NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

APPLICATION OF RECURRENT NETWORKS FOR SPACECRAFT ATTITUDE CONTROL

by

Benjamin Diehl

December 2020

Thesis Advisor: Mark Karpenko
Co-Advisor: Isaac M. Ross

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2020	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE APPLICATION OF RECURRENT NETWORKS FOR SPACECRAFT ATTITUDE CONTROL	**5. FUNDING NUMBERS**
6. AUTHOR(S) Benjamin Diehl	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

There is a need to perform a control allocation to map the torque commands from the body frame to the actuator frames in a spacecraft attitude control problem. When the number is more than three, this mapping, done by a pseudoinverse, is not unique and depends on the cost function used to perform the allocation. The infinity norm allocation is a specific control allocation method that emphasizes minimizing the maximum torque actualized by the spacecraft's momentum exchange devices. It also allows access to the full torque envelope, giving approximately an extra 20 percent of torque capacity for the system to utilize over the traditional calculation method. The infinity norm allocation is not generally used as it requires an iterative optimization to be performed. An alternative is to use a recurrent neural network to solve the allocation problem. Implementation of a recurrent neural network can solve the pseudoinverse of the torque transformation matrix of a spacecraft's momentum exchange devices. The recurrent neural network is shown to improve performance over the conventional allocation scheme for both reaction wheel and control moment gyro actuated spacecraft.

14. SUBJECT TERMS pseudoinverse, momentum exchange transformation, agility envelope, path optimization, infinity norm	**15. NUMBER OF PAGES** 93
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

APPLICATION OF RECURRENT NETWORKS FOR SPACECRAFT ATTITUDE CONTROL

Benjamin Diehl
Lieutenant, United States Navy
BS, University of California – Davis, Mechanical Engineering, 2012
BS, University of California – Davis, Aerospace Science and Engineering, 2012

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ASTRONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2020

Approved by: Mark Karpenko
Advisor

Isaac M. Ross
Co-Advisor

Garth V. Hobson
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

There is a need to perform a control allocation to map the torque commands from the body frame to the actuator frames in a spacecraft attitude control problem. When the number is more than three, this mapping, done by a pseudoinverse, is not unique and depends on the cost function used to perform the allocation. The infinity norm allocation is a specific control allocation method that emphasizes minimizing the maximum torque actualized by the spacecraft's momentum exchange devices. It also allows access to the full torque envelope, giving approximately an extra 20 percent of torque capacity for the system to utilize over the traditional calculation method. The infinity norm allocation is not generally used as it requires an iterative optimization to be performed. An alternative is to use a recurrent neural network to solve the allocation problem. Implementation of a recurrent neural network can solve the pseudoinverse of the torque transformation matrix of a spacecraft's momentum exchange devices. The recurrent neural network is shown to improve performance over the conventional allocation scheme for both reaction wheel and control moment gyro actuated spacecraft.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

I would like to thank Dr. Mark Karpenko, who helped to guide my process throughout the thesis and whose assistance as my advisor made this thesis possible.

Additionally, I would like to thank Dr. Isaac Ross, who helped get me over the finish line.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Typical methods of implementing spacecraft attitude control do not give the spacecraft access to the entire torque envelope available to the control devices onboard. This thesis will focus on the development of an infinity norm control allocation method that will allow the spacecraft to utilize the entire torque envelope available to it [1]. Additionally, it will utilize a recurrent neural network to perform the control allocation and prove the feasibility of its implementation within a spacecraft control loop and demonstrate the advantages of the control allocation scheme.

When a spacecraft is reoriented from point A to point B, the maneuver is usually done through two different processes [2]. The guidance process tells the spacecraft the path it needs to take to reach the objective. The control process describes how the vehicle's actuators need to be commanded to achieve the given path. Where these processes reside in the spacecraft architecture can be visualized in the control loop depicted in Figure 1.1, which employs a four reaction wheel system [3]. The reorientation begins when the ground station sends a command to the spacecraft. The onboard flight computer will then supply the guidance profile required in three-dimensional space to achieve the maneuver. The maneuver trajectory will be fed through a controller, e.g., Proportional-Derivative (PD) or Proportional-Integral-Derivative (PID), to generate the are body frame torque profiles for executing the maneuver. A pseudoinverse is then used to map a three-dimensional torque to four independent reaction wheel actuators.

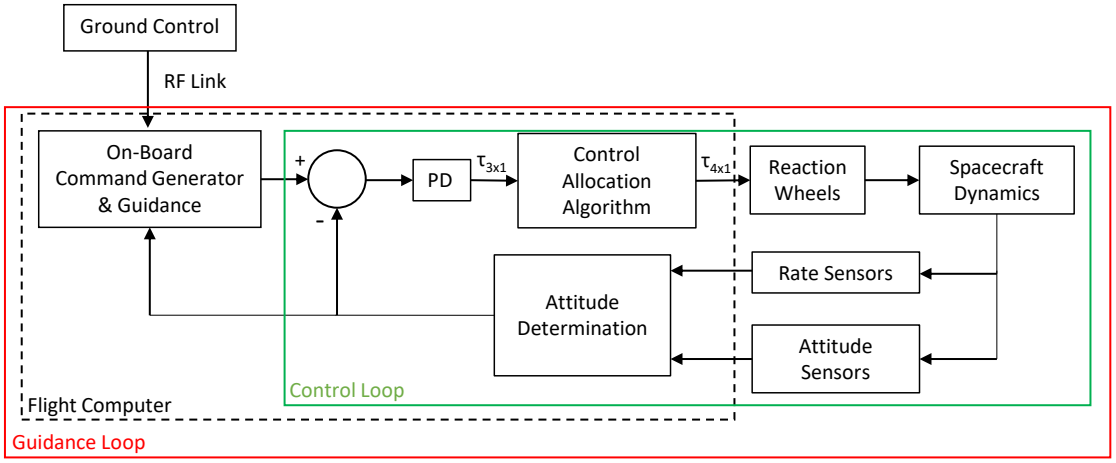


Figure 1.1. Spacecraft Guidance and Control Architecture. Adapted from [3].

Allocation of spacecraft body torques to the given control actuators are determined by the cost function used to calculate the pseudoinverse [2]. The least squares allocation minimizes the maximum of the sum of the squares of the individual torques applied. Additionally, the least squares allocation method can be solved analytically. However, the use of the pseudoinverse with the infinity norm, which minimizes the maximum torque commanded, will result in a better maneuvering space [4] and potentially improve control performance or allow the selection of smaller attitude control systems. The control costs for the least square and the infinity norm can be seen in equations (1.1) and (1.2), respectively.

$$L_2 = \min \sum \tau_i^2 \tag{1.1}$$

$$L_\infty = \min \max \tau_i \tag{1.2}$$

where L_2 and L_∞ are the least squares and infinity norm output, τ is a vector of values undergoing the allocation method, and i is the length of the vector.

Solving for the infinity norm requires the numerical solution of an optimization problem [5]. The implementation of the infinity norm allocation can be realized using an iterative linear

programming approach. This method can be resource-intensive for flight computers installed onboard spacecraft. Another approach allows for the allocation to be obtained from a recurrent neural network. A recurrent neural network differentiates itself from a standard feedforward neural network by feeding information back into itself as it converges to a solution [6]. Many different recurrent neural networks have been developed in robotics for redundant link manipulators, such as the neural network developed by Tang and Wang [7]. Dual networks have also been developed to incorporate the robotic manipulator's constraints, such as joint positions and velocities [8]. In theory, these recurrent neural networks can be adapted for use in spacecraft attitude control.

1.1 Artificial Neural Networks

A neural network is modeled loosely on the human brain in that it comprises a collection of nodes [6]. Furthermore, a node receives input from the nodes attached before it, and when the right circumstances are met, it activates and sends a signal to the nodes ahead of it. The node described is part of a neural network layer, and each layer can consist of several nodes. A neural network can consist of multiple layers of nodes. Nodes from the prior layer will connect to every node on the layer after, continuing until the last layer. The final layer of nodes will deliver the output to the user. Two types of neural networks architecture are discussed in this section: the feedforward and the recurrent networks. The recurrent network architecture is utilized in this thesis but feedfoward will be discussed first to introduce machine learning concepts.

A typical neural network is a feedforward model, meaning that the flow of information only moves forward. Feedforward neural networks are well suited for applications, such as identifying objects in images [6]. A typical feedforward model can be seen in Figure 1.2 that has an input layer, two hidden layers, and an output layer. The user will feed the input into the system and thus begins the forward process through the network. The nodes of a layer will have an activation function associated with sending information to the layer ahead. In Figure 1.2, the Rectified Linear Unit (reLU) and hyperbolic tangent (tanh) activation functions are shown. The reLU function only activates above a certain threshold and outputs a signal equal to what activated the node. The tanh activation has a hard upper and lower limit and only modify the input signal to an output that falls within the tanh bounds. The particular choice of reLU, tanh, or other activation functions is up to the designer of the network and should

be tied to the specific application that is under consideration. Finally, the combination of these outputs is sent to the output layer, which produces the user's result. The number of outputs can

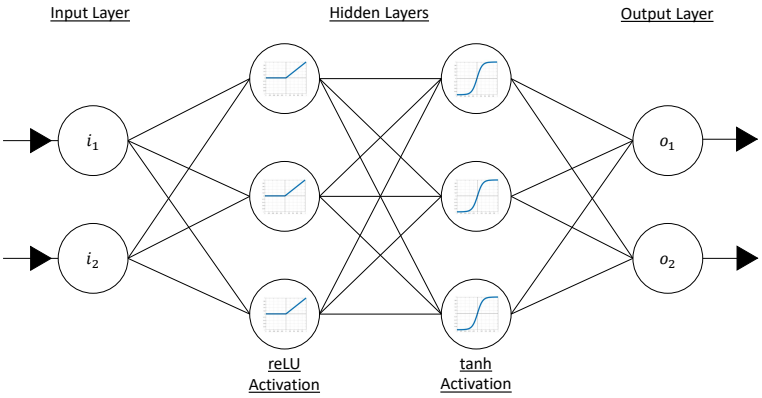


Figure 1.2. Generalized Feedforward Neural Network.

