

Annual report of the "Robotautomatisering i fjerkræslagteri og- kødforarbejdning TRIN 1"

Deliverable D1.22

Kinematic Analysis & Stability Optimization and Network Control for Compliant Motion of the RAGNAR Manipulator

Funding Instrument: Grønt Udviklings- og Demonstrationsprogram (GUDP)

Project Start: 1 January 2020

Project Duration: 36 months

Beneficiary in Charge: Henrik Schiøler

Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the Consortium (including the Commission Services)	
CO	Confidential, only for members of the Consortium (including the Commission Services)	✓

This project has received funding from the Grønt Udviklings- og Demonstrationsprogram (GUDP)

Deliverable Information

Document Administrative Information	
Deliverable Number:	D1.22
Deliverable Full Title:	Kinematic Analysis & Stability Optimization and Network Control for Compliant Motion of the RAGNAR Manipulator
Deliverable Short Title:	[RAGNAR Motion Analysis & Network Control]
Beneficiary in Charge:	Henrik Schiøler
Report Version:	vn.1.3
Contractual Date:	01/01/2020
Report Submission Date:	07/07/2022
Nature:	[Report]
Lead Author(s):	Henrik Schiøler
Co-author(s):	Shahab Heshmati-Alamdari
Keywords:	[Manipulators, Constrained Kinematics, Kinematics singularity, Kinematics analysis, Compliant Control, Network Control, Manipulator Control]
Status:	__ draft, <u>x</u> final, __ submitted

Change Log

Date	Version	Author/Editor	Summary of Changes Made
01/12/2021	v1.0	Shahab Heshmati-Alamdari	Draft report template created.
18/03/2022	v1.1	Shahab Heshmati-Alamdari	First version of the draft report modified.
30/06/2022	v1.2	Shahab Heshmati-Alamdari & Henrik Schiøler	Required changes have been modified.
07/07/2022	v1.3	Shahab Heshmati-Alamdari & Henrik Schiøler	The final changes have been modified. Ready to submit.

Table of Contents

Summary	7
1. Purpose and Scope of the Document	8
1.1 Structure of the Document	9
2. Kinematic Analysis	10
2.1 Revised Mechanical Construction	12
3. Optimization of Linear Actuator Placement	14
3.1 Description of the Optimization Problem	15
3.2 Formulating the Optimization Problem	16
3.3 Results of Solving the Optimization Problem	17
3.4 Multi-Objective Constrained Optimization of Actuator Attachment Points	19
3.5 Confirmation of Singularity Removal	21
4. State Space Model	23
4.1 Compliant Motion	23
4.2 State Space Modeling	24
5. Controller Design	26
5.1 Controllability	26
5.2 State Feedback	26
5.3 Full-Order Observer	28
5.4 State Feedback Controller	30
5.5 Full-Order Observer	32
6. Network Simulation	33
6.1 CAN-bus Simulation	33
6.2 Networked Controller - Simulation Results	37
6.3 Sample Rate Selection	39
6.4 Packet Loss	39
7. Results	47
8. Conclusions	52
Bibliography	53
Appendix A: Joint Space Transformation	54

List of Figures

Figure 1: The first version of the parallel manipulator system "Ragnar".....	8
Figure 2: Movement of the end-effector with all four proximal links locked in place.	9
Figure 3: Illustration of one motor, proximal link, distal link and end-effector, showing the notation used to model the mechanics of Ragnar.	10
Figure 4: The end-effector with the notation used to model the mechanics of Ragnar.	13
Figure 5: Feasible region for the linear actuator attachment points.	15
Figure 6: Cases where attachment points results in a clash between the linear actuators. Actuator a_5 has attachment coordinates (a_{5x}, a_{5y}, a_{5z}) , and actuator a_6 attachment coordinates (a_{6x}, a_{6y}, a_{6z})	16
Figure 7: Ragnar robot in the operating pose with the linear actuators attached.	17
Figure 8: Ragnar robot in the pick-up pose with the linear actuators attached.	18
Figure 9: Ragnar robot in the hand-off pose (f1), using the actuator attachment points in table 2. ...	20
Figure 10: Ragnar robot in the pick-up pose (f2), using the actuator attachment points in table 2.	20
Figure 11: Condition number of Jacobian through transportation trajectory after modified mechanical construction.....	21
Figure 12: Trajectory used for calculation of conditioning number of jacobian.	22
Figure 13: Condition number of Jacobian through transportation trajectory result from previous research group [4].	23
Figure 14: State feedback structure block diagram . [7]	27
Figure 15: Full-order observer block diagram.....	29
Figure 16: Block diagram configuration used in Simulink® to simulate Ragnar with state feedback....	30
Figure 17: Input signal to state feedback simulation of Ragnar.	31
Figure 18: Output signal from state feedback simulation of Ragnar.	31
Figure 19: Simulink setup for simulation of the system with a full-order observer. The green box in the top is the model for Ragnar. The blue box contains the observer model and observer gain.	32
Figure 20: Comparison of the output from Ragnar and from the observer. Blue curves are the outputs from the observer. The orange curves are outputs from Ragnar.	33
Figure 21: The difference between CAN-bus frame and CANOpen frame [9].	34
Figure 22: Network scheduling design showing the execution of the Teensy node (green), the sending of messages (red) as well as the queuing of the messages (blue).	35
Figure 23: Pseudocode of the network simulation.....	36
Figure 24: Timings in one control loop.	36
Figure 25: Simulation result of network. Figure b) is a zoomed in version of start of A. Integers on the y-axis indicate node number as well as priority, with 0 being highest. The y-values also represent sending state: $y + 0$ = Node y not sending. $y + 0,5$ = Node y sending. $y + 0,25$ = Node y waiting to send (eg. preempted by higher priority node).....	38
Figure 26: Simulation results showing packet loss consequences with and without the observer. Here, the packet loss occurs just before a transient period is about to start.	40
Figure 27: Simulation results showing packet loss consequences with and without the observer. Here, the packet loss occurs in the middle of a transient period.	42
Figure 28: Simulation results showing packet loss consequences, in respect to velocities, with and without the observer. Here, the packet loss occurs in the middle of a transient period.....	44
Figure 29: Simulation results showing packet loss consequences with and without the observer. Here, the packet loss occurs just as a transient period is about to end.....	46
Figure 30: CAN-bus network topology used for TrueTime simulations.	47
Figure 31: Simulation result of the networked observer and non-networked observer controllers.	49



Figure 32: Simulation result of the networked observer controller with and without resolution. The
grey lines represent the resolution of the nodes. 50

Figure 33: Block diagram showing conversion between configuration space and task space. 55



List of Tables

Table 1: Results from optimization, both coordinates and the obtainable condition number. The values are in meters related to the origin. 17

Table 2: Table of optimization result. Coordinate values are in meters, relative to the origin. 19

Table 3: Poses used as reference in the simulation test..... 48

Summary

This report presents geometric optimization and compliant control of the "Ragnar" parallel manipulator, designed for handling pick-and-place tasks of delicate objects. Previous work on the robot concluded that specific motor configurations resulted in multiple end-effector poses. Mathematically this is expressed as an ill-conditioned kinematic Jacobian, making forward kinematics impossible for certain poses. Therefore, it was suggested to modify the current mechanical construction. This is achieved by adding two linear actuators with optimized mounting points to yield a lower condition number [1]. The new system is modelled mathematically, its kinematics investigated, and a compliant controller designed. The controller is designed as a full-order observer with state feedback and tested on a simulated CAN-bus network. It is presented how the mounting points for the the new limbs were found. Furthermore, a simulation is performed in task space, showing the system's behavior with and without the influence of CAN-bus and the impact of packet loss on the network. The conclusion is that the modified mechanical construction significantly improves the condition number of the Jacobian. Simulations thus show that control is now achievable.

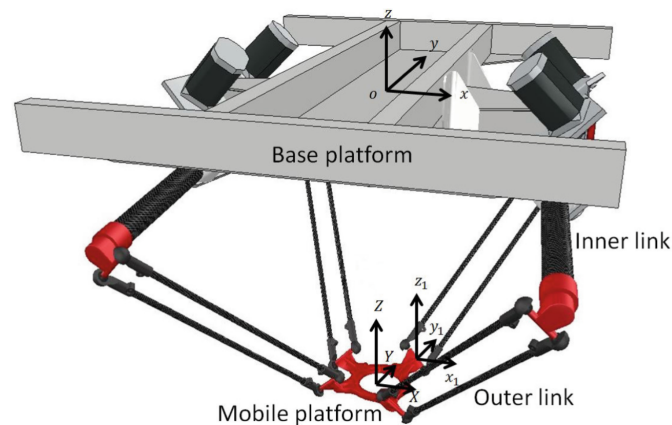
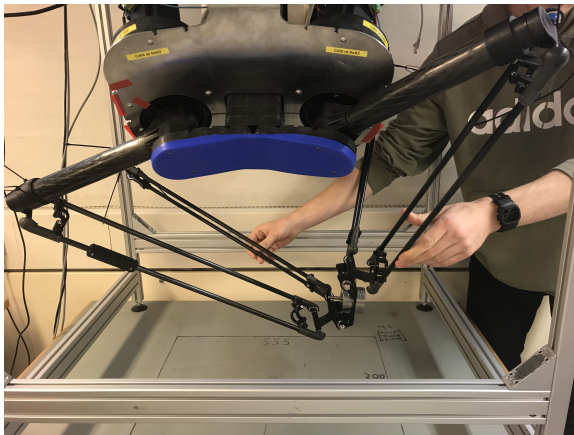


Figure 1: The first version of the parallel manipulator system "Ragnar".

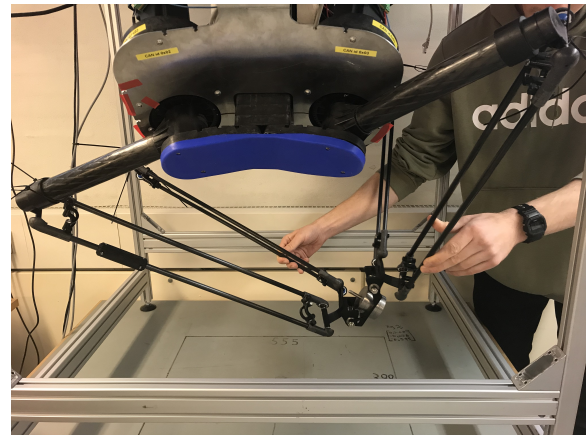
1 Purpose and Scope of the Document

It is already known that the kinematic Jacobian of the Ragnar robot becomes singular with increasing end-effector pose elements y and φ [4]. Mathematically this is expressed as an ill-conditioned kinematic Jacobian, making forward kinematics impossible for certain poses. The aforementioned singularity is not only an issue for the numerical solution, it also manifests in the physical system. With the four actuators in a locked position, it is still possible to move the end-effector, thus a mapping from one specific actuator configuration gives multiple poses. This makes it impossible to control the tilt angle of the end-effector with the four actuated joints alone. From figure 2, the movement of the end-effector with all four proximal links locked in place can be observed.

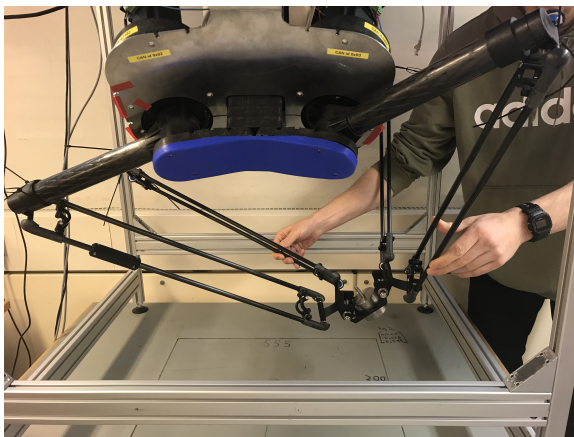
Therefore, it was suggested to modify the Ragnar's current mechanical construction. This document presents our effort to change and improve the Ragnar's construction in a such a way to improve the condition number of the kinematic Jacobian, thus suggesting that forward kinematics are now possible to derive. The result of this study suggested to rebuild Ragnar and attach the two limbs in the optimized positions. This is achieved by adding two linear actuators with optimized mounting points to yield a lower condition number [1]. Then a forward kinematics model for transformation from joint to task space was derived, in order to calculate the end-effector pose during the transportation between the pick-up and the hand-off pose. Hereafter, it can be attempted to design a controller for Ragnar that can perform control in all poses that are reachable for the end-effector. More specifically, this report presents geometric optimization and compliant control of the "Ragnar" parallel manipulator. The new system is modelled mathematically, its kinematics investigated, and a compliant controller designed. The controller is designed as a full-order observer with state feedback and tested on a simulated CAN-bus network. It is presented how the mounting points for the the new limbs were found. Furthermore, a simulation is performed in task space, showing the system's behavior with and without the influence of CAN-bus and the impact of packet loss on the network. The conclusion is that the modified mechanical construction significantly improves the condition number of the Jacobian. Simulations thus show that control is now achievable.



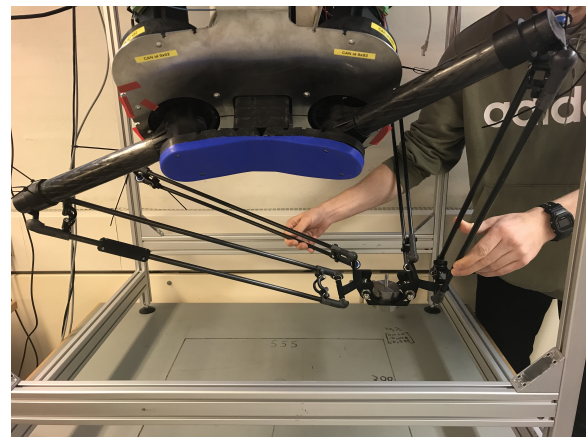
(a) Position 1



(b) Position 2



(c) Position 3



(d) Position 4

Figure 2: Movement of the end-effector with all four proximal links locked in place.

1.1 Structure of the Document

This document is organised as follows: Section 2 provides information about the kinematics of the Ragnar robot and how these are derived by means of Jacobian analysis. In Section 3 provides information about the improvement of the Ragnar stability by adding linear actuators in the term of kinematic Jacobian and its condition number. In addition, this section describes, formulates, and solves an optimization problem in order to improve the aforementioned issue and to find the best solution according to a defined set of constraints. Section 4 provides information about the state space model of the Ragnar along with its compliant motion in the task space. Section 5 provides information about the controllability of the system and the designing of a proper state feedback control architecture. section 6 provides information about the developed CAN-bus network simulation of this work. In Section 7 the final system is presented and the founding results of this work are verified though simulation experimental studies. Finally, the deliverable and the presented results are concluded in Section 8.

2 Kinematic Analysis

In this section the kinematics of the Ragnar robot are derived by means of Jacobian analysis. Hereby allowing for translation between joint velocities and end-effector velocities. Ragnar in its current configuration consists of four limbs. Each of the four limbs has an actuated proximal link and a passive distal link which are connected to the end-effector capable of rotating around the x-axis. One isolated limb connected to the end-effector can be seen from illustration 3.

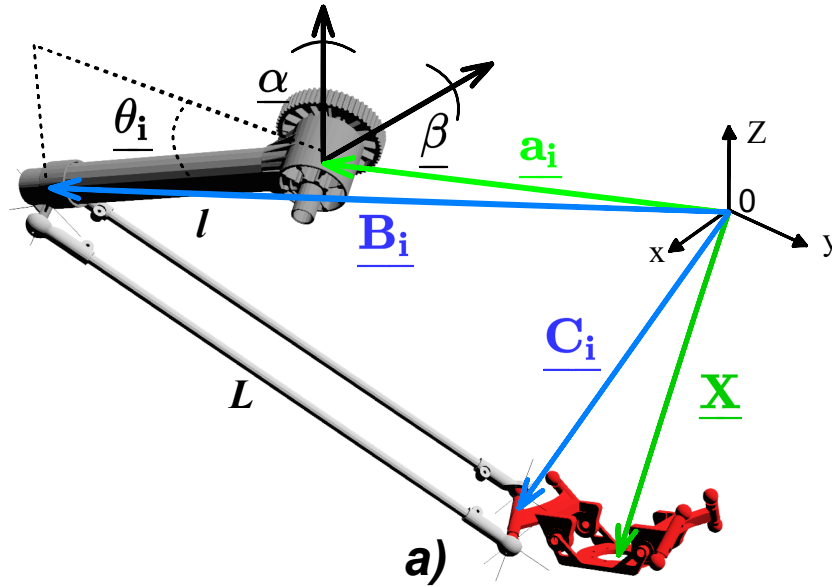


Figure 3: Illustration of one motor, proximal link, distal link and end-effector, showing the notation used to model the mechanics of Ragnar.

The entire analysis is performed in respect to the Cartesian coordinate system, with origin in the center of Ragnar. This can be observed in figure 3 together with the different notations used throughout the analysis. With subscript i indicating one to four limbs, one underline marking a vector (e.g. $\underline{x}$), and double underline a matrix (e.g. $\underline{\underline{A}}$). The rotation of the motors in its mounted positions in relation to the coordinate system is formulated as:

$$\underline{\underline{R}}_i = \underline{\underline{R}}_z^\alpha \cdot \underline{\underline{R}}_y^\beta \quad \text{for } i = 1..4 \quad (1)$$

Where:

$\underline{\underline{R}}_i$ rotation matrix of motor.

$\underline{\underline{R}}_z^\alpha$ rotation matrix of motor in the z-direction.

$\underline{\underline{R}}_y^\beta$ rotation matrix of motor in the y-direction.

α is the mount angle in the z-direction.

β is the mount angle in the y-direction.

The vector $\underline{B}_i$ from origin to the end of the proximal link is defined as.

$$\underline{B}_i = \underline{a}_i + l \cdot \underline{U}_x \cdot \underline{\underline{R}}_i \cdot \underline{\underline{R}}_z^\theta \quad (2)$$

Where:

$\underline{a}_i$ is the vector from the origin to motor rotation center.

l is the length of the proximal link.

R_z^θ is the rotation matrix of the proximal link around the z-axis.

$\underline{U}_x$ is a unit vector in the x direction.

The vector going from origin to the end-effector corner attachment of the i'th distal link $\underline{C}_i$ is defined as.

$$\underline{C}_i = \underline{X} + \underline{R}_x^\phi \cdot \underline{r}_i + \underline{r}_{a_i} \quad (3)$$

Where:

$\underline{X}$ is the vector going from origin to the center of the end-effector.

$\underline{R}_x^\phi$ is the rotation matrix of the end-effector around the x-axis.

$\underline{r}_i$ is a vector from the center of the platform to the i'th rotating axis

$\underline{r}_{a_i}$ is a vector from rotating axis to the i'th arm attachment.

