reaction time of the model and the YouBot. This can also lead to raising false alarms when trying to detect faults.

6.3 Future Work

As seen in the previous section, there are many improvements that can be added to various parts of the project. Regardless, the accomplished research provides the necessary structure for any future work or improvements to be implemented. The future work which we regard suitable for the scope of this project is given in the following list:

- Considering wheel slip and adding it to the dynamic model of the system.
- In terms of structural analysis, the residuals should be adapted in such a way to account for the dynamic system. A method should be developed such that fault flags can be raised only when genuine faults occur.
- The developed dynamic model can also be further improved by the addition of a model of the brushless DC motors which run the omnidirectional wheels. This would give better predictions to the behaviour of the platform in general.
- Machine learning can now be introduced to further improve the model after the dynamic model has been derived mathematically.
- Since most simulation softwares only care about the velocity of the robot, the developed models can also be used in simulation softwares to improve the prediction of the robot movement.

- More control algorithms can be investigated and a comparison could be made using simulation only since the model is developed already.
- The models can be used further in applications such as path following or trajectory generation by using them as the predictor for a Kalman filter.

List of Figures

List of Tables

Appendix A

Obtaining first-order ODE's

The Lagragian equation is given by equation 3.18,

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q_i}}) - \frac{\partial L}{\partial q_i} + \frac{\partial R}{\partial \dot{q_i}} = 0 \tag{A.1}$$

The equations of motion that we obtained in chapter 3 are of second-order.But often it is useful to obtain first-order differential equations as they are easy to compute and they can be directly used in numerous applications.For structural analysis, first order ODE's are useful.

To obtain first-order ODE, we express the equations of motion obtained in chapter 3 using a new set of generalized co-ordinates denoted as p_i.We can express p_i as,

$$p_i = \frac{\partial L}{\partial \dot{q_i}} \tag{A.2}$$

Then the equation A.1 can be expressed as,

$$\dot{p_i} - \frac{\partial L}{\partial q_i} + \frac{\partial R}{\partial \dot{q_i}} = 0 \tag{A.3}$$

$$\dot{p_i} = \frac{\partial L}{\partial q_i} - \frac{\partial R}{\partial \dot{q_i}} \tag{A.4}$$

We can re-write (3.19) from chapter 3 using the above formula, we get,

$$\dot{\theta_1} = \frac{1}{K} \cdot \left[p_1 - \frac{mR^2\dot{\theta_4}}{8} + \frac{R^2 J_z \left(\dot{\theta_2} - \dot{\theta_3} + \dot{\theta_4}\right)}{16\left(l + L\right)^2} \right]$$

$$\dot{\theta_2} = \frac{1}{K} \cdot \left[p_2 - \frac{mR^2\dot{\theta_3}}{8} - \frac{R^2 J_z \left(-\dot{\theta_1} - \dot{\theta_3} + \dot{\theta_4} \right)}{16\left(l+L\right)^2} \right]$$

$$\dot{\theta_3} = \frac{1}{K} \cdot \left[p_3 - \frac{mR^2\dot{\theta_2}}{8} + \frac{R^2 J_z \left(-\dot{\theta_1} + \dot{\theta_2} + \dot{\theta_4} \right)}{16\left(l+L\right)^2} \right]$$

$$\dot{\theta_4} = \frac{1}{K} \cdot \left[p_4 - \frac{mR^2\dot{\theta_1}}{8} - \frac{R^2 J_z \left(-\dot{\theta_1} + \dot{\theta_2} - \dot{\theta_3} \right)}{16\left(l+L\right)^2} \right]$$