A recurrent neural network differs slightly from the feedforward neural network in that it can feed information from layers ahead of the current layer back into itself, similar to the closed-loop feedback system seen in Figure 1.1. This thesis will focus on the particular recurrent neural network shown in Figure 1.3, which feeds the output layer results back into the input layer. However, this is not the only type of recurrent connection that can be made. For example, the hidden layer can feed the output of one or more nodes back into the layer or layers be

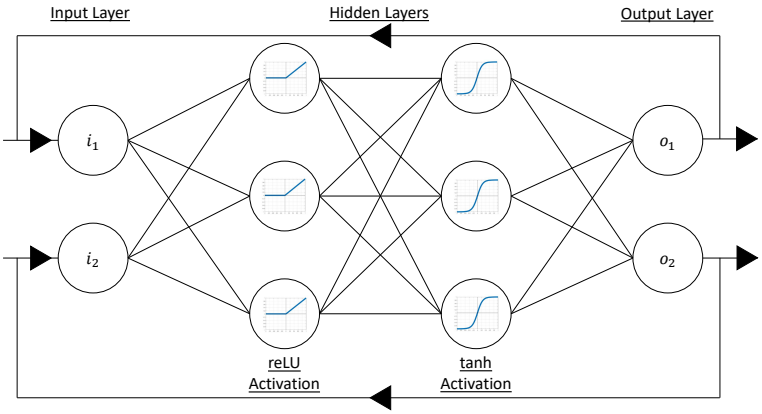


Figure 1.3. Generalized Recurrent Neural Network.

Another distinction to make is how the neural network is trained, it can either be supervised or unsupervised. Supervised training does has a set of training data that the neural network uses to produce correct results for inputs the network has not seen before. Each connection in Figure 1.2 has a weighted value associated with it which can either magnify or weaken the signal being sent forward. This weighted value changes during the supervised training of the artificial neural network to ensure the output layer produces correct results. From the previous example of image identification, the input would consist of a picture with a cat. The input would run through the network, and the networks' weights are then adjusted by the training algorithm until the output layer can identify a cat in the picture. The training is then repeated for multiple images that may or may not have cats in them so that the network weights can be adjusted to give a useful generalization of its output. Unsupervised training differs from its counterpart in that there is no training data for it to correct to and therefore can only be used to find patterns in the data fed into and adjusting the connection weights appropriately.

This thesis will use a recurrent neural network that falls into a class of unsupervised training. Instead of just finding patterns in the data fed into it, it will utilize an energy function to derive the mathematics applied to the output of the neural network that will be fed back into the system. The energy function essentially guides the output of the system to converge to a result.

1.2 Thesis Objectives and Scope

Spacecraft attitude control typically lies within the feedback control loop. This control loop is typically defined by taking a three-dimensional torque and rewriting it to the dimensions defined by the control axes of the spacecraft control allocation system. More often than not, this system does not align itself with a three-dimensional Cartesian coordinate system. When trying to go from a lower dimensionality to a higher dimensionality of control instruments, it will result in a non-unique solution, as governed by the control allocation cost, as seen by either equation (1.1) or (1.2). This thesis will investigate the traditional control allocation method of the least square, the Moore-Penrose inverse, and how it compares to the infinity norm cost allocation method in order to demonstrate the advantages of the infinity norm.

Chapter 2 will examine the infinity norm and its application within the robotics field. The

results from a paper by Tang and Wang [7] will be recreated as verify of their results and to illustrate how recurrent neural networks can be applied to a control allocation problem. A Design of Experiments (DoE) is proposed in this thesis as a method to select gains and simulation times to accelerate the control allocation process while minimizing error. This technique will be used throughout the rest of the thesis in the selection of design variables. Lastly, the robotic manipulator will be placed into a singular configuration as it concerns both spacecraft control allocation and robotics. Observations of singularity will begin to define some significant advantages in the results between the Moore-Penrose inverse and the infinity norm.

In chapter 3, the application of control allocation in spacecraft attitude control is explored by showing the benefits expected to result from using the infinity norm allocation method and the development of the mathematics to computation the infinity norm. The control law will then be verified via an implementation in a simulated satellite imaging system and a comparison of how the spacecraft performs under the two different allocation methods will be performed.

Additionally, in Chapter 3 the concept of placing restrictions on the pseudoinverse output will be examined, as introduced in the paper by Zhang, Wang, and Xu [8] introduces. This method can also be applied to the control allocation method for a Control Moment Gyro (CMG) controlled spacecraft. Ultimately, Chapter 3 will show how the implementation of infinity norm allocation via recurrent neural network will have inherent stability to its outputs, while the Moore-Penrose inverse will have jitter.

Chapter 4 will develop an optimized control trajectory for the minimum-time reorientation that uses the reaction wheels directly and from a least squares pseudoinverse. It will show that if the controls developed for a maneuver do not consider the control loop, the pseudoinverse will fall short of actualization of the desired movement path. The infinity norm pseudoinverse and its larger torque envelope can still meet a typical feedback control system's demand even without considering how the feedback control is generated.

In order to realize the benefits of the infinity norm pseudoinverse, without incurring the computational stresses that come with the allocation method, the recurrent neural network can be employed as an electrical circuit. The electrical circuit will consist of opamps, resistors, and capacitors that can be configured as adders, subtracters, multipliers, and

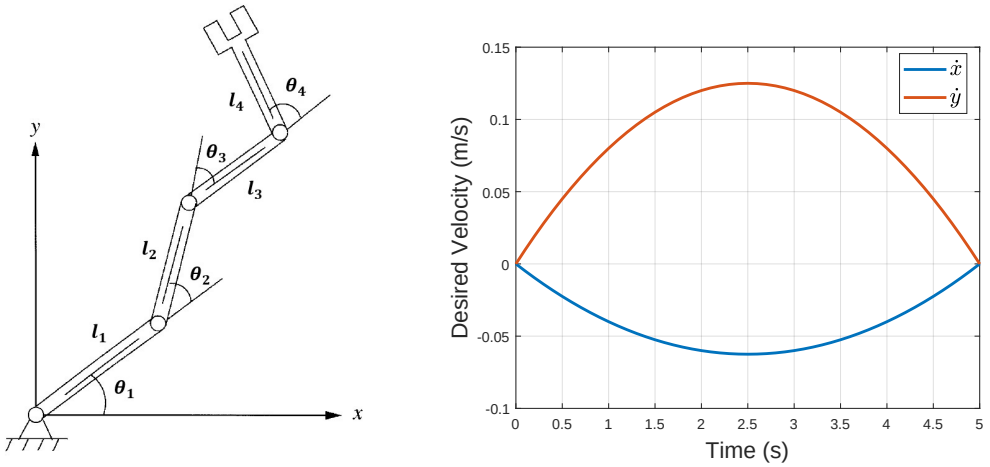
integrators to realize the dynamics of the equations implemented. While the process is still under development for this thesis, it is intended to prove that it is feasible to accomplish.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Infinity Norm Allocation in Robotics Using a Recurrent Neural Network

The field of robotics has often dealt with the pseudoinverse, different methods of solving for the pseudoinverse, and even the use of recurrent networks in the solution [7]–[16]. The pseuodoinverse is used when robotic manipulators have more degrees of freedom (e.g., more joints) than the environment it operates in. In this chapter, the maneuver from [7] is recreated to understand how a recurrent neural network can be used to generate an infinity norm pseudoinverse allocation, develop techniques for improving recurrent network output, and show the recurrent neural network handles singular configurations better than the infinity norm allocation.

Figure 2.1a shows an example of the four-link robotic manipulator used in the maneuver. In the original paper, [7], the joint space variable θ is used to describe the orientation of each link but will be replaced by q in this chapter, and l refers to the link length. Each link's orientation is referenced to the orientation of the one before it. Figure 2.1b depicts the desired velocity profiles of the end effector in both the x and y direction for the planar maneuver.



(a) (b)

Figure 2.1. Manipulator and Velocity Profile. Source: [7]. (a) Manipulator Joint Space; (b) End Effector Velocity Profile.

When generating torque command for the joint motors, the path taken by each joint must be known. In an unconstrained environment, each joint's motion is not a concern, but the desired path end effector, which is the tip of the last link, is what must be controlled. With a redundant link manipulator the mapping of the end effector motion to the individual links becomes a function of the solution to an allocation problem that typically utilizes a least squares pseudoinverse. However, an infinity norm allocation can be utilized instead to provide a minimization of joint velocities induced.

2.1 The Method

The method of path planning that this chapter examines is the first order inverse differential kinematic solution of the end effector velocity profiles shown in Figure 2.1b. For a robotic manipulator, the general joint space solution can be mapped from the end effector velocities using equation (2.1) [17].

$$\frac{d\mathbf{q}}{dt} = J^{-1}\mathbf{v}(t) \tag{2.1}$$

where J is the Jacobian that maps the individual joint velocities to the operational space, and $\mathbf{v}(t)$ is the velocity of the end effector in Cartesian space in time.

The Jacobian is further defined using equations (2.2) and (2.3). Equation (2.2) describes the end effector position in the operational space (which utilizes cartesian coordinates of x and y) using joint space variables, q. Equation (2.2) is then used to perform the calculations required for equation (2.3). The motion planned by these equations is for the joint configurations and does not take into account velocities induced on the joints.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_1\cos(q_1) + l_2\cos(q_1+q_2) + l_3\cos(q_1+q_2+q_3) + l_4\cos(q_1+q_2+q_3+q_4) \\ l_1\sin(q_1) + l_2\sin(q_1+q_2) + l_3\sin(q_1+q_2+q_3) + l_4\sin(q_1+q_2+q_3+q_4) \end{bmatrix} \tag{2.2}$$

$$\begin{aligned} J &= \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \frac{\partial x}{\partial q_3} & \frac{\partial x}{\partial q_4} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \frac{\partial y}{\partial q_3} & \frac{\partial y}{\partial q_4} \end{bmatrix} \\ &= \begin{bmatrix} -l_1\sin(q_1) - \ldots - l_4\sin(q_1+q_2+q_3+q_4) & \ldots & -l_4\sin(q_1+q_2+q_3+q_4) \\ l_1\cos(q_1) + \ldots + l_4\cos(q_1+q_2+q_3+q_4) & \ldots & l_4\cos(q_1+q_2+q_3+q_4) \end{bmatrix} \end{aligned} \tag{2.3}$$

The typical least squares pseudoinverse is a way to solve for the inverse of a non-square matrix that minimizes the solution's sum-of-squares norm. The use of a right Moore-Penrose inverse [18] is required because equation (2.3) has more columns then rows, which is the least squares norm as seen in equation (2.4).

$$J^{\dagger} = J^T(JJ^T)^{-1} \tag{2.4}$$

Alternatively, the infinity norm minimizes the maximum value of the solution. The approach from [7] to solve the infinity norm begins with equation (2.5).