How $\underline{r}_i$ and $\underline{r}_{a_i}$ relates to the physical system, can be seen on figure 4. The length L is found and squared representing the distal links length. The length of $\underline{B}_i - \underline{C}_i$ is found with 2-norm [13].

$$\|\underline{B}_i - \underline{C}_i\|_2^2 = L^2 \quad (4)$$

Both sides are differentiated w.r.t. time and the 2-norm is expanded by using the inner product.

$$\frac{d}{dt} \langle \underline{B}_i - \underline{C}_i, \underline{B}_i - \underline{C}_i \rangle = \frac{d}{dt} L^2 \quad (5)$$

$$\frac{d}{dt} \left((\underline{B}_i - \underline{C}_i)^T (\underline{B}_i - \underline{C}_i) \right) = 0 \quad (6)$$

Differentiating using the product rule and denoting derivatives with the dot-notation.

$$(\dot{\underline{B}}_i - \dot{\underline{C}}_i)^T \cdot (\underline{B}_i - \underline{C}_i) + (\underline{B}_i - \underline{C}_i)^T \cdot (\dot{\underline{B}}_i - \dot{\underline{C}}_i) = 0 \quad (7)$$

By utilizing the property $x^T y = y^T x$, the above equation can be rewritten into the following equation and reduced further.

$$(\underline{B}_i - \underline{C}_i)^T \cdot (\dot{\underline{B}}_i - \dot{\underline{C}}_i) + (\underline{B}_i - \underline{C}_i)^T \cdot (\dot{\underline{B}}_i - \dot{\underline{C}}_i) = 0 \quad (8)$$

$$2 \cdot (\underline{B}_i - \underline{C}_i)^T \cdot (\dot{\underline{B}}_i - \dot{\underline{C}}_i) = 0 \quad (9)$$

$$(\underline{B}_i - \underline{C}_i)^T \cdot (\dot{\underline{B}}_i - \dot{\underline{C}}_i) = 0 \quad (10)$$

$$(\underline{B}_i - \underline{C}_i)^T \cdot \dot{\underline{B}}_i - (\underline{B}_i - \underline{C}_i)^T \cdot \dot{\underline{C}}_i = 0 \quad (11)$$

Now Eq. (11) is multiplied by 1 on each side, denoted by $\frac{d\theta}{d\theta}$ and $\frac{dP}{dP}$ respectively.

$$(\underline{B}_i - \underline{C}_i)^T \cdot \frac{d}{dt} \underline{B}_i \cdot \frac{d\theta}{d\theta} = (\underline{B}_i - \underline{C}_i)^T \cdot \frac{d}{dt} \underline{C}_i \cdot \frac{dP}{dP} \quad (12)$$

$$(\underline{B}_i - \underline{C}_i)^T \cdot \frac{\partial \underline{B}_i}{\partial \underline{\theta}} \cdot \dot{\underline{\theta}} = (\underline{B}_i - \underline{C}_i)^T \cdot \frac{\partial \underline{C}_i}{\partial \underline{P}} \cdot \dot{\underline{P}} \quad (13)$$

Where,

$\underline{P} = [x \ y \ z \ \varphi]^T$ is the pose of the end-effector.

$\underline{\theta} = [\theta_1 \ \theta_2 \ \theta_3 \ \theta_4]^T$ is the vector containing actuator rotation angles.

These equations can now be put into matrix form:

$$\begin{bmatrix} (\underline{B}_1 - \underline{C}_1)^T \cdot \frac{\partial \underline{B}_1}{\partial \underline{\theta}} \\ (\underline{B}_2 - \underline{C}_2)^T \cdot \frac{\partial \underline{B}_2}{\partial \underline{\theta}} \\ (\underline{B}_3 - \underline{C}_3)^T \cdot \frac{\partial \underline{B}_3}{\partial \underline{\theta}} \\ (\underline{B}_4 - \underline{C}_4)^T \cdot \frac{\partial \underline{B}_4}{\partial \underline{\theta}} \end{bmatrix} \cdot \underline{\dot{\theta}} = \begin{bmatrix} (\underline{B}_1 - \underline{C}_1)^T \cdot \frac{\partial \underline{C}_1}{\partial \underline{P}} \\ (\underline{B}_2 - \underline{C}_2)^T \cdot \frac{\partial \underline{C}_2}{\partial \underline{P}} \\ (\underline{B}_3 - \underline{C}_3)^T \cdot \frac{\partial \underline{C}_3}{\partial \underline{P}} \\ (\underline{B}_4 - \underline{C}_4)^T \cdot \frac{\partial \underline{C}_4}{\partial \underline{P}} \end{bmatrix} \cdot \underline{\dot{P}} \quad (14)$$

(15)

Using letters A and B to denote the two matrices, it reduces to:

$$\underline{A} \cdot \underline{\dot{\theta}} = \underline{B} \cdot \underline{\dot{P}} \quad (16)$$

The B-matrix is moved by applying the Moore-Penrose pseudoinverse of $\underline{B}$ as it is singular and thus not invertible. The Moore-Penrose pseudoinverse of a matrix is given by:

$$\underline{B}^\dagger = (\underline{B}^T \cdot \underline{B})^{-1} \cdot \underline{B}^T \quad (17)$$

And thus the system of the linear equations becomes the following.

$$\underline{A} \cdot \underline{\dot{\theta}} = \underline{B} \cdot \underline{\dot{P}} \quad (18)$$

$$\underline{\dot{P}} = \underline{B}^\dagger \underline{A} \cdot \underline{\dot{\theta}} \quad (19)$$

$$\underline{\dot{P}} = \underline{J} \cdot \underline{\dot{\theta}} \quad (20)$$

$$\underline{J}^{-1} \cdot \underline{\dot{P}} = \underline{\dot{\theta}} \quad (21)$$

Where $\underline{J}$ is the kinematic Jacobian. Which when applied allows to convert between joint velocities and task space velocities. It is noted that in order to calculate the Jacobian, both pose and the associated joint angles are needed as $\underline{B}$ and $\underline{A}$ are dependent on them respectively. Inverse kinematics are used to find the joint angles in order to make the Jacobian only pose-dependent. These exist from previous work done on Ragnar and are based on trigonometric functions where each limb is modelled as a triangle.

2.1 Revised Mechanical Construction

The mechanical construction is revised to remove the singularity mentioned in section 1. As it is advised to not have electronics in the end-effector, due to daily power washing and appliance with food safe regulations, it is chosen to place two linear actuators on Ragnar together with the existing limbs. This will result in two new θ -values, called θ_5 and θ_6 . These values are equal to the length of the two linear actuators. The vectors $a_{5,6}$ are from the origin to the mounting point of the linear actuators on Ragnar. They can be observed together with additional notation on the figure below.

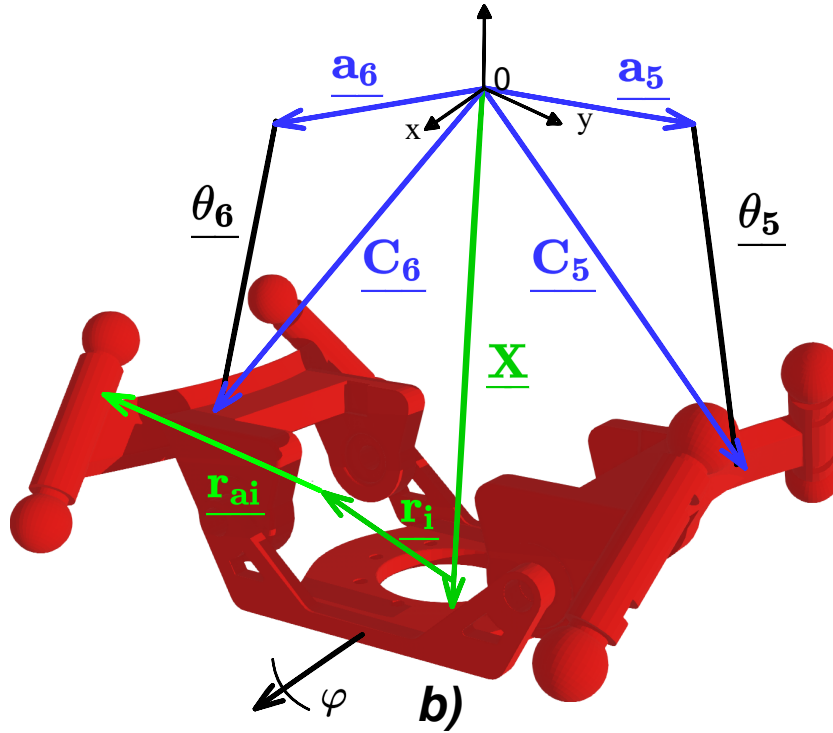


Figure 4: The end-effector with the notation used to model the mechanics of Ragnar.

Vectors from the origin to the linear actuator mounting points on the end-effector are constructed as follows:

$$\underline{C}_5 = \frac{1}{2}(\underline{C}_1 + \underline{C}_2) \quad \wedge \quad \underline{C}_6 = \frac{1}{2}(\underline{C}_3 + \underline{C}_4) \quad (22)$$

The following equations will be identical for both linear actuators, and thus only the derivations for θ_5 will be shown. As seen on figure 4 the sum of $\underline{a}_5$ and θ_5 equals $\underline{C}_5$. Thus again in same manner as in the previous section an equality constraint of lengths squared is acquired.

$$\|\underline{a}_5 - \underline{C}_5\|_2^2 = \theta_5^2 \quad (23)$$

The same steps as done with the original four equality constraint are once again made, hereby allowing the two to be combined. First rewriting it into an inner product as well as taking the derivative wrt. time.

$$\frac{d}{dt} \langle \underline{a}_5 - \underline{C}_5, \underline{a}_5 - \underline{C}_5 \rangle = \frac{d}{dt} \theta_5^2 \quad (24)$$

$$\frac{d}{dt} \left((\underline{a}_5 - \underline{C}_5)^T (\underline{a}_5 - \underline{C}_5) \right) = \frac{d}{dt} \theta_5^2 \quad (25)$$

Using the product rule when taking the derivative and making use of the dot-notation.

$$(\underline{a}_5 - \underline{C}_5)^T \cdot (\underline{a}_5 - \underline{C}_5) + (\underline{a}_5 - \underline{C}_5)^T \cdot (\dot{\underline{a}}_5 - \dot{\underline{C}}_5) = \dot{\theta}_5 \cdot \theta_5 + \theta_5 \cdot \dot{\theta}_5 \quad (26)$$

By utilizing $x^T y = y^T x$, the equation is further reduced.

$$(\underline{a}_5 - \underline{C}_5)^T \cdot (\underline{a}_5 - \underline{C}_5) + (\underline{a}_5 - \underline{C}_5)^T \cdot (\dot{\underline{a}}_5 - \dot{\underline{C}}_5) = 2 \cdot \theta_5 \cdot \dot{\theta}_5 \quad (27)$$

$$2 \cdot (\underline{a}_5 - \underline{C}_5)^T \cdot (\dot{\underline{a}}_5 - \dot{\underline{C}}_5) = 2 \cdot \theta_5 \cdot \dot{\theta}_5 \quad (28)$$

$$(\underline{a}_5 - \underline{C}_5)^T \cdot (\dot{\underline{a}}_5 - \dot{\underline{C}}_5) = \theta_5 \cdot \dot{\theta}_5 \quad (29)$$

$$(30)$$

Lastly, as done previously, the left- and right-hand side of the equation is multiplied by $\frac{dP}{d\theta}$ and $\frac{d\theta}{d\theta}$ respectively. This gives:

$$(\underline{a}_5 - \underline{C}_5)^T \cdot \frac{d}{dt}(\underline{a}_5 - \underline{C}_5) \cdot \frac{dP}{dP} = \theta_5 \cdot \frac{d}{dt}\theta_5 \cdot \frac{d\theta}{d\theta} \quad (31)$$

$$(\underline{a}_5 - \underline{C}_5)^T \cdot \frac{d}{dP}(\underline{a}_5 - \underline{C}_5) \cdot \frac{dP}{dt} = \theta_5 \cdot \frac{d}{d\theta}\theta_5 \cdot \frac{d\theta}{dt} \quad (32)$$

$$(\underline{a}_5 - \underline{C}_5)^T \cdot \frac{d}{dP}(\underline{a}_5 - \underline{C}_5) \cdot \dot{P} = \theta_5 \cdot \frac{d}{d\theta}\theta_5 \cdot \dot{\theta} \quad (33)$$

$$(\underline{a}_5 - \underline{C}_5)^T \cdot \frac{d}{dP}(\underline{a}_5 - \underline{C}_5) \cdot \dot{P} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \theta_5 \end{bmatrix} \cdot \dot{\theta} \quad (34)$$

Expanding the $\underline{A}$ and $\underline{B}$ matrices with Eq. (34), as well as the same result for θ_6 , yields the following result in matrix form:

$$\left[\begin{array}{cccc|cc} (\underline{B}_1 - \underline{C}_1)^T \cdot \frac{\partial \underline{B}_1}{\partial \theta} & 0 & 0 \\ (\underline{B}_2 - \underline{C}_2)^T \cdot \frac{\partial \underline{B}_2}{\partial \theta} & 0 & 0 \\ (\underline{B}_3 - \underline{C}_3)^T \cdot \frac{\partial \underline{B}_3}{\partial \theta} & 0 & 0 \\ (\underline{B}_4 - \underline{C}_4)^T \cdot \frac{\partial \underline{B}_4}{\partial \theta} & 0 & 0 \\ 0 & 0 & 0 & 0 & \theta_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & \theta_6 \end{array} \right] \cdot \dot{\theta} = \left[\begin{array}{c} (\underline{B}_1 - \underline{C}_1)^T \cdot \frac{\partial \underline{C}_1}{\partial P} \\ (\underline{B}_2 - \underline{C}_2)^T \cdot \frac{\partial \underline{C}_2}{\partial P} \\ (\underline{B}_3 - \underline{C}_3)^T \cdot \frac{\partial \underline{C}_3}{\partial P} \\ (\underline{B}_4 - \underline{C}_4)^T \cdot \frac{\partial \underline{C}_4}{\partial P} \\ (\underline{a}_5 - \underline{C}_5)^T \cdot \frac{\partial (\underline{a}_5 - \underline{C}_5)}{\partial P} \\ (\underline{a}_6 - \underline{C}_6)^T \cdot \frac{\partial (\underline{a}_6 - \underline{C}_6)}{\partial P} \end{array} \right] \cdot \dot{P}$$

Checking the matrix dimensions, it is seen that the equation is valid dimension-wise:

$$[6 \times 6] \cdot [6 \times 1] = [6 \times 4] \cdot [4 \times 1]$$

$$[6 \times 1] = [6 \times 1]$$

The new Jacobian is then derived as done in the previous section, and thus $\underline{B}^T \underline{A} = \underline{J}$ should by intend no longer be singular within the feasible poses of the end-effector. To achieve the best possible condition number of the Jacobian, the mounting points of the linear actuators are optimized in the forthcoming.

3 Optimization of Linear Actuator Placement

The stability improvement that the linear actuators provide can be seen in the kinematic Jacobian and its condition number. The following sections deals with this improvement by describing, formulating and solving an optimization problem.

3.1 Description of the Optimization Problem

The goal of optimization is to find the best possible solution according to a defined set of constraints. If the solution to the optimization problem changes according to the initial condition, the found solution is locally optimal. This means that the objective function has to be convex in order to find the optimal solution. [15]

The placement of the actuators should be chosen as the optimal location regarding the condition number of the Jacobian. The attachment of these to the end-effector is chosen which is at the center between arm 1 and 2 for the first one and in the center between arm 3 and 4, for the second one.

Since the hand-off part is where the most precise movement is required, this is the point that the attachment point of the actuators will be optimized in relation to. This is in practice not a single point, but rather a general area where the hand-off operation is performed. In order to simplify the optimization, this is however chosen as a single point. The hand-off pose of the end-effector is chosen as follows:

$$P_{Hand-off} = \begin{bmatrix} 0 \\ 0,5 \\ -0,4 \\ 90^\circ \end{bmatrix} \quad (35)$$

Since all points are not feasible actuator attachment points, a number of constraints are defined. First, a feasible region for the attachment of these actuators is defined in order to obtain a useful solution to the problem. It is decided that the attachments for the actuators should be in close proximity to the defined origin of the model. The chosen constraints for the region are:

$$-0,2 \leq x \leq 0,2 \quad [m] \quad (36)$$

$$-0,2 \leq y \leq 0,2 \quad [m] \quad (37)$$

$$-0,1 \leq z \leq 0 \quad [m] \quad (38)$$

In the figure below the feasible region for the attachment points can be observed. It is illustrated in blue with a coordinate system inserted in the origin to indicate orientation of the region.

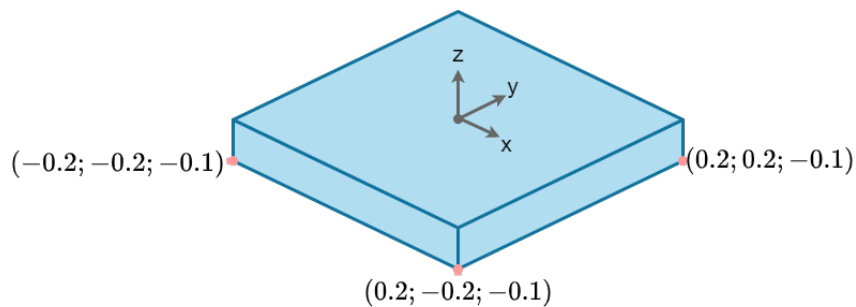


Figure 5: Feasible region for the linear actuator attachment points.

In addition to the above bounds, it is also decided that the linear actuators are not allowed to cross when looking at the yz-plane. It is done to make sure that the actuators will not clash. This is defined as an inequality constraint on the y and z coordinates. Looking at figure 6 it is seen that there are two cases where the arms may cross.

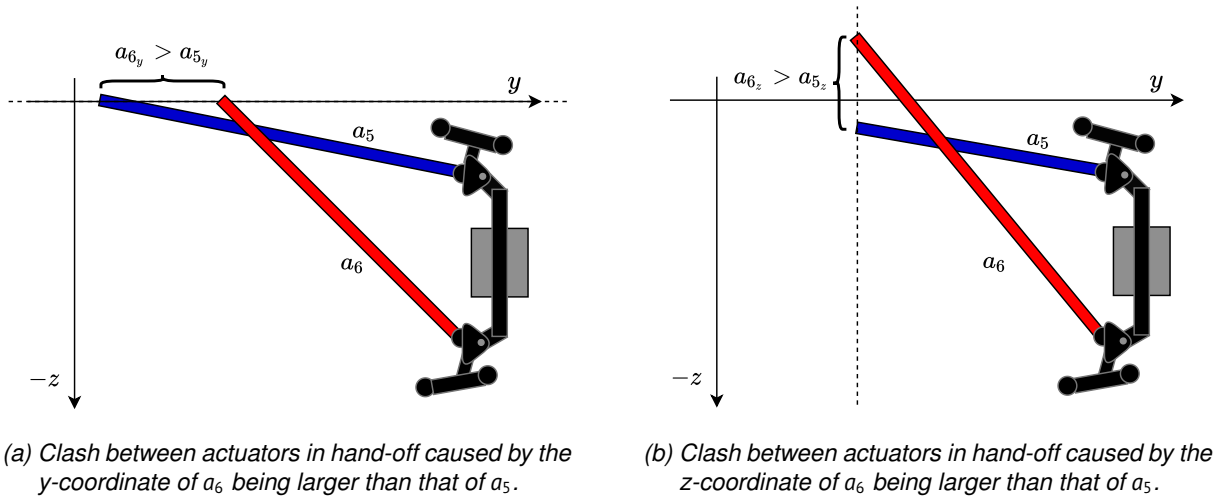


Figure 6: Cases where attachment points results in a clash between the linear actuators. Actuator a_5 has attachment coordinates (a_{5x}, a_{5y}, a_{5z}) , and actuator a_6 attachment coordinates (a_{6x}, a_{6y}, a_{6z}) .