$$\dot{p_1} = -D_\theta \cdot \dot{\theta_1}$$

$$\dot{p_2} = -D_\theta \cdot \dot{\theta_2}$$

$$\dot{p_3} = -D_\theta \cdot \dot{\theta_3}$$

$$\dot{p_4} = -D_\theta \cdot \dot{\theta_4}$$

$where,$

$$K = \left(\frac{mR^2}{8} + \frac{J_z R^2}{16(l+L)^2} + J_w \right)$$

Appendix B

Additional Figures

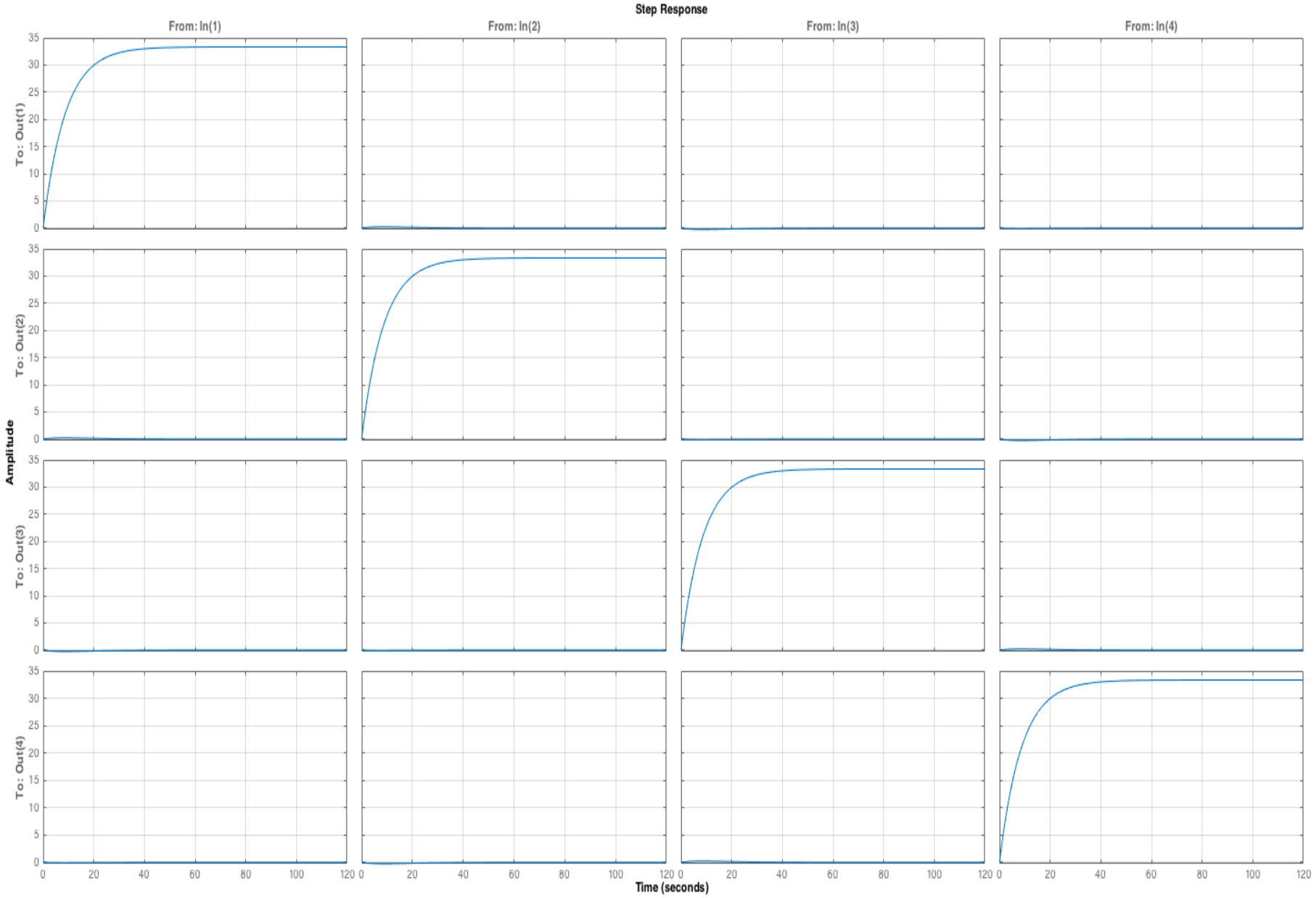

FIGURE B.1: Open Loop Response of Dynamic Model: Step input

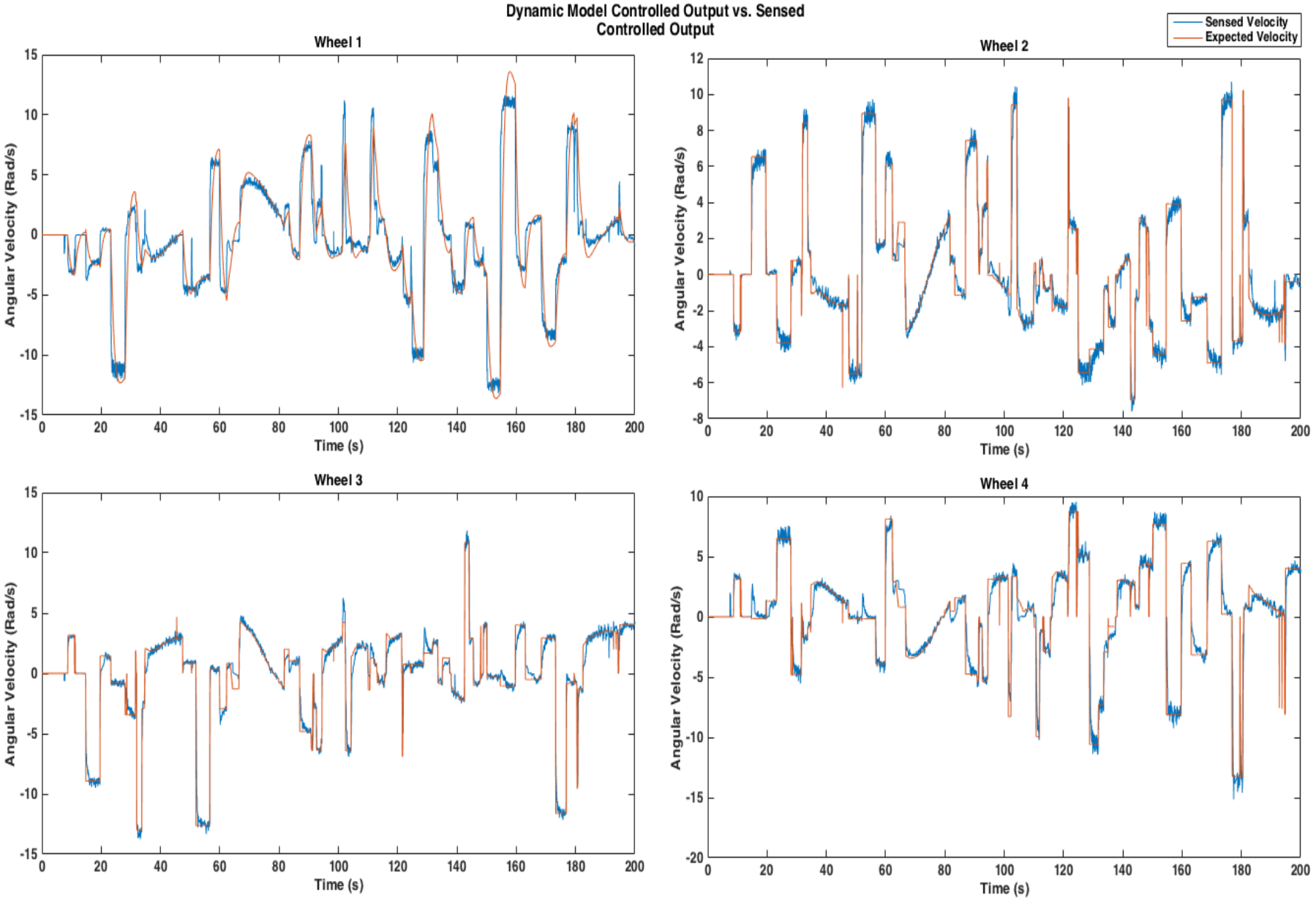

FIGURE B.2: Closed Loop Response of Dynamic Model: Different PID gains for wheel 1