$$\begin{aligned} &\text{Minimize:} \quad ||\dot{\mathbf{q}}||_\infty \\ &\text{Subject to:} \quad J(\mathbf{q})\dot{\mathbf{q}} = \mathbf{v} \end{aligned} \tag{2.5}$$

where $||\dot{\mathbf{q}}||_\infty$ is the infinity norm of the joint velocity. Equation (2.5) is equivalent to equation (2.6)

$$\begin{aligned} \text{Minimize:} \quad & max(|\dot{q}_i|) \\ & i = 1, 2, ..., n \\ \text{Subject to:} \quad & J(q)\dot{q} = \mathbf{v} \end{aligned} \tag{2.6}$$

where n is the number of links the robotic arm has. The "max" operator in equation (2.6) can be replaced with a slack variable, s, and rewritten as follows in equation (2.7)

$$\begin{aligned} \text{Minimize:} \quad & s \\ \text{Subject to:} \quad & \begin{bmatrix} -I & 1_{n\times 1} \\ I & 1_{n\times 1} \end{bmatrix} \begin{bmatrix} \dot{q} \\ s \end{bmatrix} \geq 0_{2n\times 1} \\ & \begin{bmatrix} J(q) & 0_m \end{bmatrix} \begin{bmatrix} \dot{q} \\ s \end{bmatrix} = \mathbf{v} \end{aligned} \tag{2.7}$$

where m is the number of orthogonal axes being utilized in the operational space. With this formulation the problem can be posed as a standard linear programming problem format of equation (2.8).

$$\begin{aligned} \text{Minimize:} \quad & c^T y \\ \text{Subject to:} \quad & A_1 y \geq b_1 \\ & A_2 y = b_2 \end{aligned} \tag{2.8}$$

where the variables in equation (2.8) align with the matrices and vectors seen in equation (2.7) and $c = \begin{bmatrix} 0_n & 1 \end{bmatrix}^T$. The associated dual problem [5] is now given as follows in equation (2.9)

$$\begin{aligned} \text{Minimize:} \quad & b_2^T z_2 \\ \text{Subject to:} \quad & A_1^T z_1 + A_2^T z_2 = c \\ & z_1 \geq 0 \\ -\infty \leq z_2 \leq \infty & \end{aligned} \tag{2.9}$$

where z_1 and z_2 are dual decision variables.

The algorithm utilized by [7] to solve the dual problem can be seen in equation (2.10) and is followed by Figure 2.2, which is visualization of recurrent network. Additionally, how the recurrent neural network solution is embedded into the simulation for the robotic arm can be seen in Figure 2.3.

$$\begin{aligned} \frac{dy}{dt} &= -\mu[c(c^T y - b_2^T z_2) + A_1^T (A_1 y)^- + A_2^T (A_2 y - b_w)] \\ \frac{dz_1}{dt} &= -\mu[(z_1)^- + +A_1 (A_1^T z_1 + A_2^T z_2 - c)] \\ \frac{dz_2}{dt} &= -\mu[-b_2 (c^T y - b_2^T z_2) + A_2 (A_1^T z_1 + A_2^T z_2 - c)] \end{aligned} \tag{2.10}$$

where $(x)^- = (x_1^-, x_2^-, ..., x_n^-)^-$ with $x_i^- = min(0, x_i)$ and μ in the equations is a artificial gain applied to the system to accelerate the convergence to a steady-state solution.

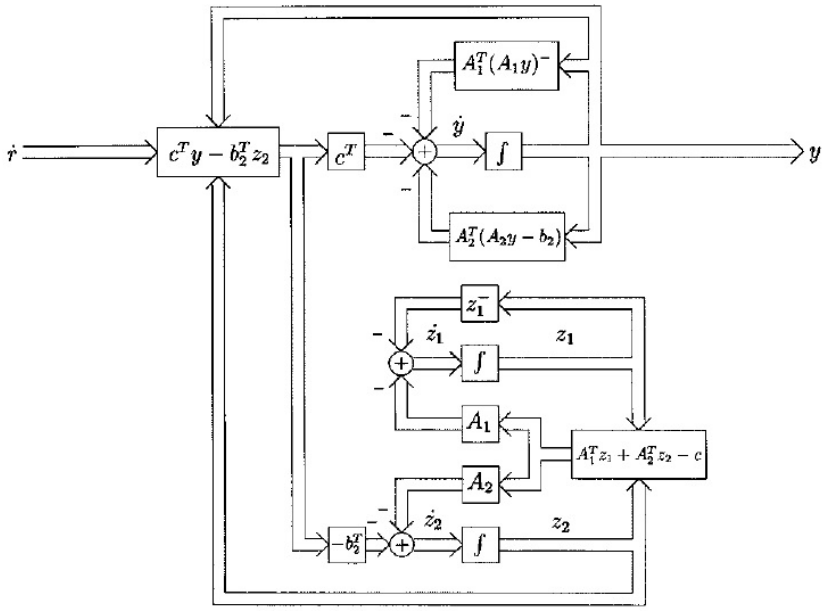


Figure 2.2. Recurrent Network Architecture. Source: [7].

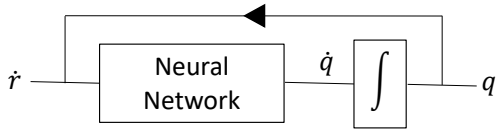


Figure 2.3. Recurrent Neural Network as Part of a Maneuver Simulation Loop. Adapted from [7].

2.2 Simulation and Results

The maneuver depicted in Figure 2.1b is mapped to the joint space using equation (2.2), where the pseudoinverse calculation is performed using either equation (2.4) for the least squares allocation or equation (2.10) for the infinity norm allocation. The manipulator's initial position is $[q_1 \quad q_2 \quad q_3 \quad q_4]^T = [-\pi/6 \quad \pi/6 \quad \pi/6 \quad \pi/6]^T$. The recurrent neural network is simulated using MATLAB's ode45 function, a simulation time of 0.01 seconds, and $\mu = 50000$. The gain used is the same gain as in [7]. The robot motion was simulated using MATLAB's ode1 fixed time step solver with a time step of 0.1 seconds.

Figure 2.4 shows the resulting link velocities utilizing both the least squares and infinity norm allocation method. It is observed that the infinity norm is working as intended for all joints except the first and last. The velocities produced for the first joint are similar and with the last joint the overall velocities incurred are small. While the results from [7] cannot be reproduced exactly in this thesis, since the recurrent network run time was not given, the results presented in Figure 2.4 of the joint motion look similar to the profiles shown in [7]. Analysis of results in further sections will show that convergence of the network was not fully achieved in [7] and it is unknown why the authors of [7] did not pursue this further.

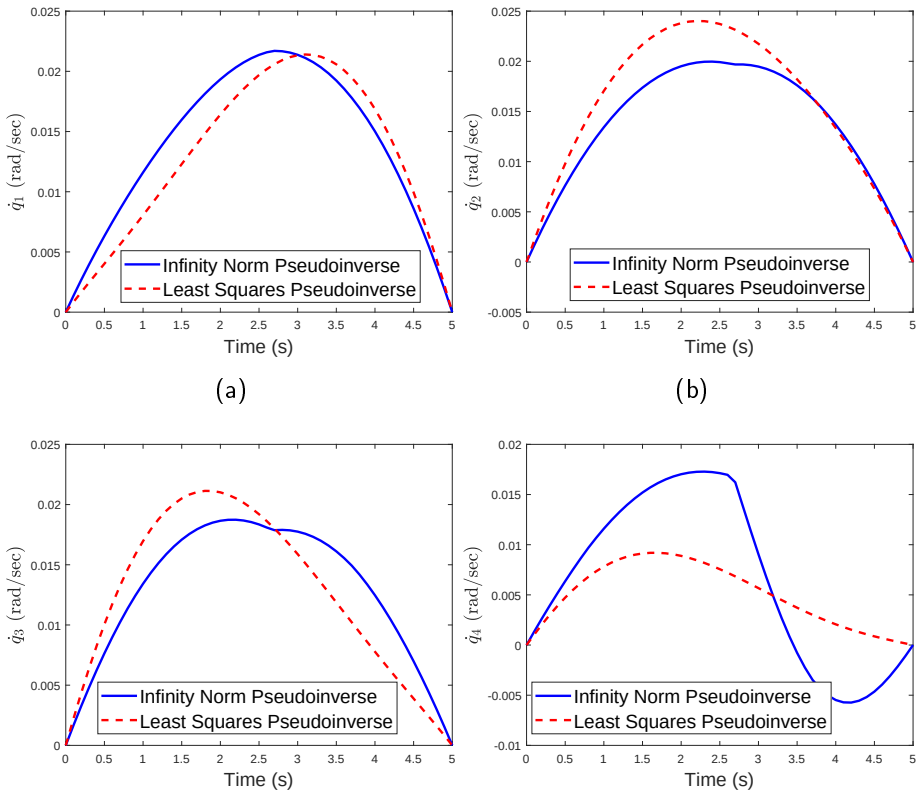
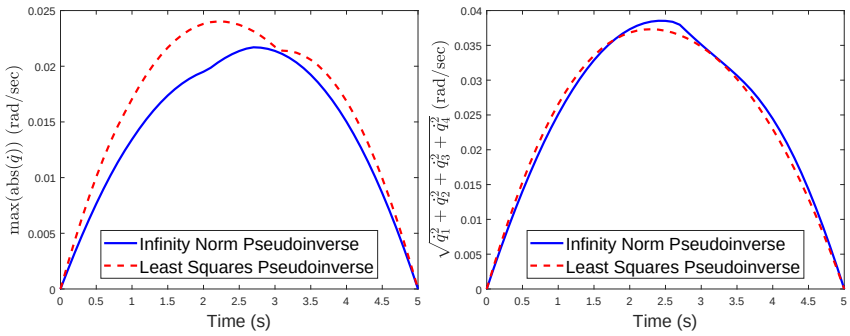


Figure 2.4. Link Velocities: (a) Link 1 Velocity Trajectory; (b) Link 2 Velocity Trajectory; (c) Link 3 Velocity Trajectory; (d) Link 4 Velocity Trajectory.

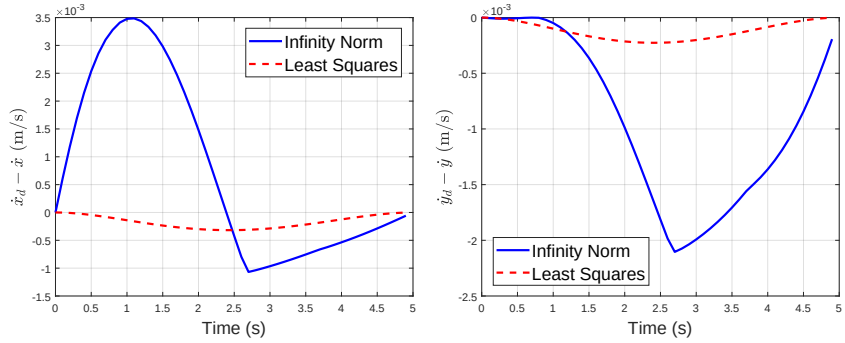
To compare the two allocation methods, the maximum absolute velocity of all joints and the square root of the sum of squared velocities are plotted against time in Figure 2.5. In Figure 2.5a, the maximum velocity experienced by the entire manipulator by the infinity norm allocation method is less. These results shows that even though link four experienced a larger velocity with the least squares allocation, the maximum joint velocity experienced by the entire robotic arm is lower, as is expected with this allocation method. Similarly, least squares allocation minimized the sum of squares better than the infinity norm allocation.