In figure 6a the clash is caused by a_{6y} being larger than a_{5y} . This means that one constraint could be defined as $a_{5y} \geq a_{6y}$. To avoid that the optimization algorithm finds a result where the two attachment points are identical, the constraint will be defined as follows instead:

$$a_{5y} \geq a_{6y} + 0,005 \quad [\text{m}] \quad (39)$$

In figure 6b the clash is caused by a_{6z} being significantly larger than a_{5z} . To avoid this, another constraint will be defined as:

$$a_{5z} \geq a_{6z} \quad [\text{m}] \quad (40)$$

With the above described intuition behind the optimization problem combined with described constraints, the optimization problem can be formulated and solved.

3.2 Formulating the Optimization Problem

The optimization problem is formulated as:

$$\begin{aligned}
 &\text{minimize} \quad \|J\|_2 \cdot \|J^\dagger\|_2 \\
 &\text{subject to} \quad -0,2 \leq a_{5x} \leq 0,2 \\
 &\quad \quad \quad -0,2 \leq a_{5y} \leq 0,2 \\
 &\quad \quad \quad -0,1 \leq a_{5z} \leq 0 \\
 &\quad \quad \quad -0,2 \leq a_{6x} \leq 0,2 \\
 &\quad \quad \quad -0,2 \leq a_{6y} \leq 0,2 \\
 &\quad \quad \quad -0,1 \leq a_{6z} \leq 0 \\
 &\quad \quad \quad a_{5y} \geq a_{6y} + 0,05 \\
 &\quad \quad \quad a_{5z} \geq a_{6z}
 \end{aligned} \quad (41)$$

The first 6 constraints, are constraining each of the coordinates according to figure 5. The last two constraints are ensuring that the actuators will not cross when in the hand off pose.

Furthermore it is also making sure that the found optimal point for each of the arms, will not be the same. The minimum is found using MATLAB®'s optimization toolbox [16]. This toolbox has been utilized using `fmincon`. As it was stated earlier, the optimal solution is found if the objective function is convex. Verifying that the objective function is convex is a complex task. The optimization problem has been solved using different initial conditions in order to make sure that the function is convex. In all of these tests, the solution was the same so it is presumed that the function is convex thus the found solution is optimal. [15]

3.3 Results of Solving the Optimization Problem

The optimization problem is solved using MATLAB® and an optimization function called `fmincon` that uses interior-point methods to find a minimum for a given optimization problem [14]. In the below figure, the operating pose for the end-effector is depicted.

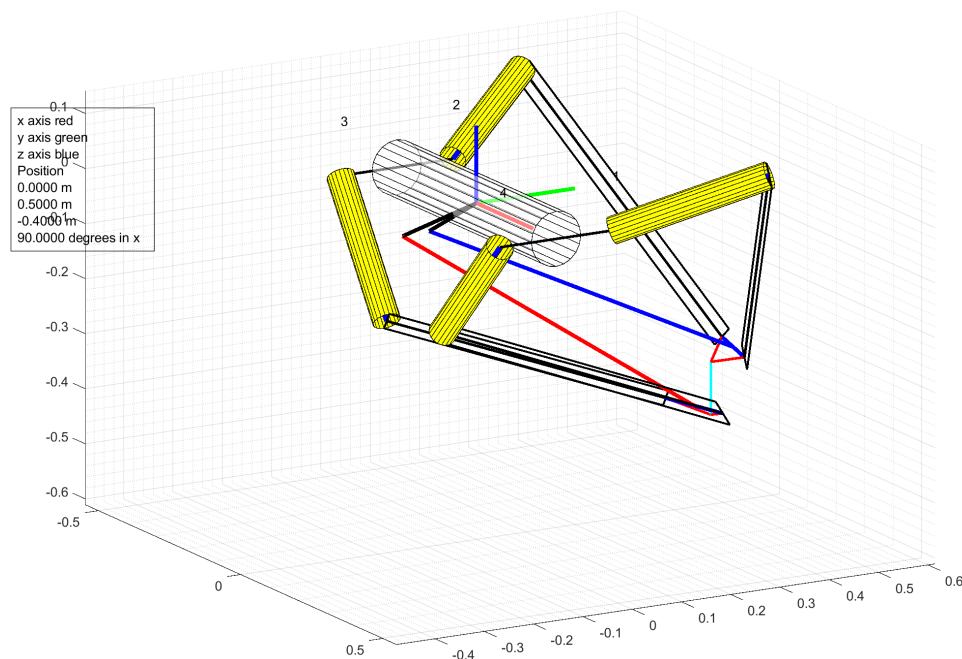


Figure 7: Ragnar robot in the operating pose with the linear actuators attached.

When solving the optimization problem it is found that the condition number can be reduced to be approximately 41 for a given actuator position in the feasible region. The condition number and attachment point coordinates of the linear actuators are presented in the following table.

Optimization Results - <code>fmincon</code>								
	Actuator 1			Actuator 2			Results	
Variable	a_{5x}	a_{5y}	a_{5z}	a_{6x}	a_{6y}	a_{6z}	Cond. Nr.	Norm
Value	0,0376	-0,1091	-0,0370	0,0022	-0,1599	0,0374	40,717	12,570

Table 1: Results from optimization, both coordinates and the obtainable condition number. The values are in meters related to the origin.

For the found configuration for the robot, it is decided to investigate the condition number in the pick-up pose as well, as this is a relevant pose to consider. The pose is depicted in the figure

8.

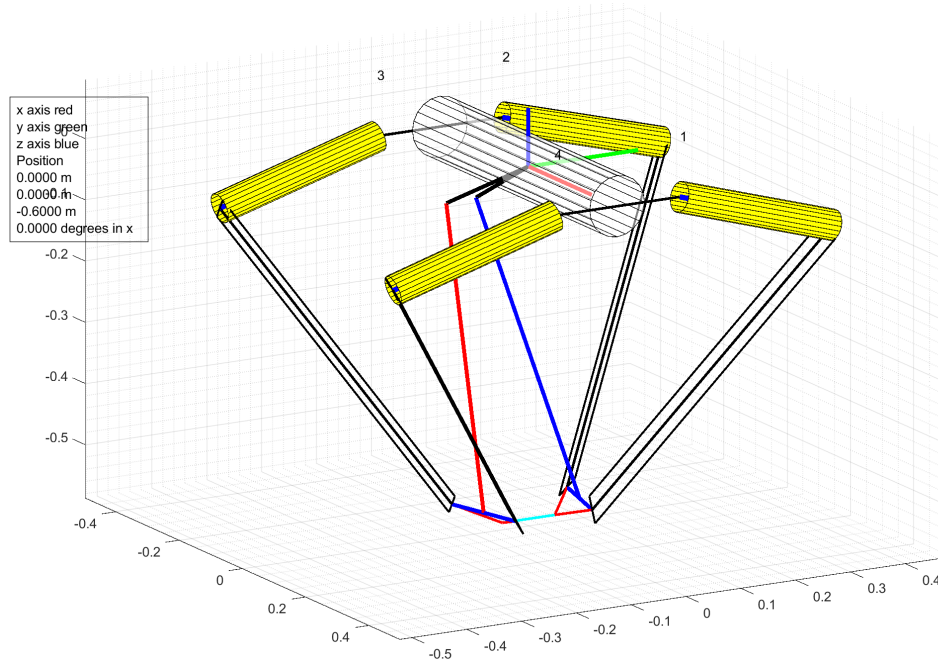


Figure 8: Ragnar robot in the pick-up pose with the linear actuators attached.

The found condition number for the pick-up pose, see figure 8, with the above specified attachment points the condition number in the pick-up pose is approximately 55,7. The condition number in the pick-up pose is a result of the optimization problem solved where the operating pose was the only pose considered. For a second attempt for an optimal solution it is decided to solve the optimization problem once more, but this time considering both poses, the pick up and hand-off pose.

$$P_{Hand-off} = \begin{bmatrix} 0 \\ 0,5 \\ -0,4 \\ 90^\circ \end{bmatrix}, P_{Pick-up} = \begin{bmatrix} 0 \\ 0 \\ -0,6 \\ 0^\circ \end{bmatrix} \quad (42)$$

With the specified end-effector poses, the optimization is executed.

3.4 Multi-Objective Constrained Optimization of Actuator Attachment Points

There are multiple methods for multi-objective optimization in MATLAB®. The `fminimax` approach is chosen. The optimization problem is then defined as:

$$\begin{aligned}
 &\text{minimize} && f_1, f_2 \\
 &\text{subject to} && -0,2 \leq a_{5x} \leq 0,2 \\
 & && -0,2 \leq a_{5y} \leq 0,2 \\
 & && -0,1 \leq a_{5z} \leq 0 \\
 & && -0,2 \leq a_{6x} \leq 0,2 \\
 & && -0,2 \leq a_{6y} \leq 0,2 \\
 & && -0,1 \leq a_{6z} \leq 0 \\
 & && a_{5y} \geq a_{6y} + 0,05 \\
 & && a_{5z} \geq a_{6z}
 \end{aligned} \tag{43}$$

Where,

$f_1, f_2 = \|\underline{J}\|_2 \cdot \|\underline{J}^t\|_2$ for both the pick-up and hand-off poses.

Solving this optimization problem yields the following results:

Optimization Results - <code>fminimax</code>								
	Actuator 1			Actuator 2			Results	
Variable	a_{5x}	a_{5y}	a_{5z}	a_{6x}	a_{6y}	a_{6z}	Cond(f1)	Cond(f2)
Value	0,0100	0,0250	-0,1000	0,0100	-0,0250	-0,1000	45,56	50,01

Table 2: Table of optimization result. Coordinate values are in meters, relative to the origin.

A plot of ragnar with these actuator attachment points in hand-off and pick-up pose, can be seen in figures 9 and 10.

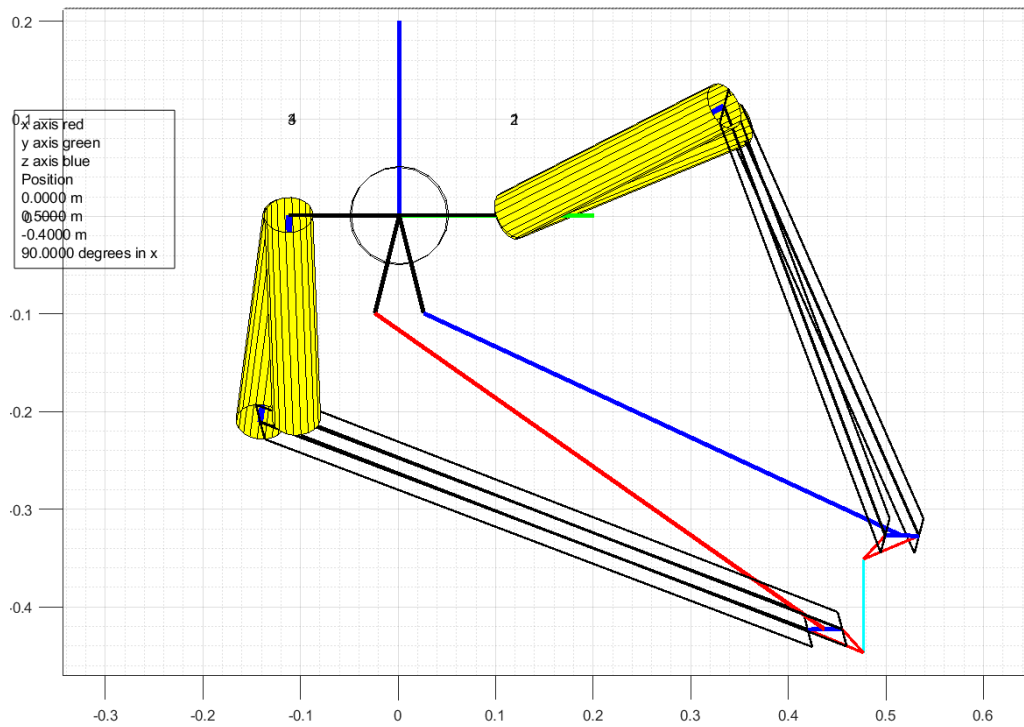


Figure 9: Ragnar robot in the hand-off pose (f1), using the actuator attachment points in table 2.

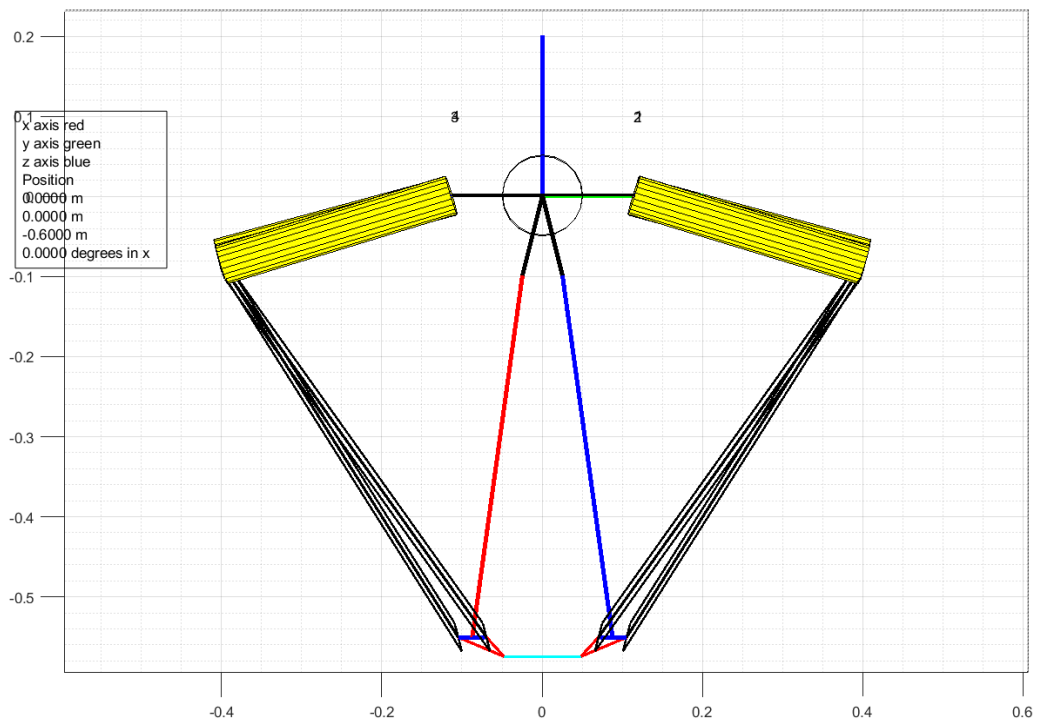


Figure 10: Ragnar robot in the pick-up pose (f2), using the actuator attachment points in table 2.

Since this arm setup has low condition numbers in both pick-up and hand-off pose as well as a feasible construction, this arm setup will be used moving forward.

3.5 Confirmation of Singularity Removal

A previous group's research found and showcased that the Jacobian is singular in the hand-off pose [4]. Mathematically this can be seen through the condition number of the Jacobian converging to infinity when following the trajectory from pick-up to hand-off. A remake of the same test, where the condition number of the Jacobian is calculated through the same trajectory, can be seen in figure 11.

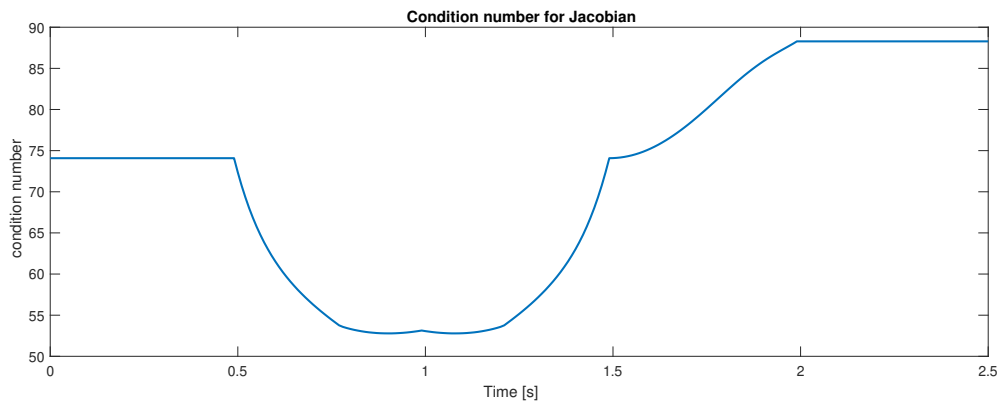


Figure 11: Condition number of Jacobian through transportation trajectory after modified mechanical construction.

It should be noted that the same trajectory is followed as used in [4]. However the new result was realised with a linear trajectory, where the previous was a polynomial, see figure 12.