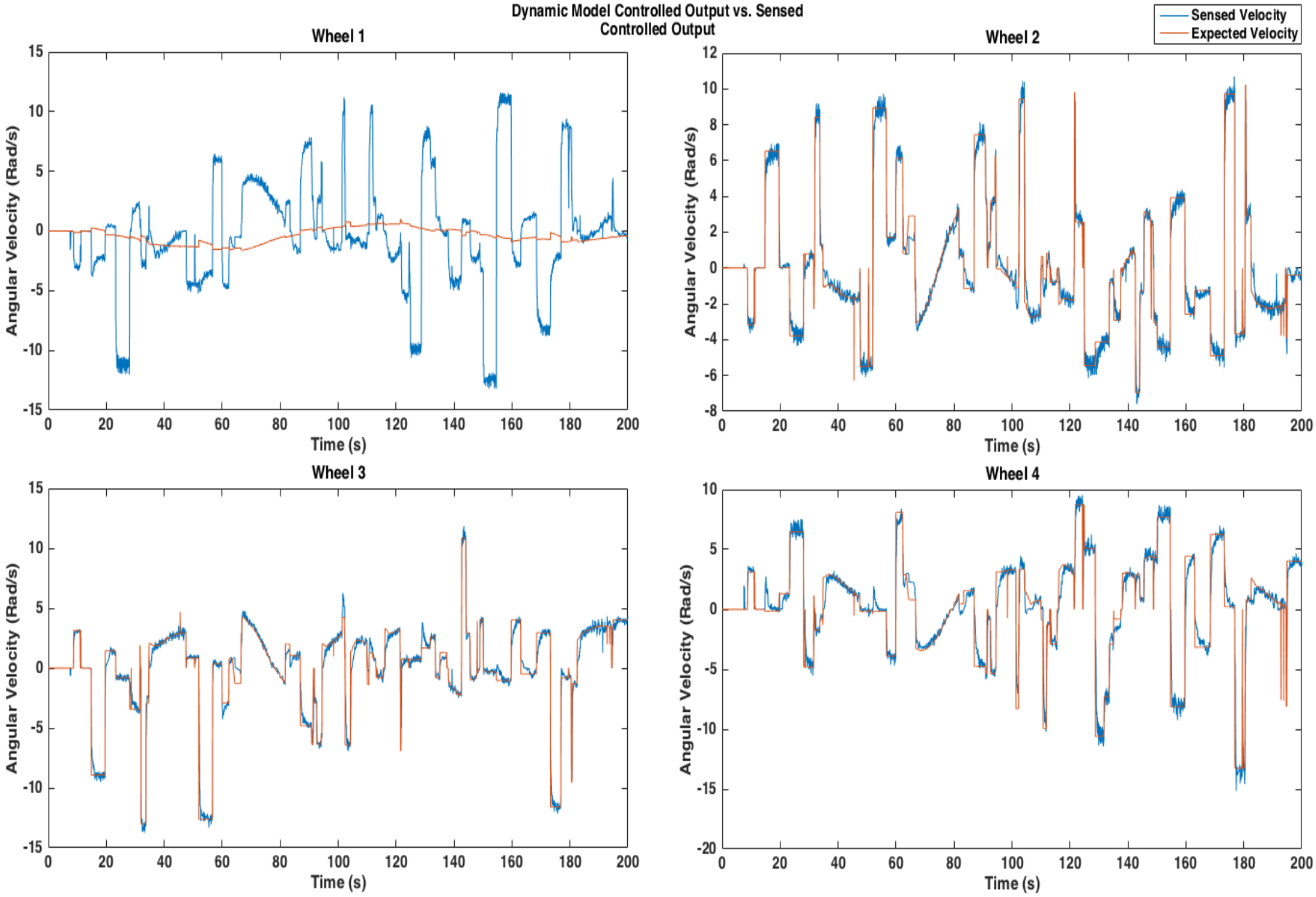

FIGURE B.3: Closed Loop Response of Dynamic Model: Different PID gains for wheel 1

Bibliography

[1] Shahrul Naim Sidek. Dynamic modeling and control of nonholonomic wheeled mobile robot subjected to wheel slip. 2008.

[2] Mihai Olimpiu, Dan Mândru, Ioan Ardelean, and Alin Ple. Design and De evelopment of an Aut onomous Directional Mobile Rob bot with h Mecanum Wheels. 2014.