(a) (b)

Figure 2.5. Joint Velocity Comparison: (a) Max Link Velocity; (b) Norm of Link Velocities.

Lastly, the error between the simulation results ($\dot{x}_{simulation}$) and desired profiles ($\dot{x}_{desired}$) are compared in Figure 2.6 by taking the difference between the two. Numerical differentiation of the end effector position was used to approximate the velocities, and as such, the last data point could not be collected. While both allocation methods produce very little error, the least squares allocation method outperforms the infinity norm allocation with a whole magnitude less in error. However, while the least squares allocation fits the profile better, the infinity norm will produce torques that can be better realized by the motors of the robotic manipulator or allow for the selection of smaller motors.



(a) (b)

Figure 2.6. End Effector Velocity Profile Error Comparison: (a) End Effector X-Velocity Error ($\dot{x}_{desired} - \dot{x}_{simulation}$); (b) End Effector Y-Velocity Error ($\dot{y}_{desired} - \dot{y}_{simulation}$).

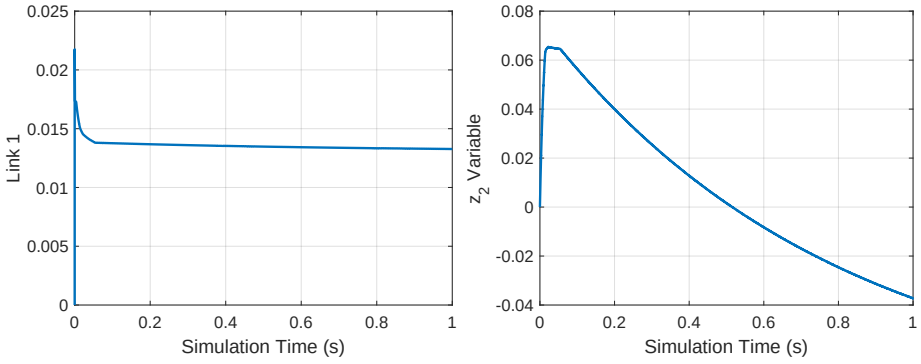
2.3 Improving Recurrent Neural Network Performance

The prior section focused on trying to obtain the results seen in [7] as closely as possible. However, from the examination of Figure 2.6, the implementation of the recurrent neural network did not exactly replicate the desired velocity profiles of the end effector. This section will explore two ways to improve the output of the recurrent neural network while also reducing the amount of time it takes MATLAB to finish the simulation of the maneuver. A major factor that plays into the results produced by the recurrent neural network is its run time. Recurrent neural network run time refers to the time span that the ode45 function uses to integrate the dynamic functions of equation (2.10). Execution time will refer to the amount of time it takes the simulation to finish computing the control trajectories of each of the joints.

2.3.1 Increasing Recurrent Network Run Time

To improve the performance of the recurrent neural network, the first modification that will be applied is an increase in run time. The simple act of giving the dynamic equations more time to run will allow for convergence to occur.

Figure 2.7 shows the convergence of the first link's position and the last variable of the z_2 equation of a recurrent neural network that had a run time of 1 second. This iteration of the recurrent neural network was solving for peak velocities of the maneuver seen in Figure 2.1b. The first link's velocity appears to have mostly leveled out while the last z_2 variable has not even begun to settle. The recurrent neural network in the prior section only ran 0.01 and would not have even seen convergence in the velocity of link one, thus more error was introduced to the final solution than what is depicted in Figure 2.7.



(a) (b)

Figure 2.7. Convergence Plots of Recurrent Neural Network: (a) q_1 Convergence; (b) z_2 Last Variable Convergence.

Table 2.1 shows the execution time for the various recurrent neural network run times chosen. While longer run times resulted in better convergence, they also have the side effect of increasing the order of magnitude of executed time taken to perform for the full five-second maneuver of the robotic manipulator. The results show that the recurrent neural network is ill-suited for simulation and may be better suited for Very Large Scale Integrated (VLSI) circuits or Field Programmable Analog Arrays (FPAA), which could perform the recurrent neural network equations faster.

Table 2.1. Execution Time of Recurrent Neural Network Convergence.

Simulation Time (s)	Execution Time (s)
0.01	~15
0.1	~200
1	~2500

The error of the desired velocity profile for different run times can be seen in Figure 2.8. Increasing the recurrent neural network run time by an order of magnitude corresponds to halving the error with respect to the desired manipulator velocity profile. These results indicate that increasing the run time of the recurrent neural network is not a efficient way of ensuring the recurrent neural network has reached convergence. Furthermore, the increased run time of the recurrent neural network comes with the cost of increased simulation execution time time.

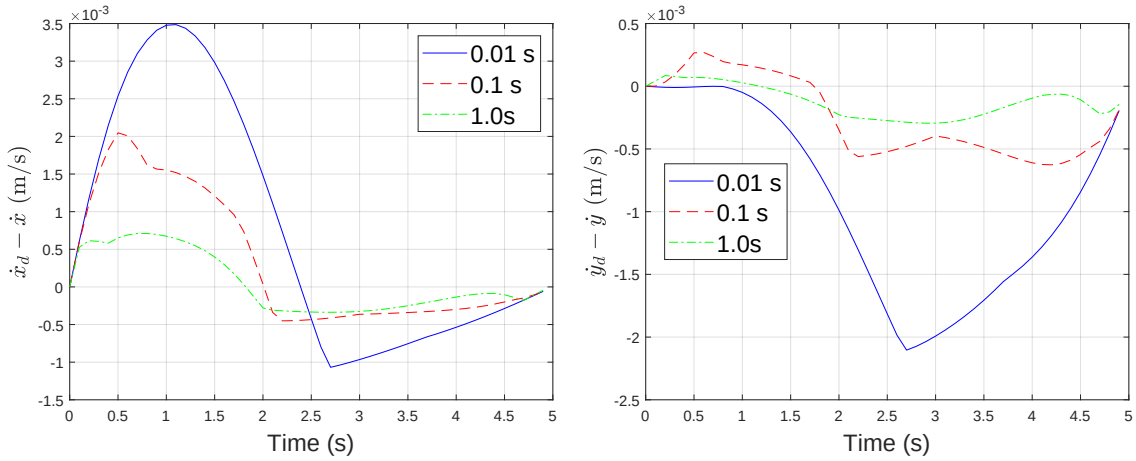


Figure 2.8. Simulation Time Velocity Error Profile Comparison: (a) End Effector X-Velocity Error ($\dot{x}_{desired} - \dot{x}_{simulation}$); (b) End Effector Y-Velocity Error ($\dot{y}_{desired} - \dot{y}_{simulation}$).

Figure 2.9 shows the maximum link velocity of the robotic arm for the various recurrent neural network run times. The increase in recurrent neural network run time results in lower overall maximum velocities for the joints, illustrating that the infinity norm solution has converged further.

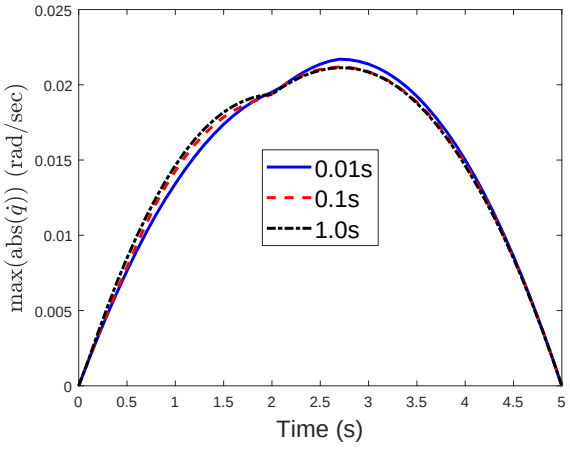


Figure 2.9. Maximum Joint Velocity of Robotic Arm Using Various Recurrent Neural Network Run Times.

2.3.2 Tuning the Recurrent Neural Network Gains

The prior section focused on increasing the run time of the recurrent neural network to try and achieve better results. Another approach that splits the scalar constant gain into a vector of three gains (one for each variable y, z_1, and z_2) could be used to accelerate the convergence of the recurrent neural network. However, choosing three different non-zero constant gains in such a balance to help expedite convergence is no easy feat. The selection of gains is further compounded by the massive gain of 50000 initially chosen for the system. A sample space of gains was generated using the DoE method [19] to efficiently test which combination of gain settings would work best for the recurrent neural network.

The DoE method is utilized in machine learning to generate sample points from which to train the neural network. Specifically, the Latin hypercube [19] is the specific DoE technique that was utilized to choose the gains for the recurrent neural network in this thesis. The data points in a Latin hypercube are randomly put in the sample space, filling it, and the distributing of the points is done by selecting statistical paramaters to optimize through a set number of iterations.

MATLAB's Latin hypercube design function was utilized, and the statistical parameters that can be chosen to distribute the data points are to minimize the correlation or maximize the minimum distance between the points. Figure 2.10 depicts a Latin hypercube which utilizes a maximization of the minimum distance between points.