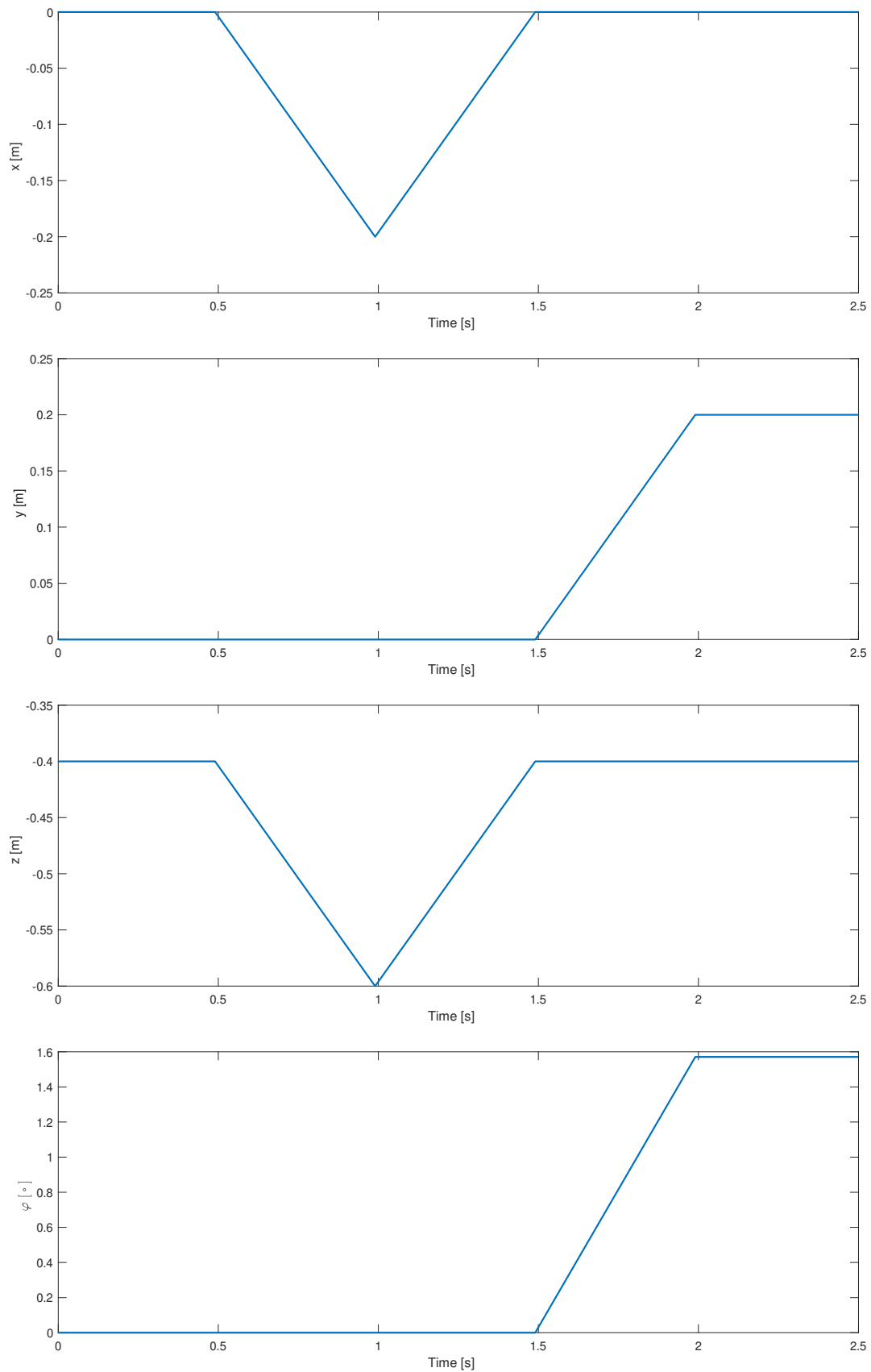


Figure 12: Trajectory used for calculation of conditioning number of jacobian.

The previous result published in [4] is depicted below.

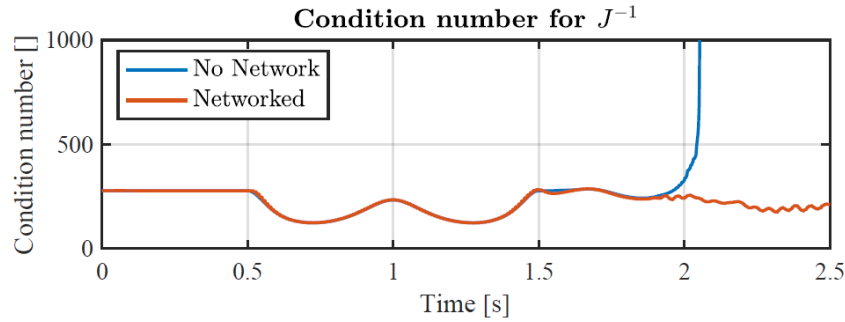


Figure 13: Condition number of Jacobian through transportation trajectory result from previous research group [4]

Comparing the two, it can be observed that the revised mechanical construction lowers the condition number to be under 100 as well as removes the singularity.

4 State Space Model

In order to make a controller for Ragnar, a model needs to be derived. In this project it has been chosen to model the system as a state space model [7]. It is however also desired to implement compliant motion into the system. In order to accomplish this, the state space model and a compliant motion controller has to be combined. This will be done by designing the compliant motion control first and then use this as the input to the state space model.

4.1 Compliant Motion

The compliant motion controller is designed in task space. The environmental interaction of the compliance controller can be modelled as:

$$\underline{F}_e = \underline{K}_p \cdot \underline{e}_x + \underline{K}_d \cdot \dot{\underline{e}}_x + \underline{M} \cdot \ddot{\underline{e}}_x \quad (44)$$

Where,

$\underline{K}_p$ and $\underline{K}_d$ are the diagonal stiffness and damping matrices with each entry for each DOF.

$\underline{M}$ is the mass matrix of the end-effector.

$\underline{F}_e$ is the force from the environment onto the end-effector.

Which is a mass-spring-damper system. The sum of forces on the end-effector can be described according to Newton's second law.

$$\Sigma \underline{F} = \underline{M} \cdot \ddot{\underline{e}}_x \quad (45)$$

$$\underline{F}_a + \underline{F}_e + \underline{F}_g = \underline{M} \cdot \ddot{\underline{e}}_x \quad (46)$$

$$\underline{F}_e = \underline{M} \cdot \ddot{\underline{e}}_x - \underline{F}_a - \underline{F}_g \quad (47)$$

Where,

$\underline{F}_a$ is the actuator force

$\underline{F}_g$ is the gravitational force

Inserting Eq. (44) into Eq. (47) yields:

$$\underline{\underline{K}}_p \cdot \underline{e}_x + \underline{\underline{K}}_d \cdot \dot{\underline{e}}_x + \underline{M} \cdot \ddot{\underline{e}}_x = \underline{M} \cdot \ddot{\underline{e}}_x - \underline{F}_a - \underline{F}_g \quad (48)$$

$$\underline{\underline{K}}_p \cdot \underline{e}_x + \underline{\underline{K}}_d \cdot \dot{\underline{e}}_x = -\underline{F}_a - \underline{F}_g \quad (49)$$

$$\underline{F}_a = -\underline{F}_g - (\underline{\underline{K}}_p \cdot \underline{e}_x + \underline{\underline{K}}_d \cdot \dot{\underline{e}}_x) \quad (50)$$

It is seen in equation 50 that in addition to the negative feedback $-(\underline{\underline{K}}_p \underline{e}_x + \underline{\underline{K}}_d \dot{\underline{e}}_x)$, gravitation is present. It is chosen to neglect the gravitational pull, $-\underline{F}_g$. This is because, the state space model of the system will become non-linear as a result of the gravity being non-linear. Since a state space model is a linear representation of a dynamic system, this will not work.

4.2 State Space Modeling

A state space model has the following representation:

$$\dot{\underline{x}} = \underline{A}\underline{x} + \underline{B}\underline{u} \quad (51)$$

$$\underline{y} = \underline{C}\underline{x} + \underline{D}\underline{u} \quad (52)$$

Where $\underline{u}$ is the input, $\underline{x}$ is a vector of system states, and $\underline{y}$ is the output.

To create a state space model, the states $\underline{x}$, of the system have to be defined. The amount of states are often found as the number of "energy storages" in a system, while the state variables within $\underline{x}$ are the variables that describe these energies. In Ragnar's case, the energy in the system is present in the end-effector. Since this is a mass it contains both potential and kinetic energy which is present in all the possible directions that the mass can move in. Thus, it has a total of eight states consisting of four position states $\underline{P}$, and four velocity states $\underline{\dot{P}}$, seen as follows:

$$\underline{x} = \left[\underbrace{\begin{bmatrix} x & y & z & \varphi \end{bmatrix}}_{\underline{P}} \underbrace{\begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\varphi} \end{bmatrix}}_{\underline{\dot{P}}} \right]^T \quad (53)$$

Where,

$\underline{P}$ is the pose of the end-effector.

This makes Ragnar an eighth-order system as it has eight states.

Now the matrix $\underline{A}$ should consist of the dynamics of these states, which could be done with Lagrange modelling for instance. However, instead of modelling the dynamics, the hand-off pose is chosen as an operating pose. This means that $\underline{A}$ can be constructed in a much simpler way where the actuators are assumed to have infinite speed, or in other words, that they are able to reach any position instantaneously. This is feasible since the model is used for the hand-off pose, where speed is not of interest. The state-variable equations of the end-effector pose will then be as follows:

$$\frac{d}{dt}\underline{P} = \underline{\dot{P}} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\varphi} \end{bmatrix}^T \quad (54)$$

Which will result in the following:

$$\frac{d}{dt} \begin{bmatrix} \underline{P} \\ \underline{\dot{P}} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & I_{4 \times 4} \\ 0 & 0 \end{bmatrix}}_{8 \times 8} \cdot \begin{bmatrix} \underline{P} \\ \underline{\dot{P}} \end{bmatrix} + \underline{B} \cdot \underline{u} \quad (55)$$

The next four state equations are of velocity states. To derive these, Newton's second law is used for the general coordinates. The actuator forces in the platform is the wrench $F_a = [F_x, F_y, F_z, \tau_\varphi]^T$. Thus, it results in the following double derivative state-variable equations:

$$\frac{d}{dt} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\varphi} \end{bmatrix} = \underbrace{\begin{bmatrix} m_p^{-1} & 0 & 0 & 0 \\ 0 & m_p^{-1} & 0 & 0 \\ 0 & 0 & m_p^{-1} & 0 \\ 0 & 0 & 0 & I_x^{-1} \end{bmatrix}}_{\underline{M}^{-1}} \cdot \underbrace{\begin{bmatrix} F_x \\ F_y \\ F_z \\ \tau_\varphi \end{bmatrix}}_{\underline{F}_a} \quad (56)$$

Where,

m_p is the mass of the end-effector platform.

I_x is the end-effector's inertia around the x-axis.

Combining Eq. (55) and Eq. (56) results in the following state space model of the system:

$$\frac{d}{dt} \begin{bmatrix} \underline{P} \\ \underline{\dot{P}} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & I_{4 \times 4} \\ 0 & 0 \end{bmatrix}}_{8 \times 8} \cdot \begin{bmatrix} \underline{P} \\ \underline{\dot{P}} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \underline{M}^{-1} \end{bmatrix}}_{8 \times 4} \cdot \underline{F}_a \quad (57)$$

Expanded:

$$\frac{d}{dt} \begin{bmatrix} \underline{P} \\ \underline{\dot{P}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ \varphi \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\varphi} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0,8 & 0 & 0 & 0 \\ 0 & 0,8 & 0 & 0 \\ 0 & 0 & 0,8 & 0 \\ 0 & 0 & 0 & 2180 \end{bmatrix} \cdot \begin{bmatrix} F_x \\ F_y \\ F_z \\ \tau_\varphi \end{bmatrix} \quad (58)$$

This means that the input to our system is the desired force on the platform $\underline{F}_a$. The output of this system is chosen to be the states. As a result of the encoders built into the stepper motors, it is possible to measure both position and velocities [11]. Doing so, the matrix $\underline{C}$ becomes an 8×8 identity matrix:

$$\underline{y} = \underline{C} \cdot \underline{x} = [I_{8 \times 8}] \cdot \begin{bmatrix} \underline{P} \\ \underline{\dot{P}} \end{bmatrix} \quad (59)$$

$$\underline{y} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ \varphi \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\varphi} \end{bmatrix} \quad (60)$$

The state space model for the system does thus consist of Eq. (57) and Eq. (59).

5 Controller Design

In this chapter, it is first checked if the system is controllable, and secondly state feedback is designed for the system, as a first step towards controlling the robot in the chosen operating pose, namely the hand-off pose. A set of poles are chosen, which are to be placed such that compliant control is achieved. Lastly an observer is also designed to estimate states, in case of packet loss.

5.1 Controllability

Before designing a control scheme for a system, it is necessary to check if the system is controllable. To check this property, the controllability matrix is constructed, which consist of the input matrix $\underline{\underline{B}}$ and the system matrix $\underline{\underline{A}}$. To determine if a system is controllable, its controllability matrix will have to fulfill the following [7]:

$$\text{rank} \left(\begin{bmatrix} \underline{\underline{B}} & \underline{\underline{A}}\underline{\underline{B}} & \underline{\underline{A}}^2\underline{\underline{B}} & \underline{\underline{A}}^3\underline{\underline{B}} & \underline{\underline{A}}^4\underline{\underline{B}} & \underline{\underline{A}}^5\underline{\underline{B}} & \underline{\underline{A}}^6\underline{\underline{B}} & \underline{\underline{A}}^7\underline{\underline{B}} \end{bmatrix} \right) = n \quad (61)$$

Where:

n is the number of states.

Inserting $\underline{\underline{B}}$ and $\underline{\underline{A}}$ derived in the previous chapter, it is observed that the system is indeed controllable.

5.2 State Feedback

For the case of a controllable system, state feedback can be designed to yield a desired system behaviour. First the poles that yield the desired system behaviour are selected.

The control in the hand-off pose needs to be compliant, as it is important that the poultry is not damaged by the robot. It is decided to model the compliance after a hand, since Ragnar is to replace manual labour. As derived in section 4.1 the system input giving compliant motion F_a consist of stiffness and damping constants. It is decided to design the system with critical damping.

The mass-spring-damper system can be rewritten as an oscillatory second-order system and is defined as [8]:

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2x = 0 \quad (62)$$

With:

$$\omega_n = \sqrt{\frac{K_{p,i,j}}{M_{i,j}}} \quad (63)$$

Through Laplace transforming and using critical damping $\zeta = 1$, the system will be dependent on $\underline{\underline{K_p}}$ only. The poles can be obtained using the following:

$$s^2 + 2\sqrt{\frac{K_{p,i,j}}{M_{i,j}}}s + \frac{K_{p,i,j}}{M_{i,j}} = 0 \quad (64)$$

Giving the roots:

$$s = -\sqrt{\frac{K_{p,i,j}}{M_{i,j}}} \pm \sqrt{0} \quad (65)$$

It should be noted that for each of the DOFs, x, y, z, φ , there will be a mass, and spring constant. Performing the calculation in Eq. (65) for each DOF, yields a double pole, as seen.

It is decided, after conducting an experiment, to put the spring constant $k = 250 \text{ N/m}$ for x, y and z . The spring constant was found based on a 1 cm displacement of a finger. For φ , the measured force of 2,5 N from the experiment, was multiplied by the length of the finger, thus $\tau = F \cdot l_f$. Since the finger is measured to $l_f = 6 \text{ cm}$, the torque becomes $\tau = 2,5 \cdot 0,06 = 0,15 \text{ N m}$. From the matrix $\underline{M}$ introduced in Eq. (56), the end-effector mass m_p in x, y, z is 1,28 kg and the inertia I_x is 0,004 267 kgm^2 . When calculating the poles for φ , the spring constant and mass is replaced by τ and I_x respectively.

Performing the calculations in Eq. (65), yields the following poles:

$$\underline{p}_{SF} = [-13,98 \quad -13,98 \quad -13,98 \quad -13,98 \quad -13,98 \quad -13,98 \quad -5,32 \quad -5,32] \quad (66)$$

Where:

$\underline{p}_{SF}$ contains the desired pole locations for the state feedback.

Now that the wanted pole locations are found, the gain matrices $\underline{K}_p$ and $\underline{K}_d$ are obtainable. These will be designed to give the system the wanted poles. Implementation of state feedback can be seen on figure 14.

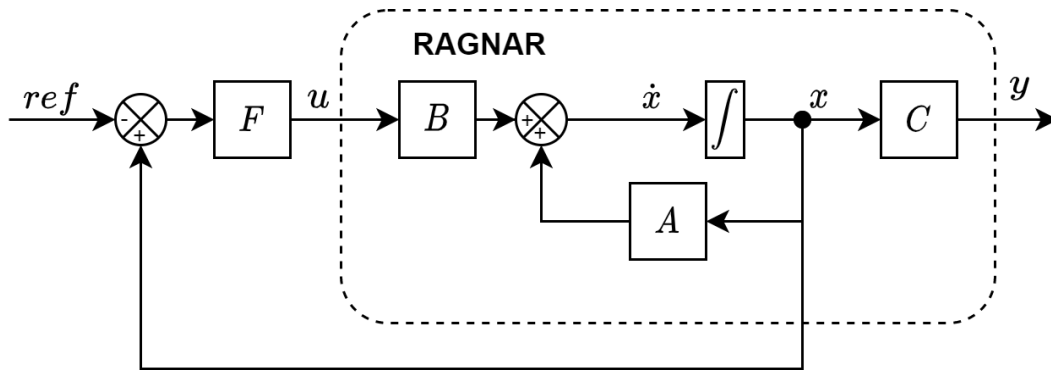


Figure 14: State feedback structure block diagram . [7]

State feedback is defined as $\underline{u} = \underline{F}\underline{x}$, thus the state space model can be rewritten based on the designed feedback $\underline{F}$. The model becomes:

$$\dot{\underline{x}} = \underline{A}\underline{x} + \underline{B}\underline{u} \quad (67)$$

$$= \underline{A}\underline{x} + \underline{B}\underline{F}\underline{x} \quad (68)$$

$$= (\underline{A} + \underline{B}\underline{F})\underline{x} \quad (69)$$

One design method for state feedback is to use controllable canonical form, however this only functions on SISO-systems. To force this method onto a MIMO-system $\underline{B}$ can be transformed only taking one input with, $\bar{\underline{B}} = \underline{B} \cdot \underline{R}$, hereby allowing to control all inputs with only one input, mimicking a SISO-system. Another method is to use the characteristic closed-loop polynomial and desired characteristic equation.

The characteristic closed-loop polynomial of the system is [7]:

$$\det(sI - (\underline{A} + \underline{B}\underline{F})) = 0 \quad (70)$$

The roots of $(\underline{A} - \underline{B}\underline{F})$, indicated with λ are equivalent to the pole locations. Indicating the following desired characteristic equation

$$\alpha_c(s) = (s - p_{SF_1})(s - p_{SF_2}) \dots (s - p_{SF_8}) = 0 \quad (71)$$

Hence the elements of state feedback $\underline{F}$ can be obtained by matching coefficients. Hereby the actual system characteristic equation becomes identical to the desired characteristic equation thus giving the desired pole locations [7].

The elements of state feedback $\underline{F}$ are found by the use of the `place`-command in MATLAB®. The command in MATLAB® picks $\underline{F}$ that gives the the desired eigenvalues, equivalent to poles, in $(\underline{A} + \underline{B}\underline{F})$ [12].

Doing so yields the following $\underline{F}$ matrix which is the controller containing the required gains to obtain the wanted poles. It shall be noted that to get negative feedback, the signs of $\underline{F}$ are changed.

$$\underline{F} = \begin{bmatrix} -K_p & -K_d \end{bmatrix} \quad (72)$$

$$\underline{F} = \begin{bmatrix} -250,50 & 0 & 0 & 0 & 35,81 & 0 & 0 & 0 \\ 0 & -250,50 & 0 & 0 & 0 & -35,81 & 0 & 0 \\ 0 & 0 & -250,50 & 0 & 0 & 0 & -35,81 & 0 \\ 0 & 0 & 0 & -0,034 & 0 & 0 & 0 & -0,0089 \end{bmatrix} \quad (73)$$

Where:

$\underline{F}$ is state feedback matrix containing the required gains.

The found gains are then used for the system. The next step is to design a full-order observer.