[3] V. Spinu I. Doroftei, V. Grosu. "Omnidirectional Mobile Robot - Design and Implementation". In Maki K. Habib, editor, *Bioinspiration and Robotics*, chapter Climbing a, pages 511–529. Advanced Robotic Systems International (Vienna) and I- Tech, 2007.

[4] Zhiwei Gao, C Cecati, and S X Ding. A Survey of Fault Diagnosis and Fault-Tolerant Techniques Part I: Fault Diagnosis. *IEEE Transactions On Industrial Electronics*, 62(6):3768 – 3774, 2015. ISSN 0278-0046. doi: 10.1109/TIE.2015.2417501.

[5] Zhiwei Gao, Carlo Cecati, and Steven X. Ding. A survey of fault diagnosis and fault-tolerant techniques-part II: Fault diagnosis with knowledge-based and hybrid/active approaches. *IEEE Transactions on Industrial Electronics*, 62(6):3768–3774, 2015. ISSN 02780046. doi: 10.1109/TIE.2015.2419013.

[6] M. Luo, D. Wang, M. Pham, C.B. Low, J.B. Zhang, D.H. Zhang, and Y.Z. Zhao. Model-based fault diagnosis/prognosis for wheeled mobile robots: a review. In *31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005.*, page 6 pp. IEEE, 2005. ISBN 0-7803-9252-3. doi: 10.1109/IECON.2005.1569256. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1569256.

[7] J Agulló, S Cardona, and J Vivancos. Kinematics of vehicles with directional sliding wheels. *Mechanism and Machine Theory*, 22(4):295–301, 1987. ISSN 0094114X. doi: 10.1016/0094-114X(87)90018-8.

[8] Universitat Politrcnica De Catalunya. Mech. Maeh. Theory. *Mechanical Engineering*, 24(1):53–60, 1989. ISSN 0094114X. doi: 10.1016/0094-114X(92)90029-H.

[9] G Indiveri. Swedish Wheeled Omnidirectional Mobile Robots: Kinematics Analysis and Control. *IEEE {T}rans on {R}obotics*, 25(1):164–170, 2009.

[10] Pakpoom Viboonchaicheep, Akira Shimada, and Yuhki Kosaka. Position Rectification Control for Mecanum Wheeled Omni-directional Vehicles. *IECON Proceedings (Industrial Electronics Conference)*, 1:854–859, 2003. doi: 10.1109/IECON.2003.1280094.

[11] George K Fourlas, George C Karras, and Kostas J Kyriakopoulos. Fault Tolerant Control for a 4-Wheel Skid Steering Mobile Robot. pages 177–184, .

[12] Panagiotis Vlantis, Charalampos P Bechlioulis, George Karras, George Fourlas, and Kostas J Kyriakopoulos. Fault Tolerant Control for Omni-directional Mobile Platforms with 4 Mecanum Wheels. pages 2395–2400, 2016. ISSN 10504729. doi: 10.1109/ICRA.2016.7487389.

[13] Liu Yutian, Li Changgang, and Hu Junjie. Fault diagnosis and fault tolerant control of nonlinear systems. *2010 IEEE International Conference on Automation and Logistics*, (1):447–450, 2010. doi: 10.1109/ICAL.2010.5585326. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5585326.

[14] R. Beard. *Failure accommodation in linear system through self reorga- nization.* Ph.d. dissertation, MIT, Cambridge, MA, USA, 1971.

[15] Steven X. Ding. *"Model-based Fault Diagnosis Techniques: Design Schemes, Algorithms, and Tools"*. Springer-Verlag Berlin Heidelberg 2008, 2008. ISBN 9781447147992 (online) • 9781447147985. doi: 10.1007/978-1-4471-4799-2.

[16] George K Fourlas, Stavros Karkanis, George C Karras, and Kostas J Kyriakopoulos. Model Based Actuator Fault Diagnosis for a Mobile Robot. pages 79–84, 2014.

[17] Janos J Gertler. Analytical redundancy methods in fault detection and isolation-survey and synthesis. In *Proceedings of IFAC/IAMCS symposium on safe process*, volume 1, pages 9–21. Preprints of IFAC/IMACS Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS'91. 1991., 1991.