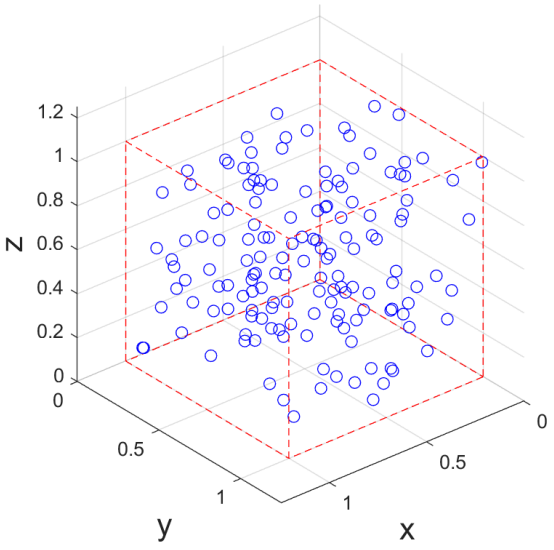
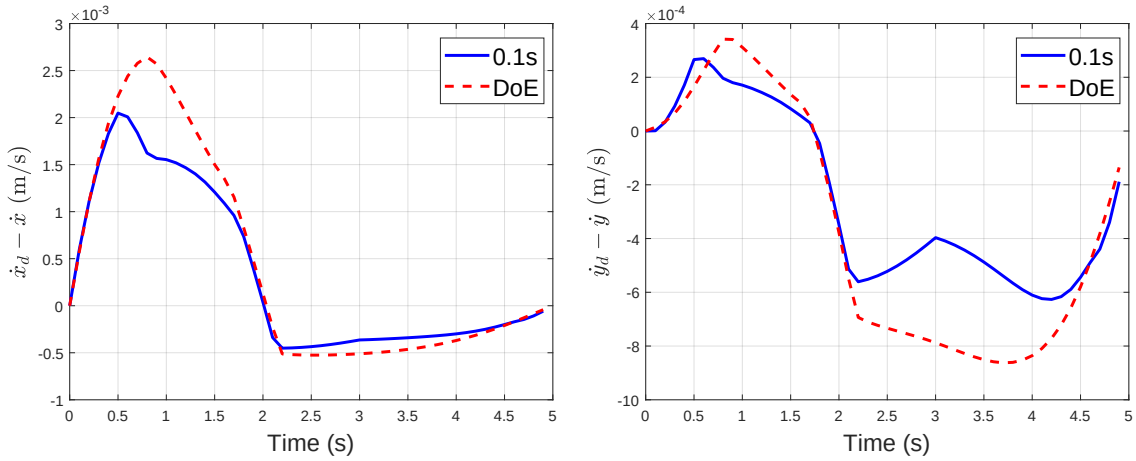


Figure 2.10. Latin Hypercube.

Note that while Figure 2.10 only uses three variables, describing a box, the Latin hypercube can be used for any number of variables. Initial testing revealed that the gain of 50000 was too large, producing no noticeably significant results. Therefore, the DoE was scaled down to between 0 and 10000 for the gains, utilizing 3000 points to fill up the design space.

Once the sample space was created, each design point was then run through a MATLAB script that measured the time it took the recurrent network to converge. The output of each run was compared to a solution of the original recurrent neural network utilizing the gain of 50000 and a run time of 100 seconds to ensure convergence. The design point that produced the closest result in the shortest amount of time was then chosen to run the overall maneuver.

Figure 2.11 shows the error in the velocity profiles of the end effector using the gains determined by the DoE with a recurrent neural network run time of 30 seconds. The DoE method produced results that closely matched the output of the recurrent neural network that ran for 0.1 seconds. The settings chosen by the DoE resulted in a control trajectory that took the execution time of 86 seconds for the simulation to produce, which is less than half of the execution time taken by the recurrent neural network run time of 0.1 second from the prior section.



(a) (b)

Figure 2.11. DoE End Effector Velocity Error Profile Comparison: (a) End Effector X-Velocity Error ($\dot{x}_{desired} - \dot{x}_{simulation}$); (b) End Effector Y-Velocity Error ($\dot{y}_{desired} - \dot{y}_{simulation}$).

2.4 Manipulator Singularities

In robotics a singularity occurs when the Jacobian becomes rank deficient and the maneuverability of the robotic arm is reduced since no direction can be enforced upon the manipulator [17]. The maneuver recreated from [7] did not encounter singular positions along the paths taken. To incur a singular condition an initial singular condition of $[q_1 \;\; q_2 \;\; q_3 \;\; q_4]^T = -0.01\pi/180 * [1 \;\; 1 \;\; 1 \;\; 1]^T$ was used with the velocity profiles of the end effector still being the same as before.

Figure 2.12 depicts a singularity measure of the Jacobian throughout the maneuver. It appears both allocation methods start at a singular condition with the least square allocation exiting the singular condition abruptly while the infinity norm allocation takes longer to exit and smoothly transitions out.

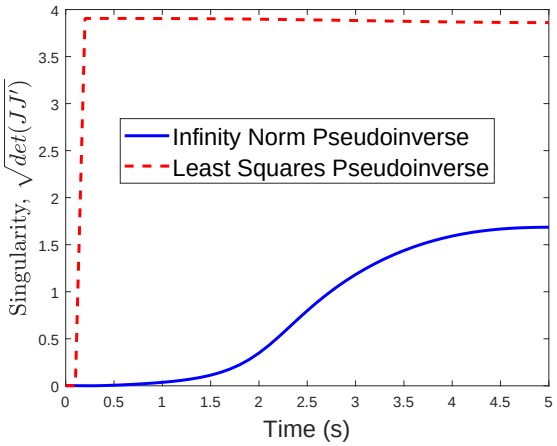
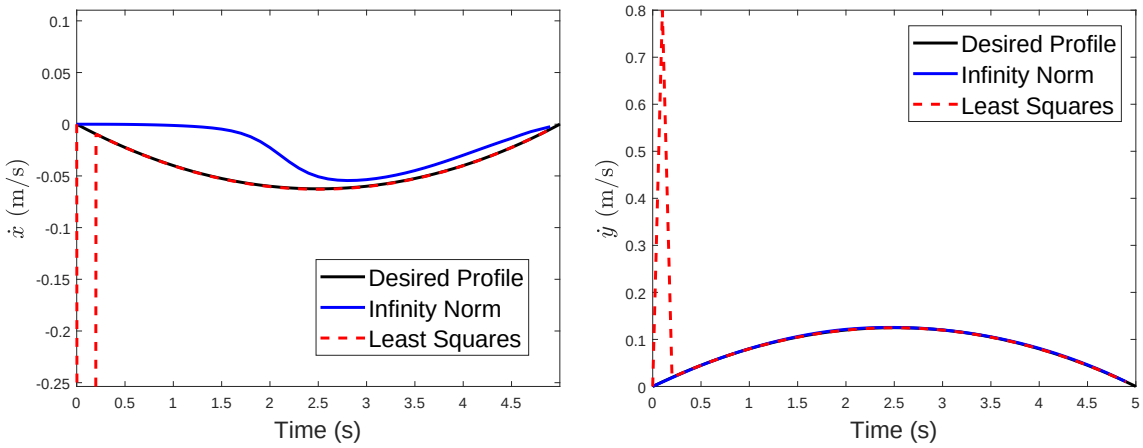


Figure 2.12. Measure of Jacobian Singularity.

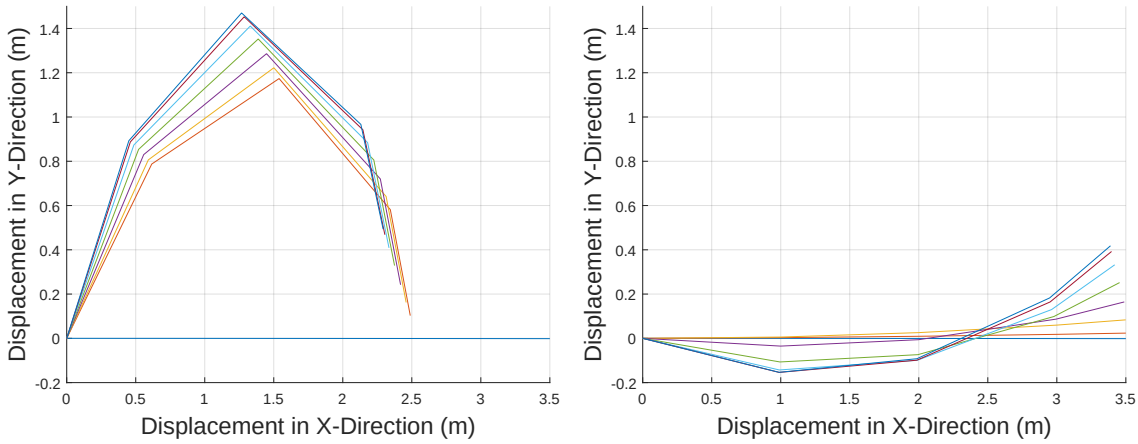
When looking at Figure 2.13, the least square allocation has a sharp change in the velocity of the end effector before transitioning back to following the desired velocity profiles. A closer view of the dynamics was required in Figure 2.13a to make comparisons between the two allocation methods, since the spike in x-velocity of the least squares pseudoinverse was so severe. The infinity norm allocation follows the y-velocity trajectory closely, while the x-velocity trajectory has issues keeping up with desired profile.



(a) (b)

Figure 2.13. Singularity Velocity Comparison: (a) End Effector X-Velocity; (b) End Effector Y-Velocity.

The sudden change in the velocities does not translate well in the prior figures, so the manipulator's overall motion was plotted in Figure 2.14 to give a better sense of what is occurring during the maneuver. It can be observed that the least square allocation starts off with the manipulator aligned with the x-axis and than the orientation jumps dramatically to the curved position. After the jump the manipulator begins to smoothly transition throughout the rest of the maneuver. However, the infinity norm allocation slowly adds curvature throughout the maneuver while trying to get back the desired x and y velocity profiles. It can be argued that while the infinity norm never fully matches the desired x-velocity profile, it is better to avoid a large jump in velocity, which is potentially damaging to the robotic manipulator.



(a) (b)

Figure 2.14. Manipulator Positing using Infinity Norm: (a) Manipulator Positing using Least Squares; (b) Manipulator Positing using Infinity Norm.

2.5 Summary

The infinity norm allocation did not closely match the desired profile when a singularity was introduced but the smooth motion of the end effector velocity can be realized in a real system. While the least squares allocation works faster, the velocity can jump dramatically when in a singular configuration. The sharp jump in the velocity of the least squares allocation could not be achieved by the motors in an actual system, this infers issues with the allocation method keeping up with the desired end effector velocity profile. The added

benefit with the infinity norm allocation having the smaller required joint velocities is that lighter torque motors can actuate the robotic manipulator's joints and save the designer the added constraint of weight and power margin.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
A Recurrent Network for Infinity Norm Control Allocation in Spacecraft Attitude Control

The previous chapter focused on the current applications of recurrent neural networks in the field of robotics. The previous chapter also further explored the design space by looking at the artificial gains employed by the neural network and introducing singularity into the system. This chapter will apply the recurrent neural network design by developing the mathematics to apply it to spacecraft orientation control utilizing a reaction wheel momentum exchange system. The infinity norm pseudoinverse produced by the recurrent neural network will be inserted into a simulation of an imaging spacecraft in orbit around Earth. The simulation will employ control closed loop feedback and then be compared to the least squares pseudoinverse. Lastly, the recurrent neural network will be applied to a CMG system to see how the infinity norm and least squares pseudoinverse react when the system reaches singularity. A version of this chapter was published as the paper AAS 20-713 at the 2020 American Astronautical Society/American Institute of Aeronautics and Astronautics Astrodynamics Specialist Conference [20].