5.3 Full-Order Observer

The full-order observer is designed based on a canonical form as well. Instead of a controllability matrix, it is instead an observability matrix that is formed. This is, however done from the same matrices as the controllability matrix is. The duality between the two makes the design process almost identical, with differences being the newly introduced observer gain $\underline{L}$ which insures the pole placements of the observer. Furthermore, the feedback is now taken from the estimated states, as seen from figure 15, $\underline{u}$ is now $\underline{u} = \underline{F}\hat{\underline{x}}$, instead of the previous $\underline{u} = \underline{F}\underline{x}$. [7]

5.3.1 Full-Order Observer - Design

The full-order observer method is based on a model of the system, where the goal is to estimate the states in the system. When both measurements and estimates are available, an error can be found and used to correct the observer. However, when a measurement is unavailable the full-order observer can be used as a predictor for the state of the system. This means that in the case of packet loss the observer is used as replacement for the system. [7]

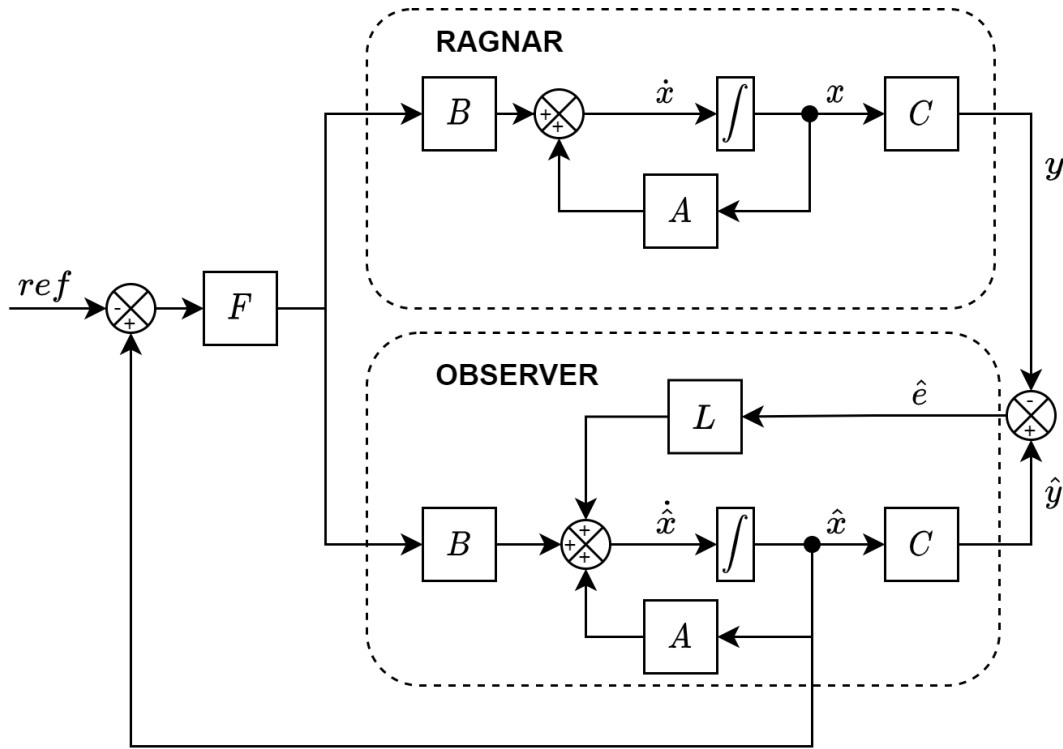


Figure 15: Full-order observer block diagram.

For the observer design, some additional elements are added to the block diagram, as seen from figure 15. Some of the additions are an exact copy of the system, which is the model that is to provide the estimations of the states. The block containing an $\underline{L}$ is the block containing the observer gain. A rule of thumb for designing these gains is that the eight poles placed with the full-order observer, are to be placed such that they are 2-6 times faster than the state feedback poles [7]. The poles for the full-order observer is decided to be placed five times faster than the system poles p_{SF} . This yields the following poles:

$$\underline{p}_{FO} = [-69,88 \quad -69,88 \quad -69,88 \quad -69,88 \quad -69,88 \quad -69,88 \quad -26,60 \quad -26,60] \quad (74)$$

Where:

$\underline{p}_{FO}$ is the desired pole locations for the full-order observer.

Now having the observer as well, the equations describing the combined system has been expanded with following set of equations [7].

$$\dot{\underline{\hat{x}}} = \underline{A} \underline{\hat{x}} + \underline{B} \underline{u} + \underline{L}(\underline{C} \underline{\hat{x}} - \underline{y}) \quad (75)$$

$$\underline{\hat{y}} = \underline{C} \underline{\hat{x}} \quad (76)$$

$$\underline{u} = \underline{F} \underline{\hat{x}} \quad (77)$$

Putting the equation for estimated states (eq76) together with the equation for real states

(eq67), the observer error can be derived as follows [7]:

$$\dot{e} = \dot{\hat{x}} - \dot{x} \quad (78)$$

$$= \underline{A}\hat{x} + \underline{B}\underline{F}\hat{x} + \underline{L}(\underline{C}\hat{x} - y) - (\underline{A}x + \underline{B}\underline{F}\hat{x}) \quad (79)$$

$$= (\underline{A} + \underline{L}\underline{C})(\hat{x} - x) \quad (80)$$

$$= (\underline{A} + \underline{L}\underline{C})e \quad (81)$$

Combining the equations from the state feedback with the full-order observer will yield the following set of equations. [7]

$$\begin{bmatrix} \dot{x} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} \underline{A} + \underline{B}\underline{F} & \underline{B}\underline{F} \\ 0 & \underline{A} + \underline{L}\underline{C} \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix} \quad (82)$$

Based on p_{FO} MATLAB®'s `place`-command is used to find $\underline{L}$ giving the desired eigenvalues of $(\underline{A} + \underline{L}\underline{C})$ [12].

$$\underline{L} = \begin{bmatrix} -69,88 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & -69,88 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -69,88 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -69,88 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -69,88 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -69,88 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -26,60 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -26,60 \end{bmatrix} \quad (83)$$

Now having described the system with a full-order observer included, the next step is to perform simulations of the controlled system, with and without the observer.

5.4 State Feedback Controller

To test the designed controller, Simulink® is utilised to perform a simulation of the controlled system. The simulation setup used is presented in figure 16.

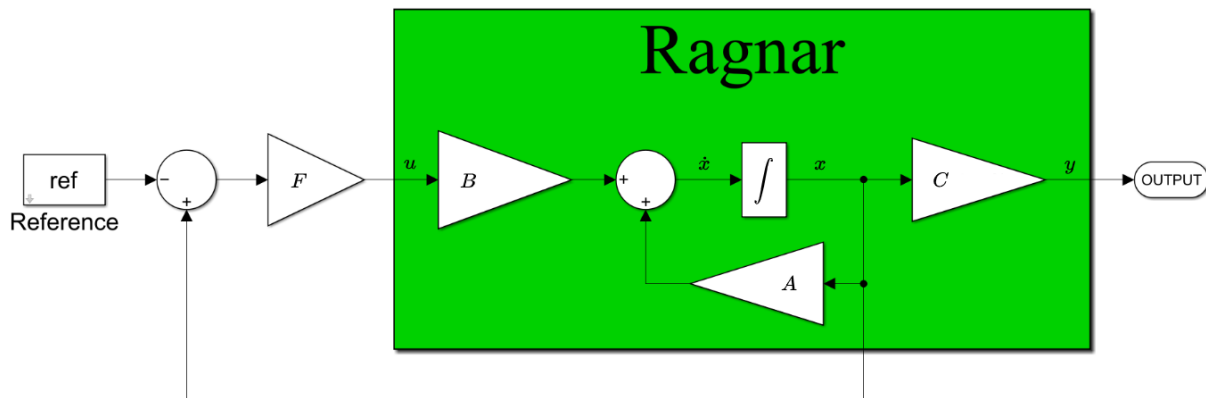


Figure 16: Block diagram configuration used in Simulink® to simulate Ragnar with state feedback.

The reference is chosen as the operating pose, which is where the system is initiated and then a small disturbance is applied. This is done in order to check system behaviour in close proximity to the operating pose. The response of the system is then observed. The input signal can be seen in figure 17.

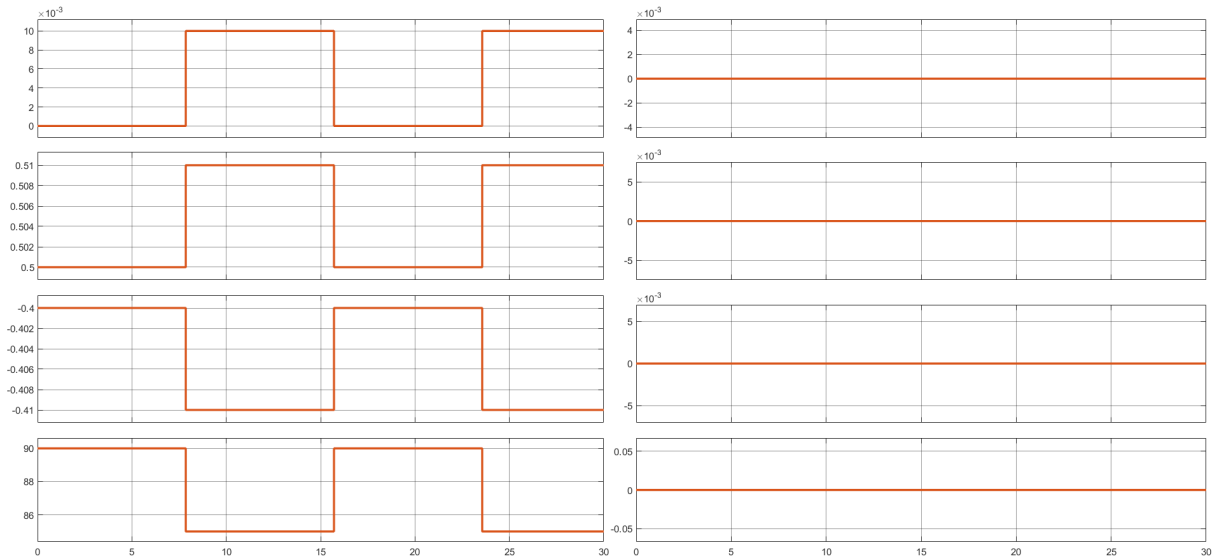


Figure 17: Input signal to state feedback simulation of Ragnar.

The input to the four pose states x , y , z and φ are in the first column of figure 17, respectively. The second column of the input signal is made up of the velocity states, $\dot{x}$, $\dot{y}$, $\dot{z}$ and $\dot{\varphi}$.

It can be noted that the input signal is a repeating step response in the position states, as it is a square signal. The velocity states get a zero input, as the end pose should be a fixed pose with zero velocity.

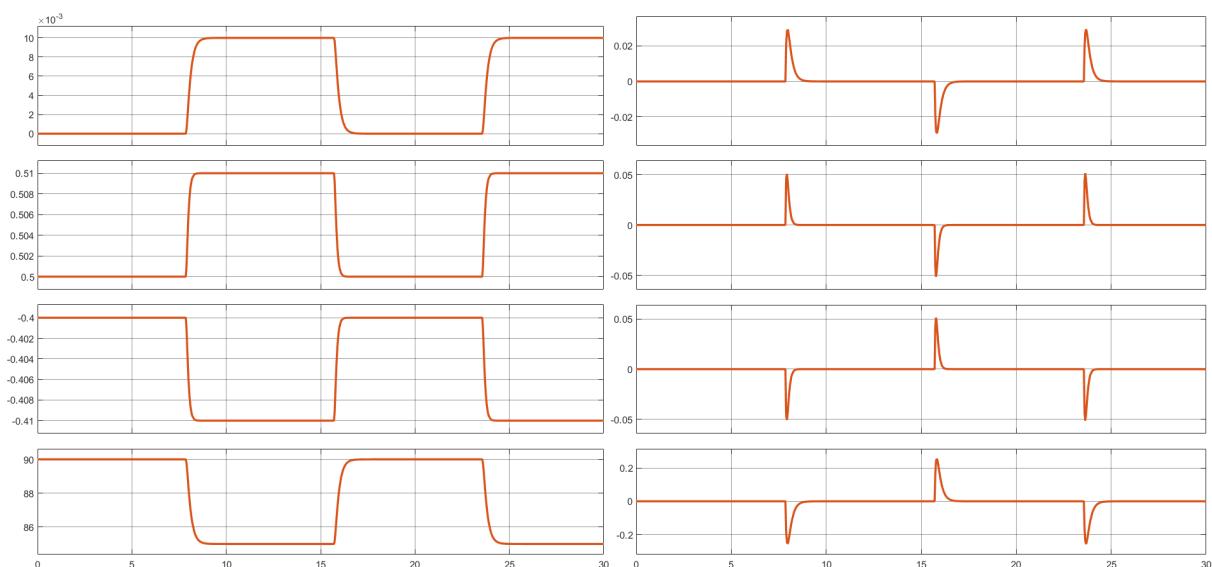


Figure 18: Output signal from state feedback simulation of Ragnar.

The output signal above, is separated in the same manner as the input signal. First considering

the pose outputs, it is seen that each of the states follows the input signal furthermore, the system has critical damping as intended in the design, thus no overshoot is observed. In the velocity outputs, it is noted that there are steps as well. This is due to the changes in reference pose that results in a velocity, while correcting the pose.

Overall considering the observed output, it is concluded that the controller is performing satisfactory. With a seemingly feasible state feedback simulated, the next step is to simulate the characteristics of the designed full-order observer.

5.5 Full-Order Observer

A simulation is set up in order to examine the performance of the full-order observer. The simulation is executed using Simulink® and the setup can be seen below in figure 19.

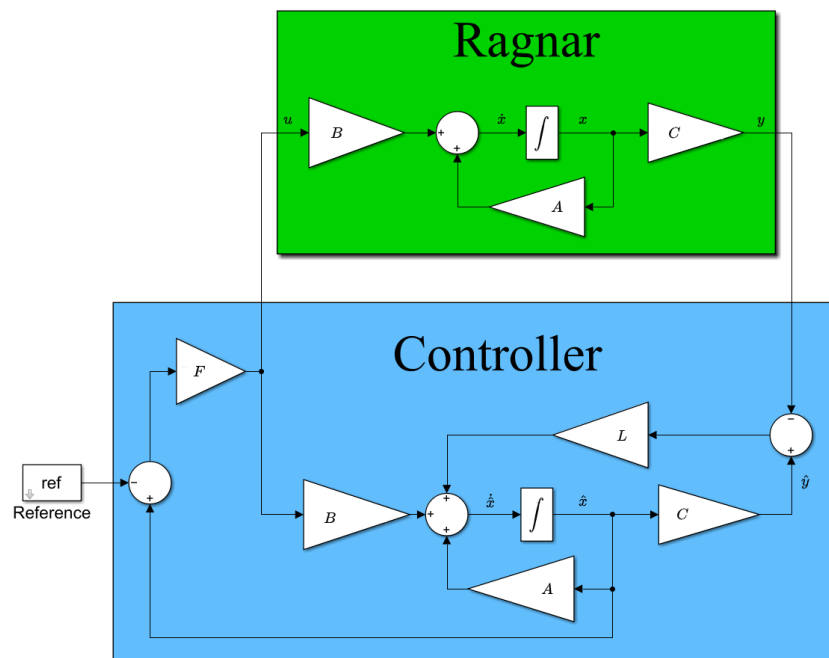


Figure 19: Simulink setup for simulation of the system with a full-order observer. The green box in the top is the model for Ragnar. The blue box contains the observer model and observer gain.

The Ragnar model is still contained in the green box and the rest, contained in the blue box marked as the controller, is the observer model. The input used for this simulation is the same as for the previous simulation shown in figure 17.

Figure 20 shows a comparison of the difference between the output from the state feedback and the observed output. In order to observe any differences, the observer is initiated in zero for all states, whereas the state feedback output is initiated close to the operating pose.

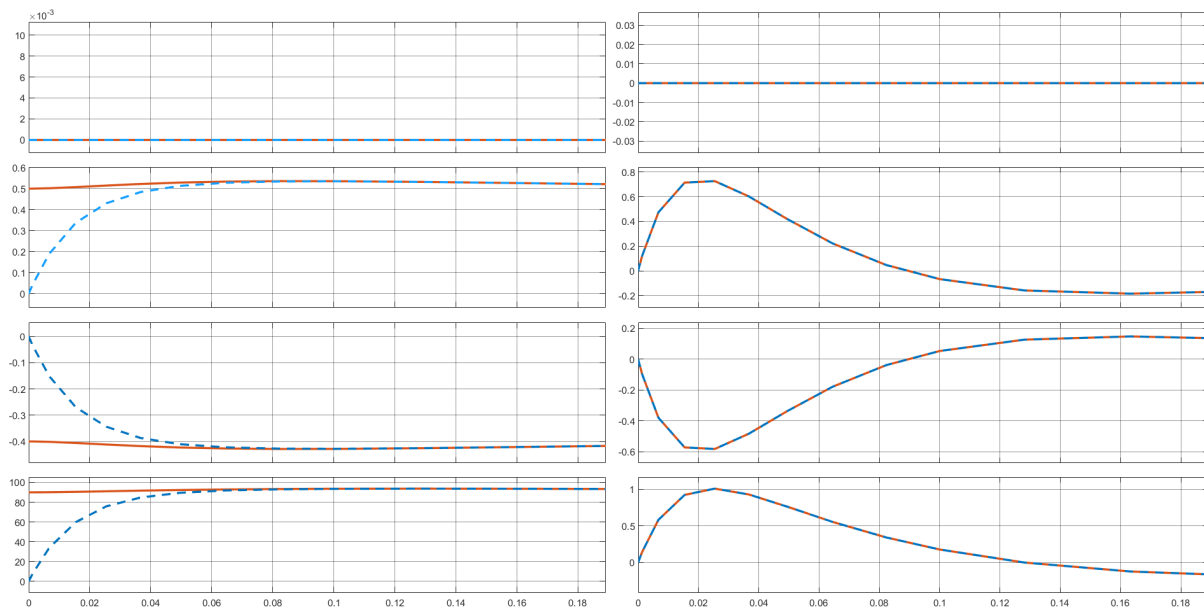


Figure 20: Comparison of the output from Ragnar and from the observer. Blue curves are the outputs from the observer. The orange curves are outputs from Ragnar.

The above figure shows the two systems reacting to the disturbance, where it is seen that the observer quickly removes the error and thus converges to the system output. The poles of the observer is placed five times higher to intentionally make it faster than the system. When the system calculates the control signal on startup, based on the observer states, the result will falsely be that the system is not in the reference pose, since the observer is not initiated in the correct pose. But since the observer and the system receives the same control signal, the actual system will start to move even though it is situated in the correct pose, when the observer starts to correct itself. This is where the advantage of a fast observer shows, since the observer is able to reach the actual output, before a significant change to the output has been made. When the observer reaches the system output, the actual output corrects itself back into the operating pose. Consequently showcasing the effect of the observer gain is as intended. The correction is completed in less than 0,1 s. Afterwards they both follow the input signal in an identical manner, as expected.

When investigating the presented results, it is important to note that only the end-effector mass has been modelled and not the mass of the links, nor the speed of the actuators. That means that it is uncertain whether or not Ragnar can be controlled as fast as assumed in the simulations.