[18] Demetris Stavrou, Demetrios G. Eliades, Christos G. Panayiotou, and Marios M. Polycarpou. Fault detection for service mobile robots using model-based method. *Autonomous Robots*, 40(2):383–394, 2016. ISSN 15737527. doi: 10.1007/s10514-015-9475-7.

[19] Ngoc Bach Hoang, Hee-jun Kang, and Young-shick Ro. Fault Detection and Isolation in Wheeled Mobile Robot. pages 563–569, 2012.

[20] Dilek Düştegör, Erik Frisk, Vincent Cocquempot, Mattias Krysander, and Marcel Staroswiecki. Structural analysis of fault isolability in the DAMADICS benchmark. *Control Engineering Practice*, 14(6 SPEC. ISS.):597–608, 2006. ISSN 09670661. doi: 10.1016/j.conengprac.2005.04.008.

[21] M. Staroswiecki Declerck and P. "Analytical redundancy in nonlinear interconnected systems by means of structural analysis". In *IFAC Advanced Information Processing in Automatic Control, 1989*, 1989.

[22] K. Valavanis A. Monteriù, P. Asthan and S. Longhi. "Model-Based Sensor Fault Detection and Isolation System for Unmanned Ground Vehicles: Experimental Validation (part II)". In *2007 IEEE International Conference on Robotics and Automation Roma, Italy, 10-14 April 2007.*, 2007.

[23] R. Izadi-Zamanabadi. "Structural Analysis Approach to Fault Diagnosis with Application to Fixed-wing Aircraft Motion". In *in Proc. of American control Conference, USA, 2002.*, 2002.

[24] et al. Control'98. Cocquempot, Vincent. "Residual generation for the ship benchmark using structural approach.". In *UKACC International Conference on (Conf. Publ. No. 455). IET, 1998.*

[25] G.K. Fourlas. "Theoretical Approach of Model Based Fault Diagnosis for a 4 - Wheel Skid Steering Mobile Robot". In *21st IEEE Mediterranean Conference on Control and Automation (MED '13), Platanias-Chania, Crete, GREECE, June 25-28, 2013.*, 2013.

[26] K. Valavanis A. Monteriù, P. Asthan and S. Longhi. "Model-Based Sensor Fault Detection and Isolation System for Unmanned Ground Vehicles: Theoretical Aspects (part I)". In *2007 IEEE International Conference on Robotics and Automation Roma, Italy, 10-14 April 2007.*, 2007.

[27] M Blanke and T Lorentzen. SaTool—A software tool for structural analysis of complex automation systems. *Fault Detection, Supervision and ...*, (431): 673–678, 2006. doi: 10.3182/20060829-4-CN-2909.00104. URL http://www.ifac-papersonline.net/Detailed/31553.html.

[28] Lorentzen, Torsten and Mogens Blanke. "Sa Tool–Users Manual.". Technical report, 2004.

[29] Deebul Nair. SaTool -python. URL https://github.com/deebuls/sa{_}tool{_}python.git.

[30] George K Fourlas, George C Karras, and Kostas J Kyriakopoulos. Fault Tolerant Control for a 4-Wheel Skid Steering Mobile Robot. pages 177–184, .

[31] Vasilyev A.S.; Ushakov. "Modeling of dynamic systems with modulation by means of Kronecker vector-matrix representation.". *Scientific and Technical Journal of Information Technologies, Mechanics and Optics.*, 15(5):839–848, 2015.

[32] Norman S. Nise. *Control Systems Engineering.* John Wiley & Sons, Inc., 6th edition, 2010. ISBN 978-0-470-54756-4.

[33] Mogens Blanke, Jan Lunze, Michel Kinnaert, Marcel Staroswiecki, and Jochen Schröder. *Diagnosis and fault-tolerant control.* Berlin: springer, 3rd edition, 2006. ISBN 3540356525. doi: 10.1007/978-3-540-35653-0.

[34] Jinghua Zhong. PID Controller Tuning. 2006.

[35] Sonja Dwersteg. Module for Stepper Motors Tmcl ™ Firmware Manual, 2011. URL http://www.trinamic.com.