3.1 Spacecraft Attitude Control

Quaternion feedback, where the quaternion is a $4x1$ vector, is typically used for maneuvering a spacecraft that is trying to point a sensor at a specific object [2]. The quaternion is useful for pointing applications because it allows the spacecraft to rotate through large angles without the singularities associated with Euler angles. The use of quaternions is also minimally taxing to the flight software in terms of computations performed. In [2] the method for quaternion feedback is described. Equation (3.1) depicts the calculation for error quaternion, $\mathbf{q_e}$, which describes the rotation the spacecraft has to execute to match the commanded quaternion, $\mathbf{q_c}$, of the target. Vector $\mathbf{q}$ is the instantaneous orientation of the spacecraft as the reorientation maneuver of the space is being executed. The commanded quaternion will have to be recalculated throughout the maneuver since the spacecraft will be moving along its orbit changing the relative position of the target to the spacecraft.

$$\mathbf{q_e} = \begin{bmatrix} q_{4c} & q_{3c} & -q_{2c} & -q_{1c} \\ -q_{3c} & q_{4c} & q_{1c} & -q_{2c} \\ q_{2c} & -q_{1c} & q_{4c} & -q_{3c} \\ q_{1c} & q_{2c} & q_{3c} & q_{4c} \end{bmatrix} \mathbf{q} \tag{3.1}$$

Equations (3.2) is Euler's equation for rotational motion in the absence of external torque and equation (3.3) defines the control law that governs the quaternion feedback law. In equation (3.2), $\mathbf{I}$ is the inertia tensor of the spacecraft, $\boldsymbol{\omega}$ is the 3 dimensional angular velocity of the spacecraft, $\mathbf{h_{RW}}$ is the angular momentum of the individual reaction wheels, and $\mathbf{Z}$ is the matrix that translates the individual reaction wheel spin axes to the three axis body-fixed reference frame. Lastly $\mathbf{u}$ is the control torque applied to the system. The control torque is further defined in [2] and can be seen in equation (3.3) where $\mathbf{K}$ and $\mathbf{C}$ are proportional gain matrices.

$$\mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega} + Z\mathbf{h_{RW}}) = \mathbf{u} \tag{3.2}$$

$$\mathbf{u} = K \begin{bmatrix} q_{1e} & q_{2e} & q_{3e} \end{bmatrix}^T - C\boldsymbol{\omega} \tag{3.3}$$

The state-space model for a spacecraft can be seen in equation (3.4) which helps to illustrate the point that while the control law deals with body torques, the actual driver for the dynamics requires the torques to be mapped to the individual reaction wheels. Since the $\mathbf{Z}$ matrix that describes the transformation from the frame defined by the spin axes of the four reaction wheels to the three dimensional body reference frame is non-square means that an inverse is not unique. This means that the control allocation of the individual reaction wheels is dependent on the specific pseudoinverse used for control allocation.

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\boldsymbol{\omega}} \\ \dot{\mathbf{h}}_{\mathbf{RW}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\mathbf{q} \otimes \begin{bmatrix} \boldsymbol{\omega} & 0 \end{bmatrix}^T \\ \mathbf{I}^{-1}(\boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega} - \mathbf{Z}\mathbf{h}_{\mathbf{RW}})) + Z\dot{\mathbf{h}}_{\mathbf{RW}} \\ \boldsymbol{\tau}_{\boldsymbol{RW}} \end{bmatrix} \tag{3.4}$$

where $\boldsymbol{\tau}_{\mathbf{RW}}$ is the torque supplied by the reaction wheels.

3.2 Solutions for Pseudoinverse Control Allocation

The approach typically used to perform a pseudoinverse for attitude control is the Moore-Penrose psuedoinverse, which is a least squares allocation. Since the transformation matrix, $\mathbf{Z}$, from which the pseudoinverse calculation takes place has linearly independent rows means that this is a right-inverse pseudoinvers and takes of the form of equation (3.5). This solution to the pseudoinverse problem minimizes the sum of squares of the reaction wheel torques thereby minimizing the overall error to this solution for the commanded input torque.

$$Z^{\dagger} = Z^T(ZZ^T)^{-1} \tag{3.5}$$

Other methods of calculation minimize the error of the solution of the pseudoinverse for other costs. The infinity norm allocation will minimize the maximum torque commanded to the reaction wheels needed to achieve the desired body torques. This has the desirable effect of allowing the spacecraft to better utilize the reaction wheel maneuvering torque space [4]. In order to perform a infinity norm allocation control allocation, the following problem is solved:

$$\begin{aligned} &\text{Minimize:} \quad ||\boldsymbol{\tau}_{\mathbf{RW}}||_{\infty} \\ &\text{Subject to:} \quad Z\boldsymbol{\tau}_{\mathbf{RW}} = \boldsymbol{\tau}_{\mathbf{c}} \end{aligned} \tag{3.6}$$

where $||\boldsymbol{\tau}_{\mathbf{RW}}||_{\infty}$ is the infinity norm allocation of the reaction wheel torques and $\boldsymbol{\tau}_{\mathbf{c}}$ are the body control torques which are equivalent to the control vector $\mathbf{u}$ and is applied in equations

(3.2) and (3.3). Equation (3.6) is equivalent to the problem in equation (3.7).

$$\begin{aligned} \text{Minimize:} \quad & max(|\boldsymbol{\tau}_{\mathbf{RW},i}|) \\ & i = 1, 2, ..., N_w \\ \text{Subject to:} \quad & Z\boldsymbol{\tau}_{\mathbf{RW}} = \boldsymbol{\tau}_{\mathbf{c}} \end{aligned} \tag{3.7}$$

where N_w is the number of reaction wheels in the system in equation (3.7). However, this formulation does not consider the "finite" capability of the reaction wheel system, namely that $|\boldsymbol{\tau}_{\mathbf{RW},i}| \leq \boldsymbol{\tau}_{max}$ for all $i = 1, 2, ..., N_w$ and where $\boldsymbol{\tau}_{max}$ is the saturation torque where the reaction wheel has hit its maximum capacity to produce. To incorporate this constraint equation (3.7) can be modified to equation (3.8).

$$\begin{aligned} \text{Minimize:} \quad & max(|\boldsymbol{\tau}_{\mathbf{RW},i}|) + s_1 \\ & i = 1, 2, ..., N_w \\ \text{Subject to:} \quad & Z\boldsymbol{\tau}_{\mathbf{RW}} - s_1\hat{\boldsymbol{\tau}}_{\mathbf{c}} = \boldsymbol{\tau}_{\mathbf{c}} \\ & -\tau_{max} \leq \tau_{RW,i} \leq \tau_{max}, i = 1, 2, ..., N_w, s_1 \geq 0 \end{aligned} \tag{3.8}$$

where s_1 is a scalar slack variable [5] whose non-zero value ensures that the torque output of the reaction wheel array is aligned with the commanded torque when one or more wheels are saturated.

The "max" operator in equation (3.8) can be replaced by introducing a second slack variable, s_2, as follows in equation (3.9).

$$\begin{aligned} \text{Minimize:} \quad & s_1 + s_2 \\ \text{Subject to:} \quad & \begin{bmatrix} I_{N_w \times N_w} & 0_{N_w \times 1} & -1_{N_w \times 1} \\ -I_{N_w \times N_w} & 0_{N_w \times 1} & -1_{N_w \times 1} \\ 0_{1 \times N_w} & -1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\tau}_{\mathbf{RW}} \\ s_1 \\ s_2 \end{bmatrix} \leq 0_{2N_w+1 \times 1} \\ & \begin{bmatrix} Z & \hat{\boldsymbol{\tau}}_{\mathbf{c}} & 0_{3 \times 1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\tau}_{\mathbf{RW}} \\ s_1 \\ s_2 \end{bmatrix} = \boldsymbol{\tau}_{\mathbf{c}} \\ & -\tau_{max} \leq \tau_{w,i} \leq \tau_{max}, i = 1, 2, ..., N_w, s_1 \geq 0 \end{aligned} \tag{3.9}$$

Equation (3.9) is in the form of a standard linear programming problem with bounded variables and can be written in the following more generalized form:

$$\begin{aligned} &\text{Minimize:} && p^T x \\ &\text{Subject to:} && Ax \leq b \\ & && Cx = d \\ & && x^L \leq x \leq x^U \end{aligned} \tag{3.10}$$

where the variables in equation (3.10) align with the matrices and vectors seen in equation (3.9) and $p = \begin{bmatrix} 0_{N_w} & 1_{N_w+1} \end{bmatrix}^T$.

Linear programming problems such as those posed by equation (3.10) can be solved iteratively using algorithm such a MATLAB's "linprog," solver. However, even if a linear programming solver could be implemented on a spacecraft, it would be taxing on the in-flight processing capabilities, and there is a potential that a solution is never reached, causing the programming to be stuck in a loop.

As an alternative to utilizing the limited computational capacity of the flight computers, the control allocation can be determined by implementing a recurrent neural network in an electronic circuit. To develop such a network, the necessary conditions for optimization need to be developed. The Lagrangian, L, for the problem posed by equation (3.10) is given as:

$$L = p^T x - y(Ax - b) + z(Cx - d) - lx + ux \tag{3.11}$$

where $y \in \mathbb{R}^{2N_w+1x(n+1)}$, $z \in \mathbb{R}^3$, $l \in \mathbb{R}^{N_w+2}$, and , $u \in \mathbb{R}^{N_w+2}$. The associated dual problem is therefore given as:

$$\begin{aligned} &\text{Maximize:} && b^T z - lx^L - ux^U \\ &\text{Subject to:} && A^T y - C^T z + l - u - p = 0 \\ & && y \geq 0, l \geq 0, u \geq 0 \end{aligned} \tag{3.12}$$

where l and u are Karush–Kuhn–Tucker (KKT) multipliers and x^L and x^U represent the

lower and upper boundaries of the variables x, respectively. Many different approaches to developing a recurrent neural network for solving the primal-dual problem [5] of equations (3.10) and (3.12) exist. In this thesis, a modified solution of Hu and Zhang's [11] algorithm is used to perform the infinity norm allocation. The network dynamics are defined as seen in equation (3.13). The neural network dynamics are determined by taking the gradient of an energy function that goes to zero when the optimal point has been found [7]. By doing this the motion taken by the neural network follows the gradient down towards towards the minimum, which is the solution to the infinity norm pseudoinverse.

$$\begin{aligned} \frac{d\mathbf{x}}{dt} &= -\mu_x\{\mathbf{x} - P_x(x - p - A^T y + C^T z)\} \\ \frac{d\mathbf{y}}{dt} &= -\mu_y\{\mathbf{y} - (\mathbf{y} + A(P_x(x - p - A^T y + C^T z) - b)^+\} \\ \frac{d\mathbf{z}}{dt} &= -\mu_z\{CP_x(x - p - A^T y + C^T z) - d\} \end{aligned} \tag{3.13}$$

where $(x)^+ = (x_1^+, x_2^+, ..., x_n^+)^+$ with $x_i^+ = max(0, x_i)$ and μ is an artificial gain applied to the dynamics, with each equation having a different gain. $P_x(x - p - A^T y + C^T z)$ is a projection operator defined as:

$$P_{x_i}(x_i) = \begin{cases} x_i^L, & x_i < x_i^L \\ x_i, & x_i^U \leq x_i \leq x_i^U \\ x_i^U, & x_i > x_i^U \end{cases}$$

A schematic of the associated recurrent neural network is shown in Figure 3.1, using the nomenclature associated with the reaction wheel control allocation problem.