6 Network Simulation

6.1 CAN-bus Simulation

CAN-bus is an industrial field-bus used for real-time communication. It is a message-based protocol which does not require a host computer. CAN-bus can be used with two frame formats; base frame format with 11 bits identifier and extended frame format with 29 bits identifier. The IDE bit must be set to one in the control field to use the extended frame format. [9] An illustration of the whole frame can be seen in figure 21.

The priority of messages in a CAN-bus frame is determined by the node identifier where nodes with lower ID have higher priority. The data stored in each frame varies from 0 to 8 bytes depending on the data length specified in the control field. [9]

CANopen uses CAN-bus as a physical layer but with some changes to the CAN-bus frame. In the base frame format, the 11 bit identifier is divided into a 4-bit function code and 7-bit node ID, which can be seen in figure 21. The node ID is known as Communication Object Identified (COB-ID). The function code is used to define the type of message sent (SYNC, PDO, SDO, ...). In CANopen the priority is determined by the criticality of the functions and after that by the COB-ID. [9]

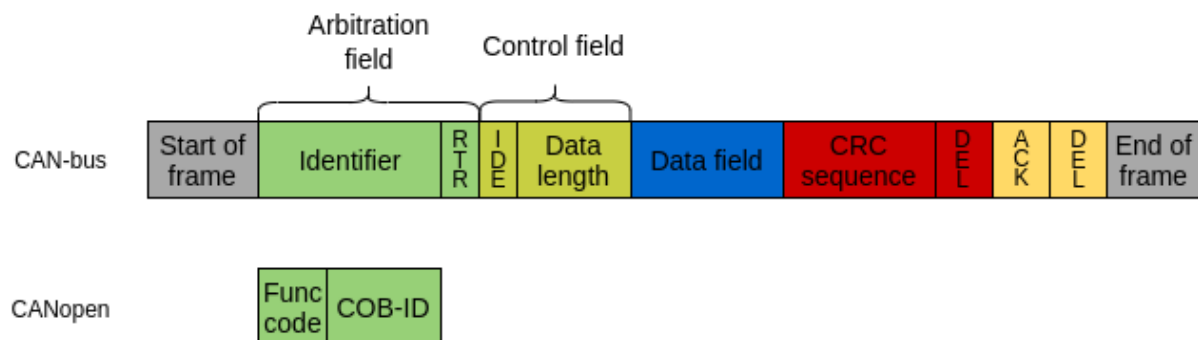


Figure 21: The difference between CAN-bus frame and CANopen frame [9].

The robot is equipped with 4 rotational and 2 translational actuators that are used to move the end-effector. These actuators are controlled using the CANopen communication protocol and are connected to a CAN-bus network. The actuators have a finite resolution of 10 bits, where in the case of the rotational actuators it corresponds to 360 degrees rotation per 1024 steps whereas in the translational actuators it corresponds to 1 cm per 1024 steps. [11]

The resolution have been added to the simulation to make sure that the measurements from the different nodes comply with it. This allows for a more accurate representation of the data sent over the network as this ensures that the data received by the pose nodes are more representative of what their values would be in reality.

TrueTime has been used to simulate how the bandwidth and the different network nodes interact with each other. TrueTime is a MATLAB[®] toolbox that is used for distributed network simulation. It supports CAN-bus, Ethernet among other network architectures.

Process Data Objects (PDOs) are used to request the data from the motors as well as to receive the different sensor readings. The PDOs are then coupled with a synchronisation message in order to ensure that the motors are moving to a specific joint configuration simultaneously to reach a desired pose without working against each other.

CANopen protocol has been simulated on the CAN-bus network in TrueTime by adding the PDO implementations. Since all of the different stepper motors are connected to the same network, a CAN-bus network node has been created in TrueTime with a 1 Mbps bandwidth.

Although the motors are controlled in joint space, the network simulation is done in task space. The reason for this, is to avoid multiple joint space/task space conversions. Furthermore, the different resolutions have been converted to the task space by adding them to a vector which is multiplied by the Jacobian matrix then taking its norm. This norm represents the resolution

in the task space. This is shown in equation 84.

$$\|P_{res}\| = \|J(\theta) \cdot \theta_{res}\| \quad (84)$$

The TrueTime network has 5 nodes; a controller node representing the Teensy, and 4 nodes representing each of the pose elements x, y, z and φ . The communication starts within the Teensy node which has two different tasks; a periodic task and an interrupt task.

The periodic task is used to send a synchronization PDO requesting the pose from the 4 nodes. This allows saving the current pose of the robot at the time when the actuator nodes have received the PDO and then the nodes send the saved data when the network becomes available. The interrupt task is triggered whenever *any* new message is sent on the network. It processes the data that is received from the 4 nodes and saves their values in a buffer. Only when the data from the 4 nodes has been received, a control signal F_a is calculated and sent to each of the 4 nodes as a PDO. When each new commanded F_a entry has been sent to the corresponding node, then a sync frame is sent to the nodes to initiate the movement. From this a design can be seen in figure 22.

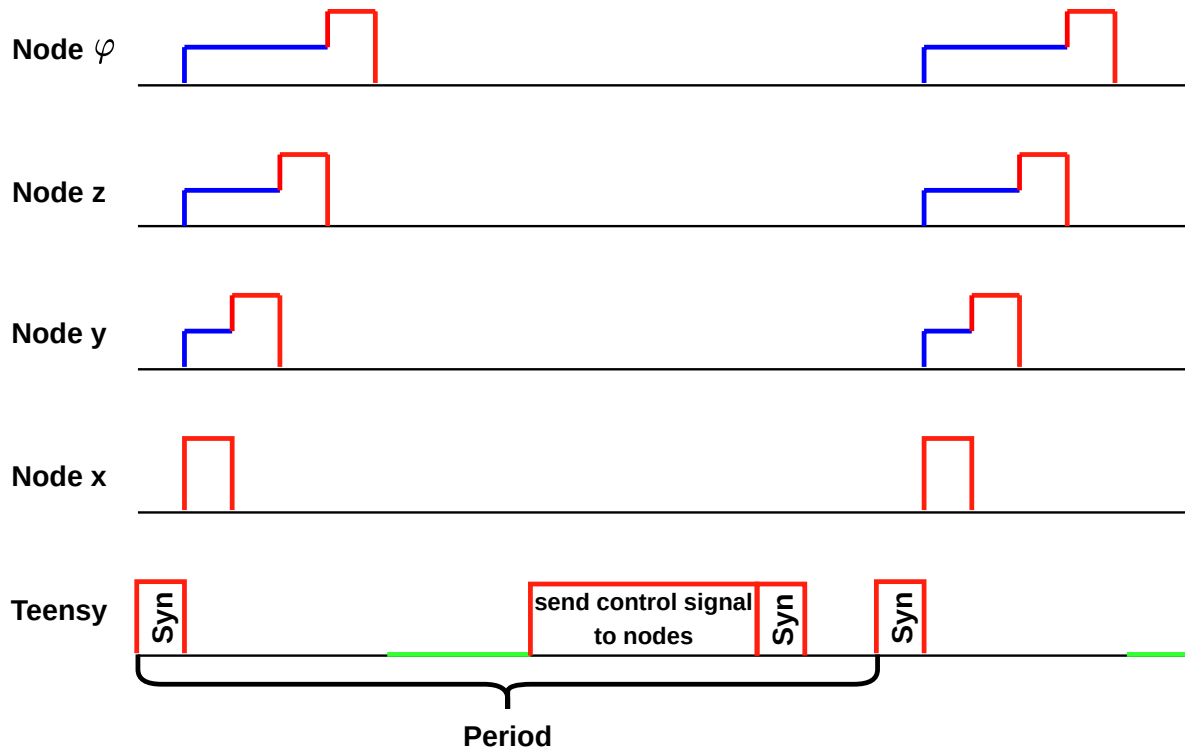


Figure 22: Network scheduling design showing the execution of the Teensy node (green), the sending of messages (red) as well as the queuing of the messages (blue).

As it may be seen in the scheduling above, the sync message has two different behaviours; requesting data and initiating movement. This behaviour is controlled by setting different sync counters when configuring the PDOs. In this case, the sync counter for initiating the movement is 1 whereas the sync counter for requesting data is 2.

Algorithm 1: Pseudocode of the network simulation

```

1 while robot is running do
2   repeat every  $T_s$  ms
3     | send synchronization PDO to request end-effector pose;
4   end
5   if data received from all nodes then
6     | read pose;
7     | calculate desired generalized forces;
8     | for  $i \leftarrow 2$  to 5 do
9       | send force to node  $i$ ;
10    | end
11    | send synchronization PDO;
12  end
13 end
  
```

Figure 23: Pseudocode of the network simulation.

Note that the signal lengths and execution times do not represent reality since this is only used to design how the nodes will communicate on the network. The algorithm showing the implementation can be seen in figure 23.

6.1.1 Computation Timings for Network Simulation

In order to make an accurate simulation, the timings of the network has to be accurate as well. The TrueTime simulator is used to find the transmission times, given the cable length and the number of bits to be sent. The execution time of the synchronisation signal and the control signal is found by measuring the time before and after executing each task. The timings are found using `micros()`. The execution times of the node pack-and-send signal in addition to the receive and execute times are found in the datasheets of the motor [11]. The found timings can be seen in figure 24.

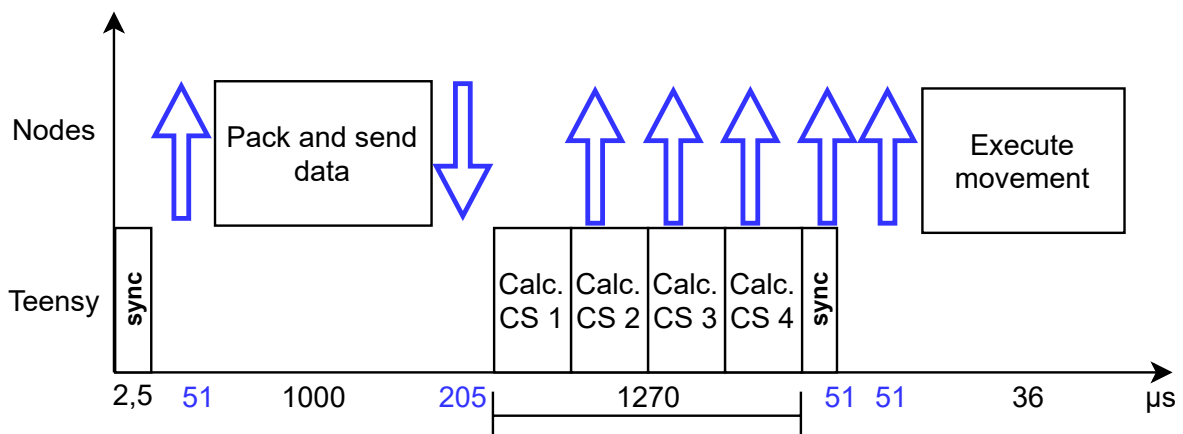


Figure 24: Timings in one control loop.

Looking at figure 24, the black boxes and numbers represent computation time on either the nodes or the Teensy, while the blue arrows and numbers represent transmission times. The time for the motors to save the control signal is found to be $36\ \mu\text{s}$. Since this time is lower than the transmission time of a sync signal, and therefore do not contribute to the total loop time, these are completely omitted on the figure. The total computation time is then found to be $2665\ \mu\text{s}$ which results in a maximum system frequency of 375 Hz.

6.2 Networked Controller - Simulation Results

Using the timings from section 6.1.1 the networked controller simulation can be made. It is chosen to run with a period of 3 ms resulting in an error margin of $3000 - 2666,5 = 0,3335\ \text{ms}$. The result of the simulation can be seen in figure 25.

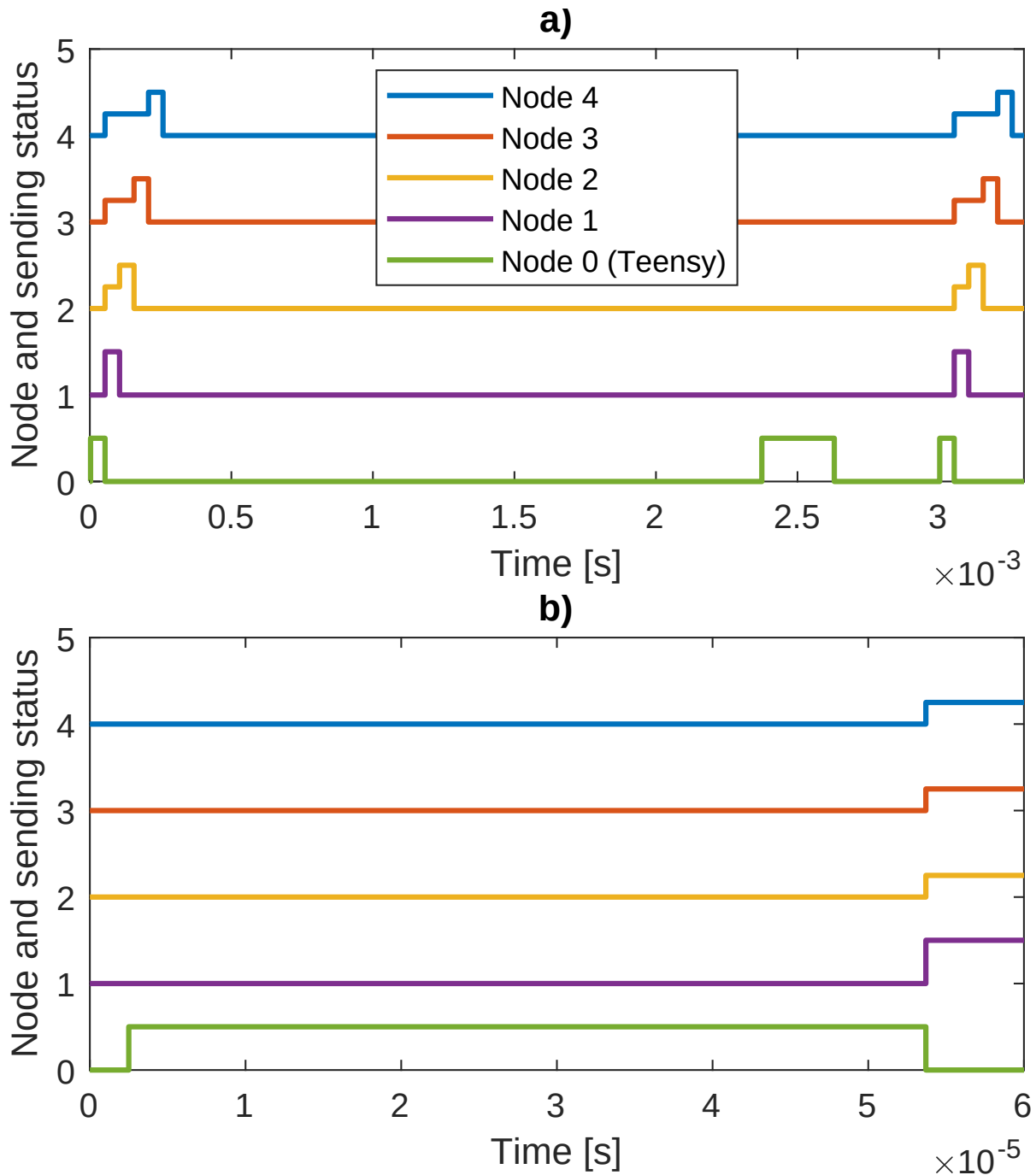


Figure 25: Simulation result of network. Figure b) is a zoomed in version of start of A. Integers on the y-axis indicate node number as well as priority, with 0 being highest. The y-values also represent sending state: $y + 0$ = Node y not sending. $y + 0,5$ = Node y sending. $y + 0,25$ = Node y waiting to send (eg. preempted by higher priority node).

Figure 25a shows the communication cycle. At time 2,5 μs (see 25b) the first sync frame is sent, and at time 2630,5 μs (see 25a) the last sync message has finished sending. This corresponds with the calculated time $2666,5 - 36 = 2630,5 \mu s$. In figure 25a it is also seen that a new cycle is started at 3 ms resulting in a system frequency of 333,33 Hz.

6.3 Sample Rate Selection

To find a proper sample rate for the Teensy, it is chosen to look at the fastest observer pole. From Eq. (74) the fastest pole is:

$$p_{obs} = -70,02 \quad (85)$$

In frequency it gives the following:

$$f_{p_{obs}} = \frac{70,02 \text{ rad/s}}{2\pi} \quad (86)$$

$$\approx 11 \text{ Hz} \quad (87)$$

This means that the minimum sample rate to satisfy the Nyquist sampling theorem is $2 \times f_{p_{obs}} = 22 \text{ Hz}$. With a Teensy processor it is possible to push the sampling rate even further, thus it is decided to go with a rate of 333,33 Hz based off of the timing/calculations done in 6.2.

6.4 Packet Loss

The full-order observer designed in section 5.3 is intended to improve performance in case of packet loss. In this section the improvements gained when losing packets is investigated. The observer allows the system to regulate of estimated states even when measured states are lost during transmission. A procedure for how to handle packet loss is set up for both state feedback only and observer, to allow comparison between the two.

When only having state feedback, in the case of having lost packets, the system will not be able to calculate a controller output. The procedure to be used is to reuse the previous calculated controller output until new measurements arrive.

When using the observer the procedure is chosen to be only relying on the observer, without it correcting it self through the observer gain. This is achieved by making the observer error zero during packet loss.

The biggest impact of packet loss is during transient periods. Therefore three worst case scenario tests are performed where packet loss of all measurements are simulated for a period of seconds, in start, middle and end of a transient period.

It must be noted that the following simulations with packet loss, are extreme cases of packet loss. The simulations show what happens when not only one packet is lost, but what happens when all packets are lost in a period of 0,2 s, occurring at different times.

Beginning of a Transient Period

For the simulations, in figure 26, it is presented what happens in case of packet loss starting just before a transient period and ending during the transient period.

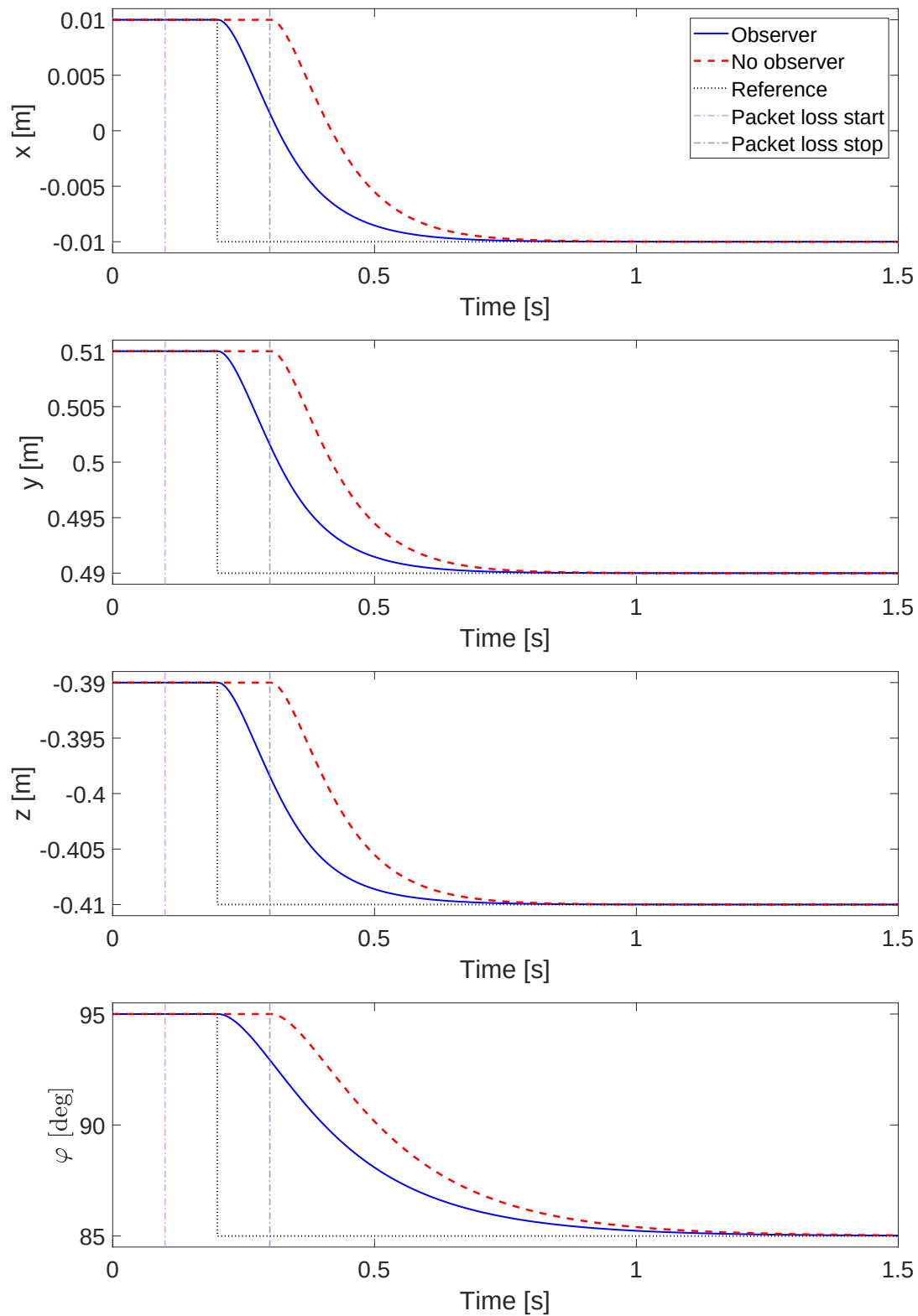


Figure 26: Simulation results showing packet loss consequences with and without the observer. Here, the packet loss occurs just before a transient period is about to start.

It can be seen from figure 26, that the case where no observer is used, see the red dashed

curve, the transient period is delayed. The case where the observer is included, it is seen that the observer takes over and starts the transient period at the right time, namely at 1 s. This is expected as the non-observed system keeps reusing its steady state input until packets arrive.

Middle of a Transient Period

For the simulations, in figure 27, it is presented what happens in case of packet loss in the middle of a transient period.

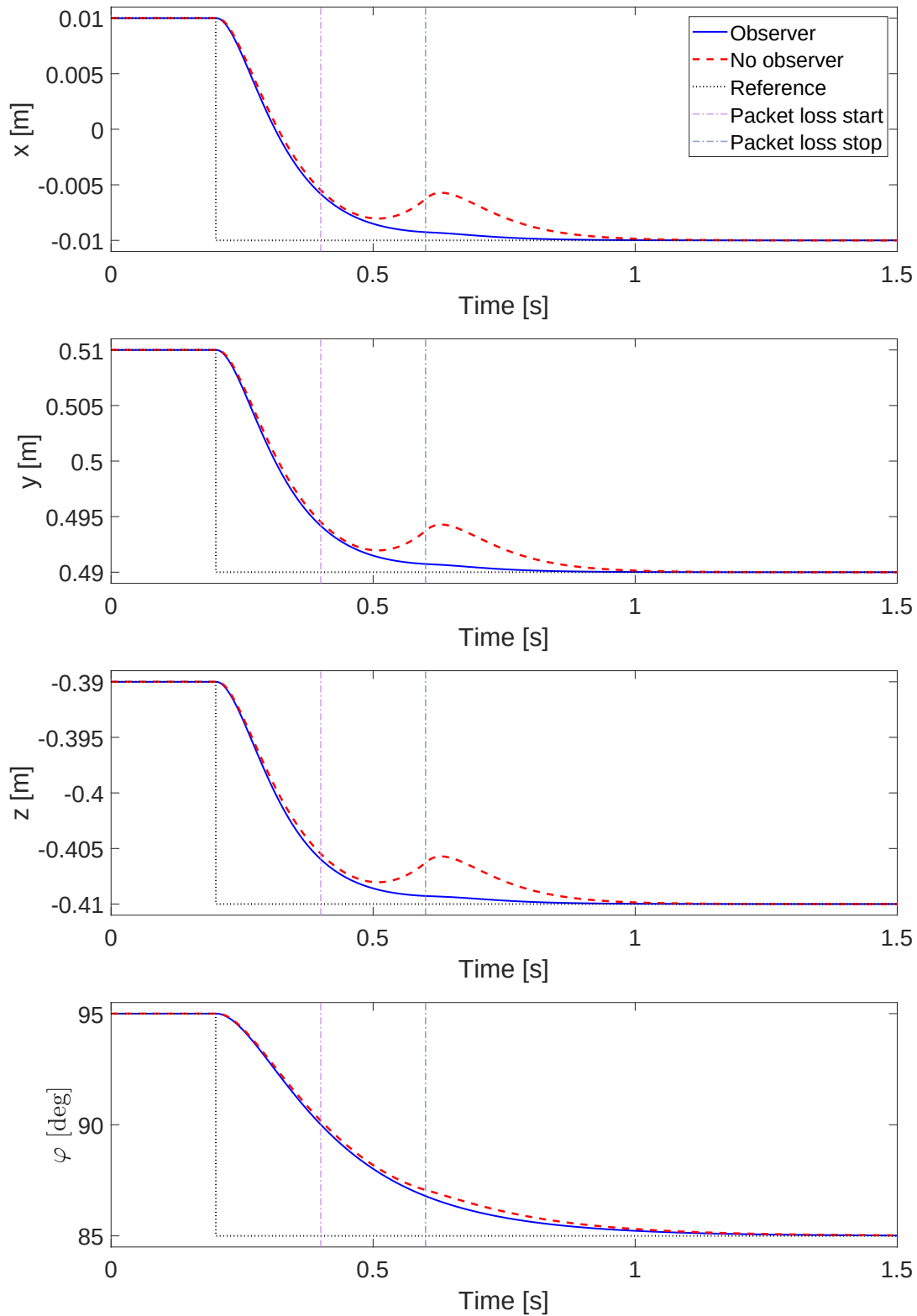


Figure 27: Simulation results showing packet loss consequences with and without the observer. Here, the packet loss occurs in the middle of a transient period.

As observed on figure 27 for x and φ the packet loss does not have a significant impact. This is due to the fact that the control output should not change much in the packet loss period. During

this period x and φ , looking at the observer curve, experiences close to constant deceleration, this is seen from the curvature not changing. Therefor applying a constant torque throughout the period gives an acceptable result.

For y and z a different behaviour is seen. In these states it is observed that the end-effector starts to move in the wrong direction. At the point of packet loss a more substantial torque compared to x and φ is applied to decelerate. Keeping this throughout the period, the system decelerates to a stop and start moving the wrong way. This behaviour is also observed on the velocities, seen in the forthcoming figure 28.

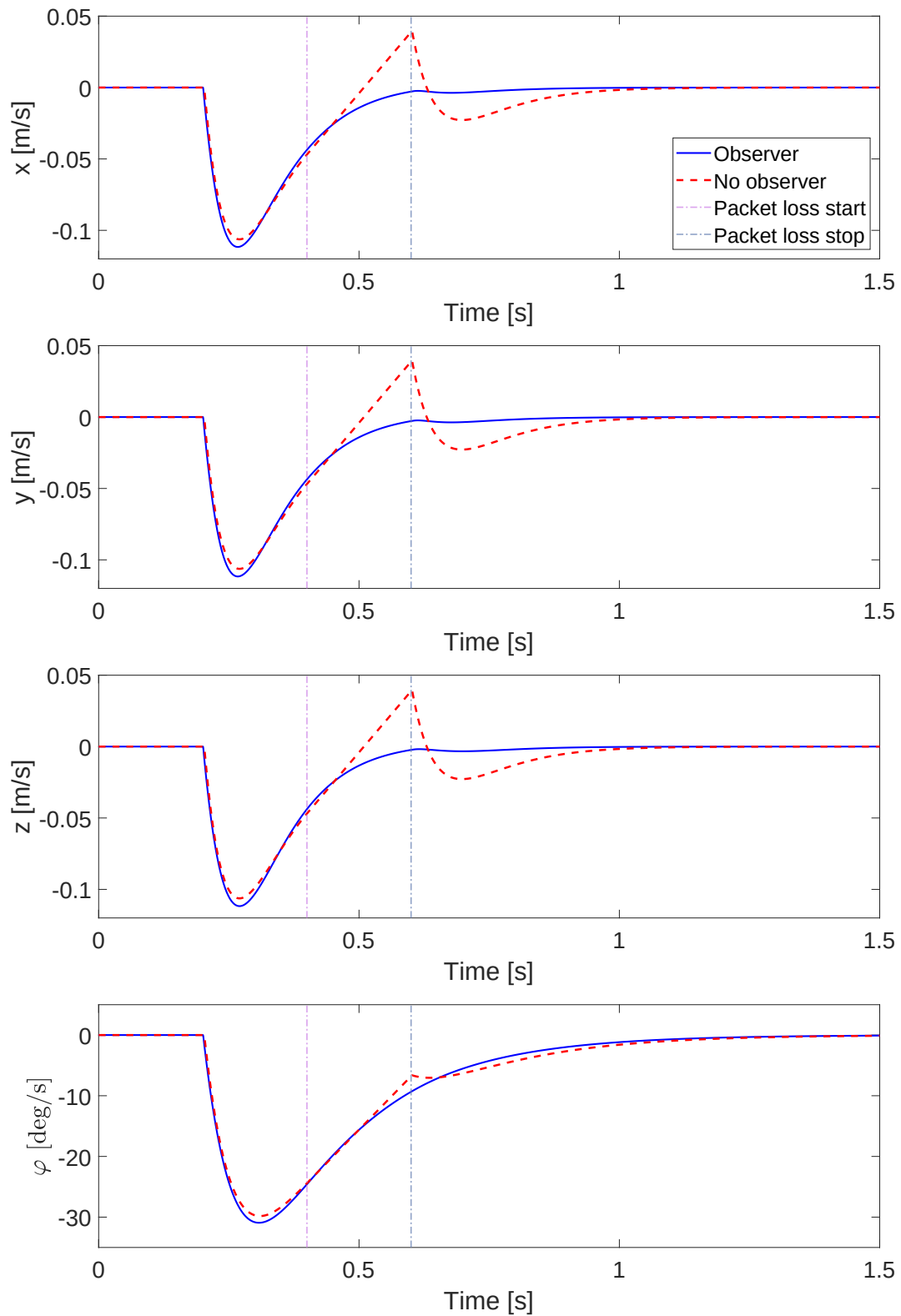


Figure 28: Simulation results showing packet loss consequences, in respect to velocities, with and without the observer. Here, the packet loss occurs in the middle of a transient period.

From figure 28 the consequence of applying the same torque input during packet loss periods

is directly seen. As can be observed, throughout the period accelerations are constant, hereby making velocity changes linear, that is without the observer. It can be seen that this gives only small differences for the for x and φ , as the accelerations is close to constant throughout the period. For y and z however, the accelerations significantly decrease as the end-effector nears its desired pose, not doing so makes the non-observed systems velocities cross zero, and hereby move in the wrong direction.

End of a Transient Period

For the simulations, in figure 29, it is presented what happens in case of packet loss starting just as a transient period is about to end and ending at a point where the period should be concluded.

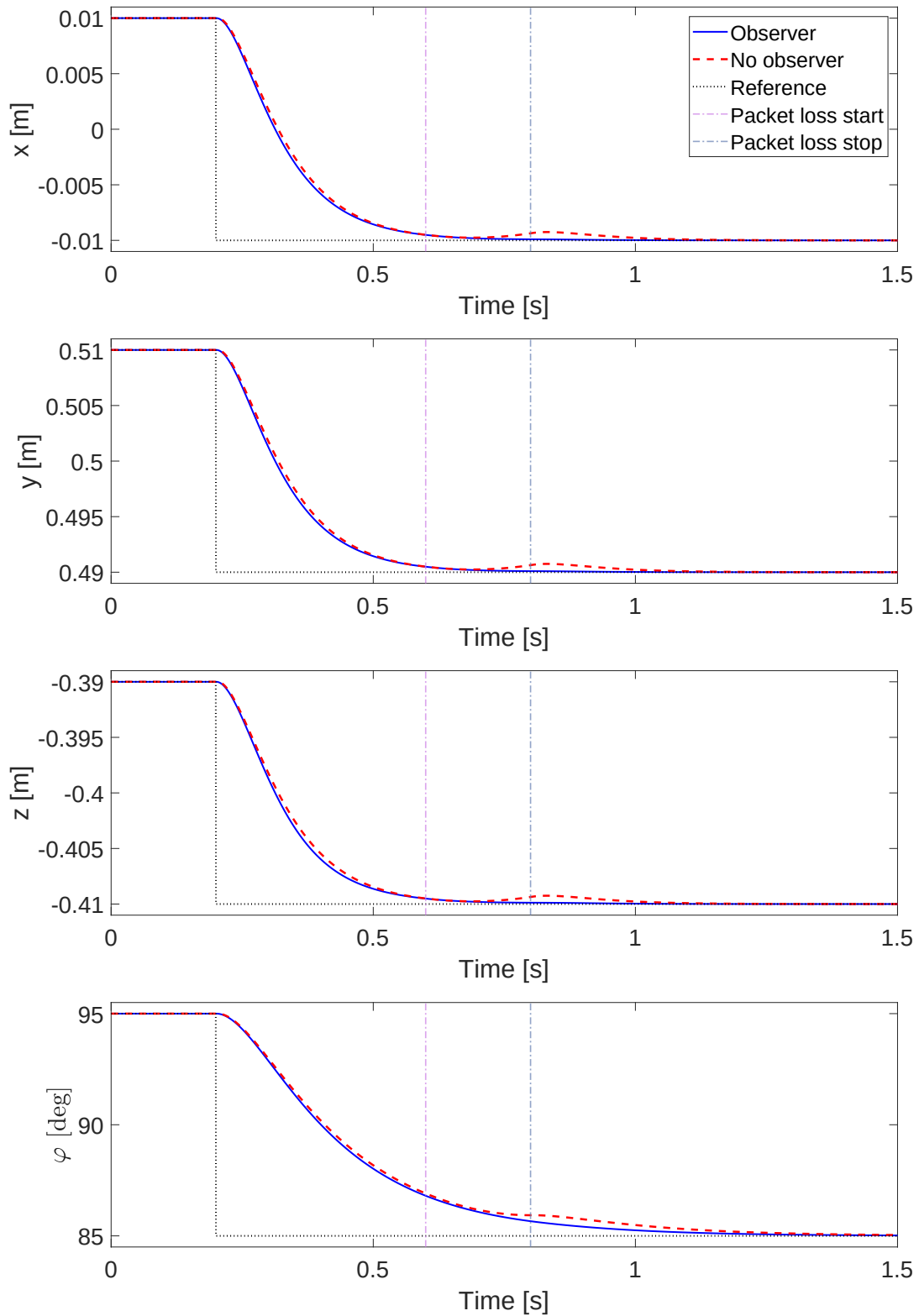


Figure 29: Simulation results showing packet loss consequences with and without the observer. Here, the packet loss occurs just as a transient period is about to end.

From figure 29, a similar phenomenon is seen, as seen from figure 27, for the y and z state without the observer. Again, this is due to the fact that the packet loss is happening at a point

where the velocity is decreasing. The packet loss occurs and therefore the velocity decrease is maintained leading to the little notch, for all states in the non-observed case.

7 Results

In this chapter the final system will be presented, as well as the simulation results. The final system is a combination of the controller design in chapter 5 and the network simulation in chapter 6. The networked version of the full-order observer controller implemented with the TrueTime toolbox, can be seen in figure 30.

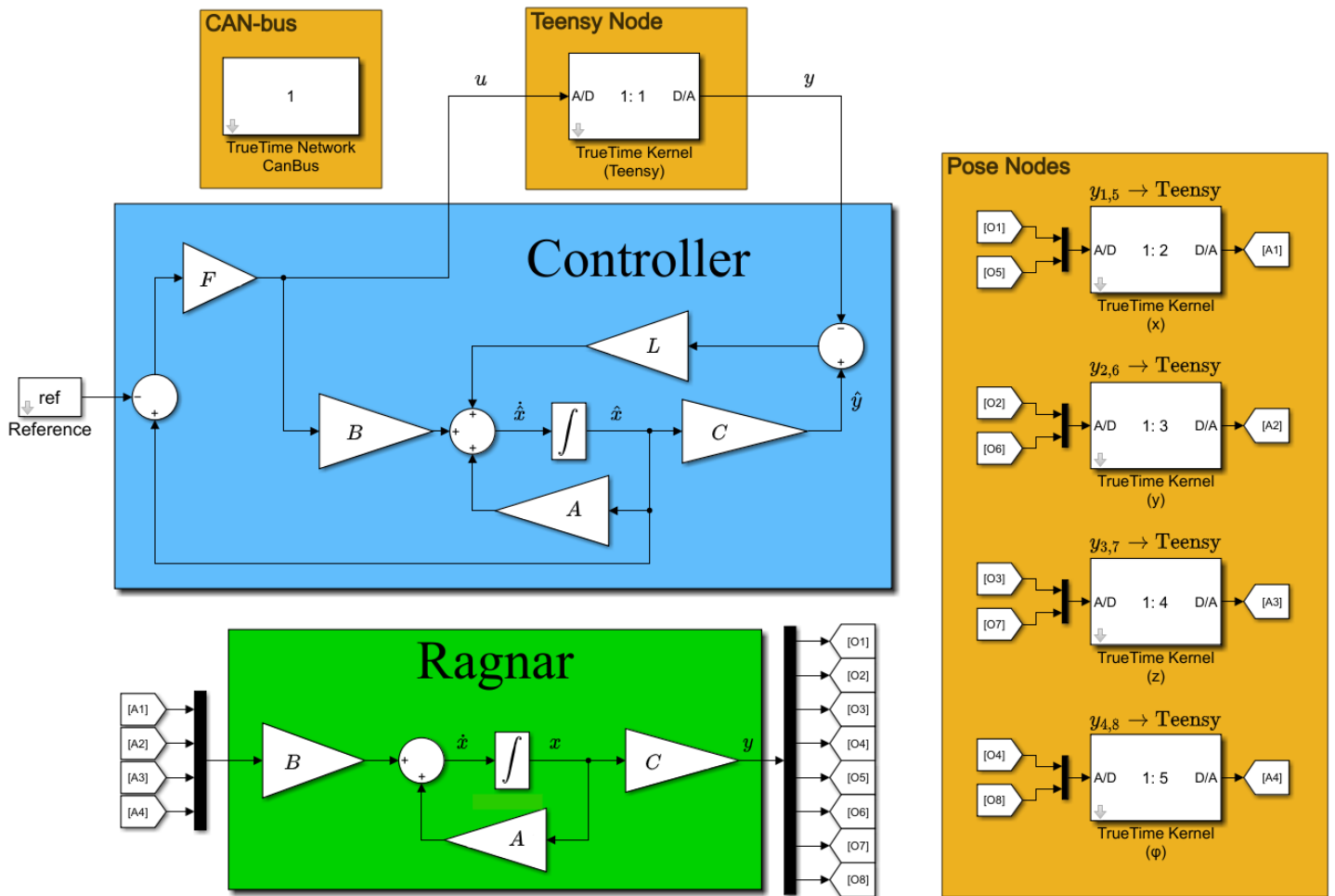


Figure 30: CAN-bus network topology used for TrueTime simulations.