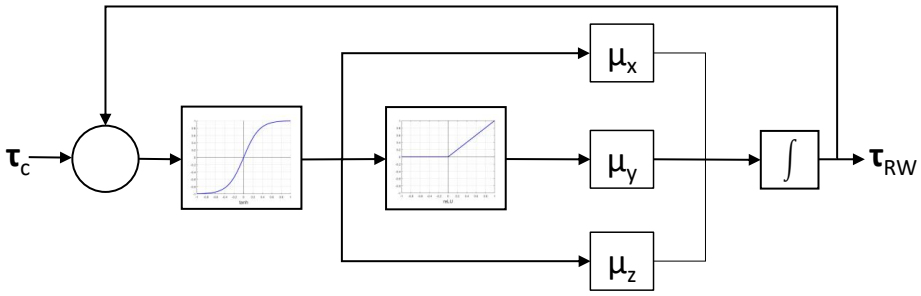


Figure 3.1. Neural Network Architecture for Infinity Norm Reaction Wheel Control Allocation.

The architecture shown in Figure 3.1 can be implemented as an analog circuit using operational amplifiers and other passive components to realize the functions of addition, multiplication, integration, inversion, saturation, etc. In this chapter, MATLAB's ode solvers were utilized to numerically integrate the equation, (3.13), as opposed to developing an electronic circuit. This is done to illustrate the concept and leaves an electronic implementation for future work.

In order to test the validity of the recurrent neural network solution, the network output was tested against a linear programming solution that utilized MATLAB's "linprog" solver. The maximum torque allowed for any of the reaction wheels was 1 Nm and a test torque command of $\begin{bmatrix} 2 & -2 & 2 \end{bmatrix}^T$ Nm was used to ensure reaction wheel torque saturation would be reached. The test utilized a 4 reaction wheel system selected to be representative of a typical pyramidal configuration [1] with the following properties:

$$Z = \begin{bmatrix} 0.6123 & 0.6123 & -0.6123 & -0.6123 \\ -0.6123 & 0.6123 & 0.6123 & -0.6123 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}$$

Figure 3.2 shows that all of the variables in the recurrent neural network have converged to a steady-state solution. In Figure 3.2a the reaction wheel torques do not exceed the +/- 1 Nm limit. In Figure 3.2c the constraint that each KKT multiplier has to be greater then 0 is also upheld. Furthermore, Table 3.1 shows that the recurrent network arrives at a solution with an error tolerance of $\times 10^{-5}$ to that of the reference solution using "linprog".

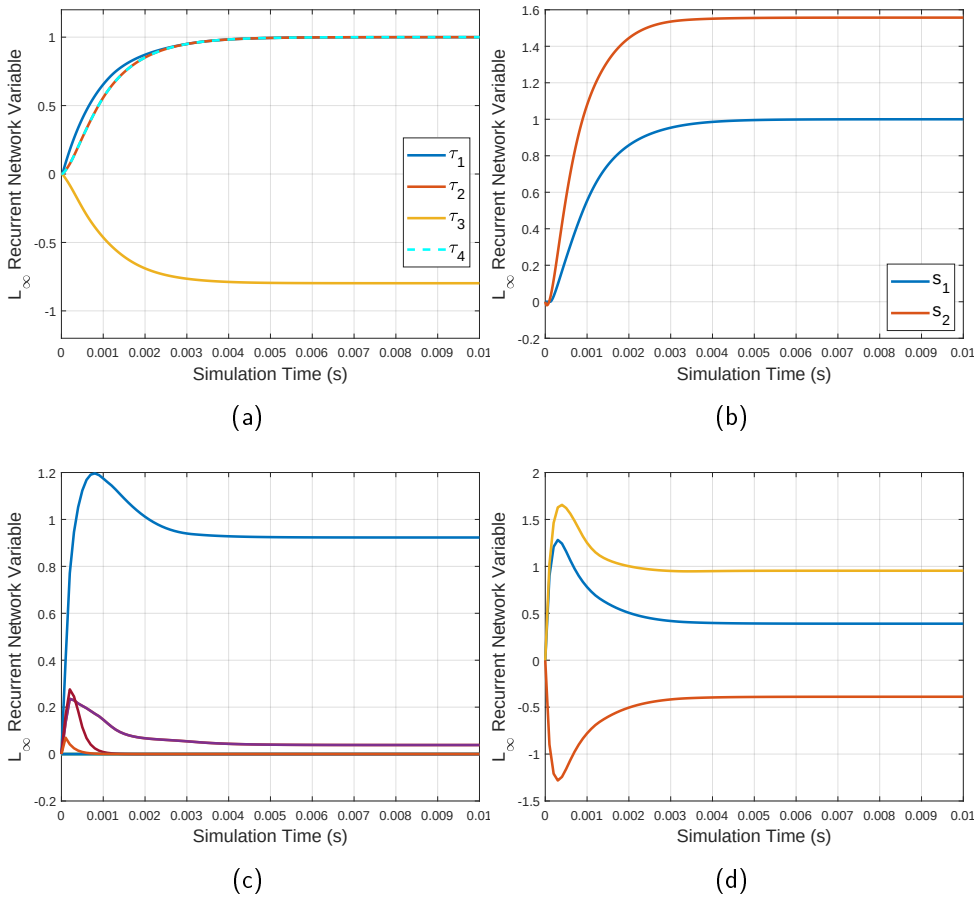


Figure 3.2. Time History of Recurrent Network Response for Infinity Norm Control Allocation: (a) Recurrent Network $\boldsymbol{\tau_{RW}}$ Variables; (b) Recurrent Network Slack Variables; (c) Recurrent Network y Variables; (d) Recurrent Network z Variables.

Table 3.1. Infinity Norm Comparison.

Reaction Wheel	Linear $L_{\infty,LINPROG}$	Neural Network $L_{\infty,NeuralNet}$
1	1.0000	1.0000
2	1.0000	1.0000
3	-0.7980	-0.7980
4	1.0000	1.0000
Error	$\sqrt{sum(L_{\infty,L} - L_{\infty,NN})^2}$	4.2947×10^{-5}

Lastly, to show the benefit of using the infinity norm over that of the least squares allocation of equation (3.5) the reaction wheel torque envelopes of both allocations were plotted. The reaction wheel system was the same pyramidal configuration used in Figure 3.3 and Table 3.1. The commanded torque envelope was defined over a unit sphere unit sphere with 3600 points. Figure 3.3a shows the solution using the pseudoinverse of equation (3.5) which is of a double pyramid defined by the +/- torque capability and orientation of each reaction wheel. Figure 3.3b is the envelope obtained by the recurrent neural network of the equations (3.12) and (3.13), which shows that the recurrent neural network produces the correct results as compared to [1]. Figure 3.3b also shows that the infinity norm adds an extra face to the envelope for each reaction wheel providing a larger torque envelope capability for the same reaction wheel system.

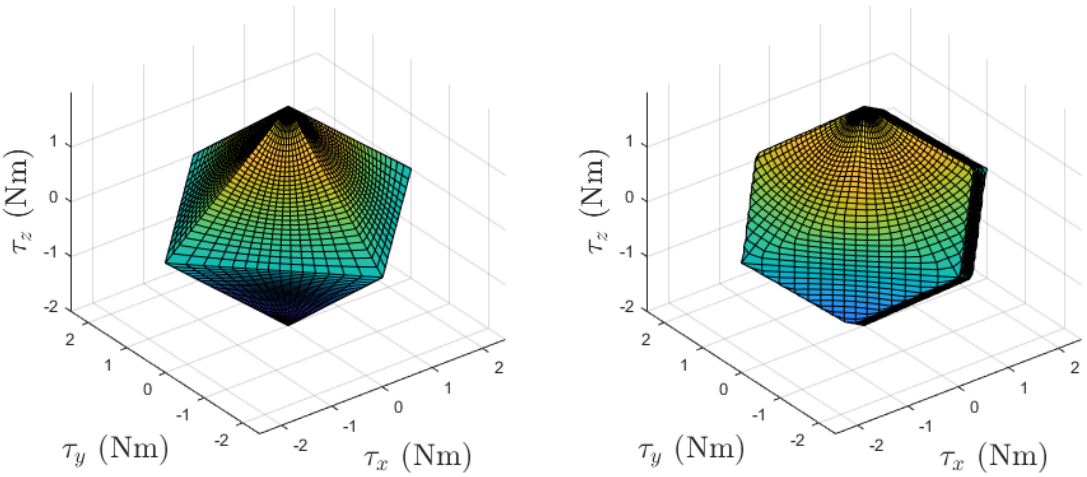


Figure 3.3. Reaction Wheel Torque Envelope: (a) Reaction Wheel Torque Envelope Using Least Squares; (b) Reaction Wheel Torque Envelope Using Infinity Norm Allocation.

3.3 Attitude Control Simulations

The simulation implemented to illustrate the performance of the infinity norm allocation allocation is that of a imaging satellite in a 650 kilometer orbit that is observing multiple points of interest on the Earth's surface. The total list of potential observations as well as the order of desired observations was randomly generated in a area roughly 1200 by

600 kilometers in size. The simulation was run for a total of 300 seconds of spacecraft maneuvering time to emulate a collection window. The spacecraft requires a 2 second lock on target to acquire an image. The 2 second count begins when the target is within 0.01° of pointing and $0.01^\circ/sec$ of rate error. Once a target has been locked on for 2 seconds the spacecraft will then proceed to acquire the next observation in the list of randomly prioritized observations.