It can be seen that figure 30 is identical to figure 19, except that the connections between the controller and Ragnar is now done over a network. The resulting pose of the system given the reference seen in table 3 can be seen in figure 31. The simulation will be compared to the non-networked simulation.

Time [s]	-0	+0	1	2	3	4	5
x [m]	0	0,01	0,01	0,01	0,01	-0,01	0
y [m]	0,5	0,5	0,51	0,51	0,51	0,49	0,5
z [m]	-0,4	-0,4	-0,4	-0,39	-0,39	-0,41	-0,4
φ [°]	90	90	90	90	90,5	89,5	90

Table 3: Poses used as reference in the simulation test.

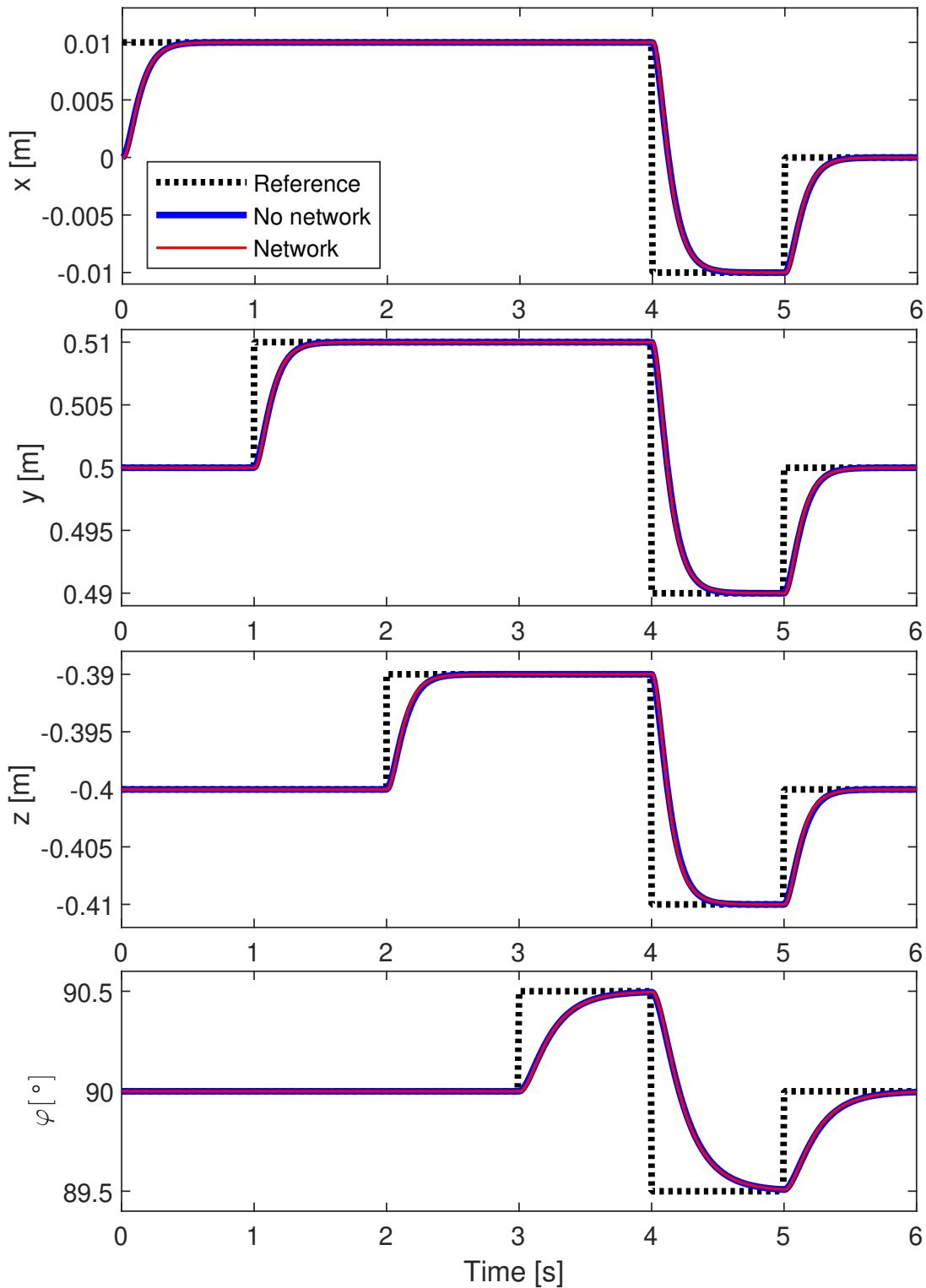


Figure 31: Simulation result of the networked observer and non-networked observer controllers.

Looking at figure 31 it is seen that the networked simulation follows the non-networked simulation nicely, indicating that the network only has a minimal impact. As described in chapter 6 the task space resolution of the stepper motors are also simulated, giving a square movement pattern. A comparison plot between the networked system, with and without resolution accounted

for can be seen on figure 32.

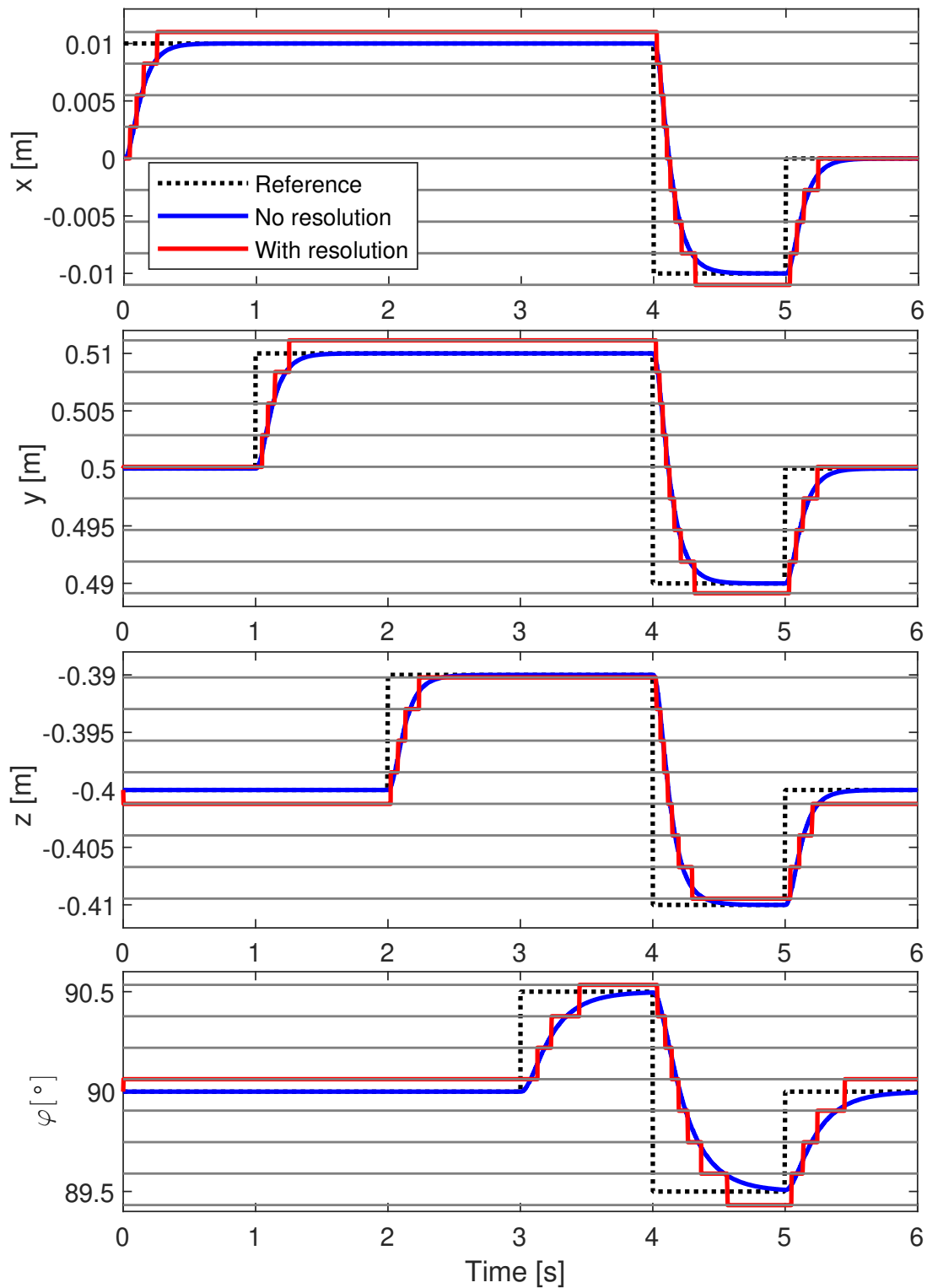


Figure 32: Simulation result of the networked observer controller with and without resolution. The grey lines represent the resolution of the nodes.

Looking at the networked resolution simulation, figure 32, it can be seen that the result is presented in a discrete manner and that the signal seem to have small steady-state errors. This

is due to the resolution of the motors. Since the motors are stepper motors, they can only exist stationary in certain positions which are 1024 evenly distributed in one full axis rotation. Each of the grey lines in figure 32 represents such a position. This means that when a measurement of the current motor axis position is requested by the Teensy, only values on these grey lines can be returned. Therefore, it is seen that if a reference is requested which is not exactly on one of these lines, the motors will instead converge to the nearest feasible position.

8 Conclusions

For all the simulations, the results that are obtained are very good regarding the performance of the observer. In a real world test, it would most likely be less so. The niceness of the results are due to everything running in Simulink®, thus, the model of Ragnar used for the observer is an exact copy of the model acting as Ragnar in the simulation without the observer. Normally the model used in an observer is exactly that, a model, which will lead to some inaccuracies when the observer takes over, which is why it is corrected during normal service, where measurements are available. The results look as expected for a simulation, and therefore are concluded to be satisfactory.

References

- [1] Trefethen, L. N. & Bau, D. (1997). Numerical Linear Algebra. SIAM
- [2] Food and agriculture organization of the united nations, Gateway to poultry production and products, accessed: 17/11/2021, Available: <https://www.fao.org/poultry-production-products/products-processing/en/>
- [3] Teknologisk Institut, Robot kan erstatte løftearbejdet, accessed: 17/11/2021, Available: <https://www.teknologisk.dk/robot-kan-erstatte-loeftearbejdet/39605>
- [4] J. Bjerregaard, J. Reynskor, M.L. Winther, N Birwe, V. Kuldaite, "Compliant Networked Control of Parallel 3T1R Robot", 2020
- [5] Solve Minimax Constraint Problem, accessed: 14/12/2021, Available: <https://se.mathworks.com/help/optim/ug/fminimax.html>
- [6] J. Flores-Mendez, H. Schiøler, O. Madsen, S. Bai, "Impedance Control of a Redundant Parallel Kinematic Manipulator", 2017, In Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO, ISBN 978-989-758-263-9; ISSN 2184-2809, pages 104-111. DOI: 10.5220/0006433301040111
- [7] Gene F. Franklin, David J. Powell, and Abbas Emami-Naeini. 2014. Feedback Control of Dynamic Systems (7th. ed.). Pearson.
- [8] John J. Craig. Introduction to Robotics. 3rd ed. Pearson Educational International, 2005.
- [9] CiA. CANopen. accessed: 14/12/2021. Available: <https://www.can-cia.org/canopen/>
- [10] Cyberbotics Ltd, Webots <https://cyberbotics.com/>
- [11] Nanotec, PD4-C Technical manual. accessed: 15/10/2021. Available: https://en.nanotec.com/fileadmin/files/Handbuecher/Plug_Drive/PD4-C/PD4C_CANopen_Technical-Manual_V2.0.2.pdf
- [12] MathWorks, Documentation for place. accessed: 13/12/2021. Available: <https://se.mathworks.com/help/control/ref/place.html>
- [13] Lnorm, L^2 -Norm, accessed: 30/10/2021. Available: <https://mathworld.wolfram.com/L2-Norm.html>
- [14] Mathrowks, Documentation for fmincon, accessed: 25/10/2021. Available: <https://se.mathworks.com/help/optim/ug/choosing-the-algorithm.html#bsbwxm7>
- [15] Stephen Boyd, Lieven Vandenberghe, Convex Optimization, Cambridge University Press 2004.
- [16] Mathworks, Optimization Toolbox, accessed: 25/10/2021. Available: <https://se.mathworks.com/products/optimization.html>

Appendix A: Joint Space Transformation

The system model obtained in Chapter 4 takes the wrench $\underline{F}_a$ as input and give the states as output. Bearing in mind that the model is in task space. Ragnar is however in configuration/joint space, with it having torques as input to each of the four joints, and a force input to the two linear actuators. The actual output of the robot also consist of four angles and angular velocity measurements plus an additional two lengths and velocities for the linear actuators.

This means that in order to model the robot with the derived task space model. A conversion between task space and configuration space is needed.

Outputs from the model will be converted from task space to configuration space by the use of the Jacobian defined in section 2.1 Eq. 21. As the transformation only works for velocities and not poses, the operating pose is utilized. This can be seen from equation 88, where the bar indicates operating pose value.

$$\underline{\theta} = \underline{J}^{-1}(\underline{P} - \bar{\underline{P}}) + \bar{\underline{\theta}} \quad (88)$$

Applying this the use of forward kinematics is avoided, however, inverse kinematics will handle finding the operating point value in configuration space.

The input to the model is as mentioned the wrench $\underline{F}_a$. The robot input in configuration space is $\underline{\tau}_\theta = [\tau_{\theta_1}, \tau_{\theta_2}, \tau_{\theta_3}, \tau_{\theta_4}, F_{\theta_5}, F_{\theta_6}]^T$. An expression for converting between the two is derived from the power formula. As the power is constant when changing between spaces.

$$\begin{aligned} \text{Power} &= \langle \underline{\tau}_\theta, \dot{\underline{\theta}} \rangle = \langle \underline{F}_a, \dot{\underline{P}} \rangle \\ &= \underline{\tau}_\theta \cdot \dot{\underline{\theta}}^T = \dot{\underline{P}}^T \cdot \underline{F}_a = (\underline{J} \cdot \dot{\underline{\theta}})^T \cdot \underline{F}_a \\ \updownarrow \underline{\tau}_\theta &= \underline{J}^T \cdot \underline{F}_a \\ \updownarrow \underline{F}_a &= \underline{J}^{-T} \cdot \underline{\tau}_\theta \end{aligned} \quad (89)$$

To achieve a model of Ragnar in configuration space, the above conversions need to be added. Moreover, as the designed controller is in task space, the output and input also needs converting. All conversions can be seen in the block diagram below.

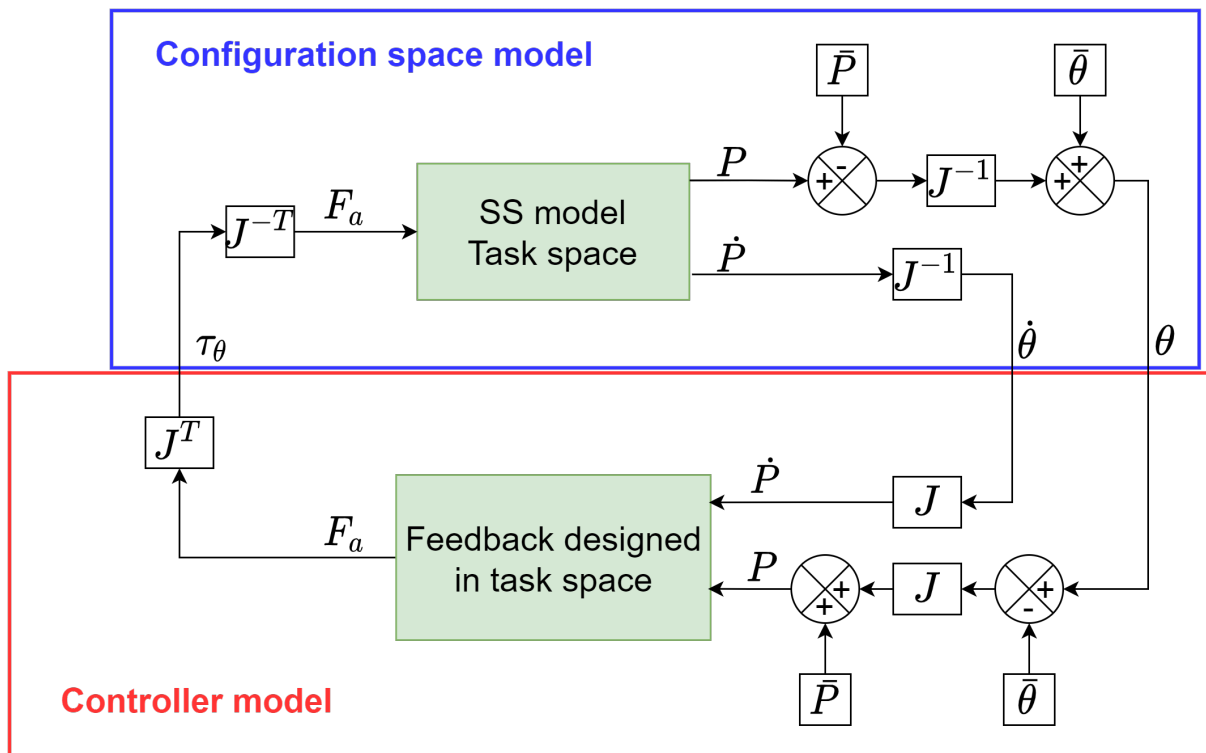


Figure 33: Block diagram showing conversion between configuration space and task space.

As illustrated on figure 33 the conversions are first made one way then the other. In reality this will not be the case, as Ragnar by itself takes inputs and gives outputs in configuration space. Consequently when implemented, only the conversions in the red box labelled "Controller model" will be needed.