The control architecture in Figure 1.1 was utilized, ignoring any error that might be introduced by attitude or rate sensor bias or noise. The on-board momentum exchange system utilizes 4 reaction wheels in a pyrimidal configuration. There are many variations of the control being applied in equation (3.3) that exist, such as the one employed by the Lunar Reconnaisance Orbiter [21], which utilizes a PID controller. The control law implemented in the PID controller for this simulation can be seen in equation (3.14).

$$\boldsymbol{\tau}_c = K_P I \mathbf{a}_{e,1} - K_R I \omega + K_I I \mathbf{a}_{e,2} + \boldsymbol{\omega} \times (I\boldsymbol{\omega} + Z\mathbf{h}_{\mathbf{RW}}) \tag{3.14}$$

where $\mathbf{a}_e = 2\mathbf{q}_{e,(1:3)} sign(q_{e,4})$ and K_P, K_R, and K_I are artificial gains applied to the system, similar to a PID controller . The method of calculating $\mathbf{q}_e$ given a pointing vector can be found in appendix A. In (3.14) $\mathbf{a}_{e,1}$ is proportionally limited to the bounds of +/-$1^\circ K_P/K_R$ and $\mathbf{a}_{e,2}$ is limited to the bounds of +/-$0.24/(250K_P)$. Furthermore, the variable K_I only turns on when the spacecraft boresight triggers the imaging device to start accumulating. All the constants for the spacecraft, the control law, and recurrent neural network can be found in Table 3.2.

Table 3.2. Parameters used for Closed Loop Simulation.

Variable	Value	Units
	Spacecraft Properties	
I	$\begin{bmatrix} 222.17 & 2.58 & -9.10 \\ 2.58 & 264.51 & 2.42 \\ -9.10 & 2.42 & 171.91 \end{bmatrix}$	Nms
Z	$\begin{bmatrix} 0.6123 & 0.6123 & -0.6123 & -0.6123 \\ -0.6123 & 0.6123 & 0.6123 & -0.6123 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}$	–
τ_{max}	0.6	Nm
	Control Law Constants	
K_P	0.4630	–
K_R	1.3609	–
K_I	0.0370	–
	Neural Network Gains	
μ_x	1087	–
μ_y	6894	–
μ_z	5185	–

3.3.1 Single Slew Maneuver

In order to validate the correct function of the recurrent neural network in the attitude control loop, an example slew maneuver was performed. Figures 3.4, 3.5, and 3.6 depict various responses of the spacecraft as it slewed its boresight to the target. Figure 3.4 shows the quaternion error as the spacecraft reorients itself to acquire the image. As expected the $\mathbf{q}_e = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$ signifying that the spacecraft has locked onto the target.

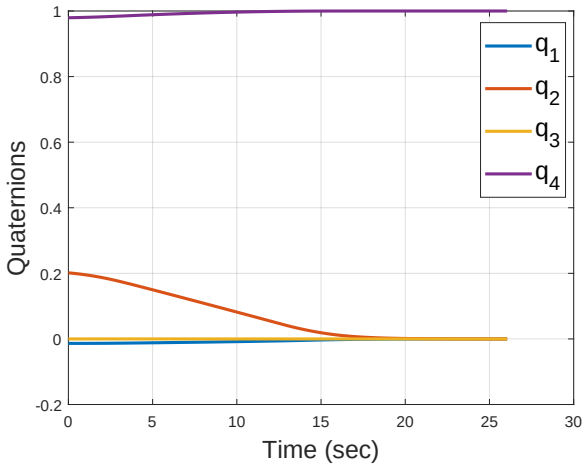


Figure 3.4. Quaternion Error for an Example Slew using an Infinity Norm Neural Network.

Figure 3.5 shows the angular rate of the spacecraft less the commanded angular rate required to lock on to the target. This was done to eliminate the angular rate bias that would be required to keep the spacecraft on target while it progresses along its orbit. As can be observed, the spacecraft reaches a zero state in its angular velocity as it locked onto the target.

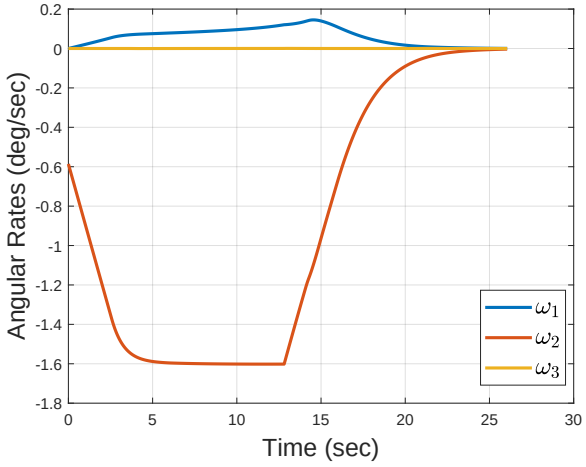


Figure 3.5. Relative Angular Rate for an Example Slew using an Infinity Norm Neural Network.

The corresponding angular momenta of the reaction wheels of the spacecraft can be seen in Figure 3.6. Note that in Figure 3.6, the reaction wheel momenta do not return to zero because the maneuver is not rest-to-rest.

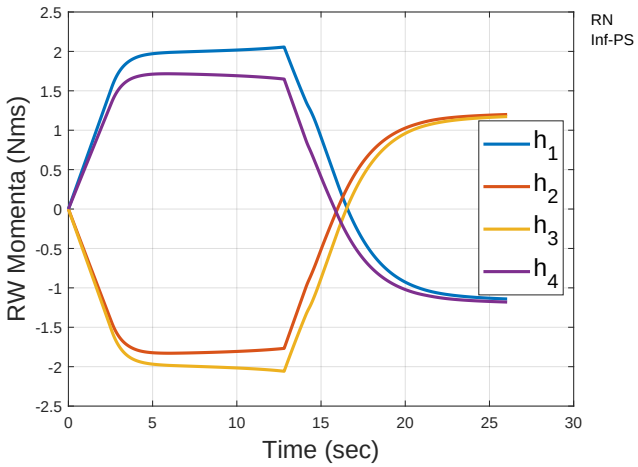


Figure 3.6. Reaction Wheel Momenta for an Example Slew using an Infinity Norm Neural Network.

3.3.2 Multiple Target Imaging Simulation

While the single slew maneuver helps to show that the recurrent neural network functions properly for spacecraft attitude control, the ability for it to work through multiple maneuvers is still required to be shown. Now the simulation of the spacecraft will be run for 300 seconds to try and obtain 10 image acquisition targets that were randomly generated in a space of 600 by 1200 kilometers and can be seen in Figure 3.7.

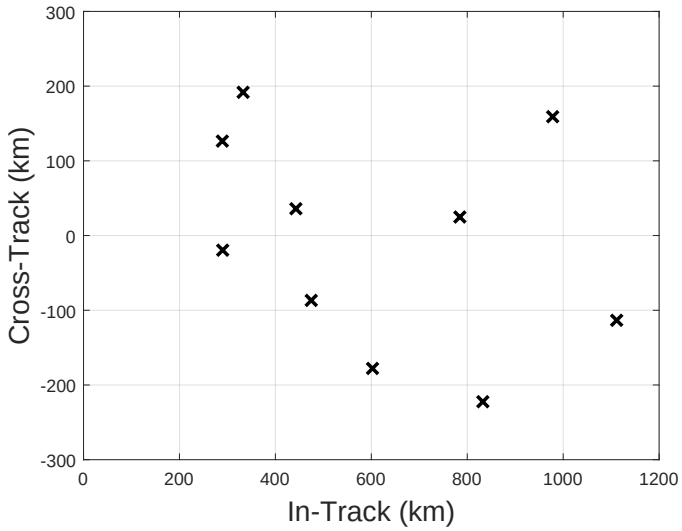


Figure 3.7. Image Observation Target Deck.

Figure 3.8 shows the commands produced by the recurrent neural network of the individual reaction wheels throughout all 300 seconds of maneuvering during an imaging window. All torque saturation bounds can be observed in Figure 3.8 and are held to without being exceeded. The commanded torques are also smooth without any nonidealities, such as chattering, observed.

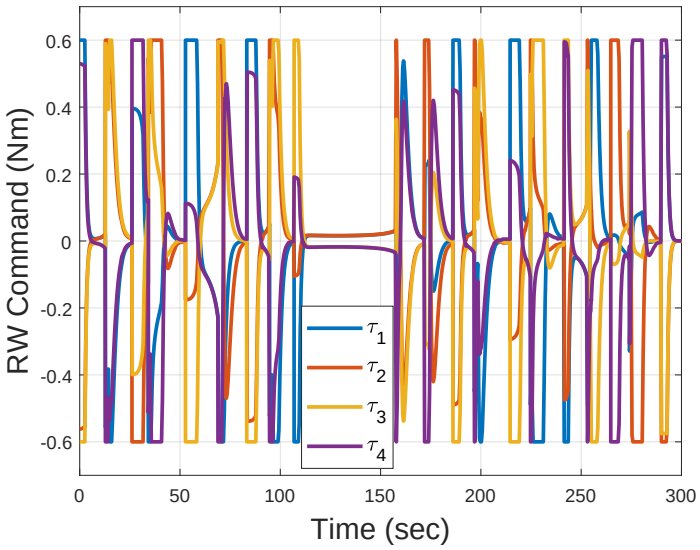


Figure 3.8. Reaction Wheel Commands for Imaging Simulation.

The next metric to be measured is the infinity norm performance against the least squares allocation and that is measured by how many observations each algorithm completed out of the 10 total that were sought. If the imaging device overshoots the target the imaging accumulator will pause until the spacecraft reacquires the target with the required accuracy. In Figure 3.9 it can be observed that both allocations each acquired and imaged all the targets. Both control allocation schemes perform similarly until the fifth target in which the least squares allocation is seen to lose lock on the target. Since the infinity norm allocation never lost lock it continues to gather the rest of the imaging opportunities leading the least squares allocation by 3-5 seconds.

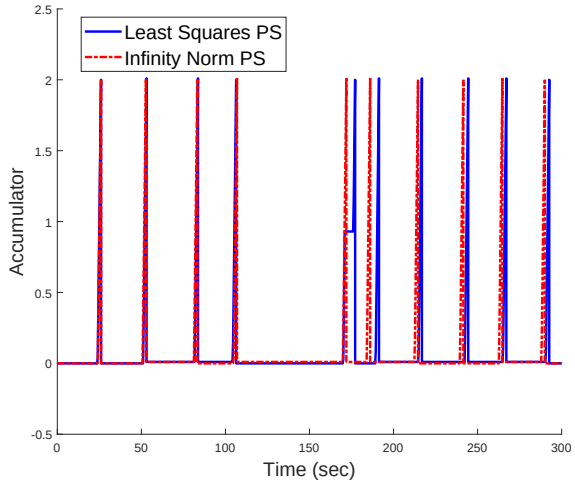


Figure 3.9. Image Accumulation over 300 Second Imaging Window.

Figure 3.10 depicts the magnitude of the body torques resulting from the commanded reaction wheel torques for both allocation types. Due to the the infinity norm having a larger torque envelope, the body torques incurred can be higher then the ones imposed by the least squares allocation and is evident in Figure 3.10.

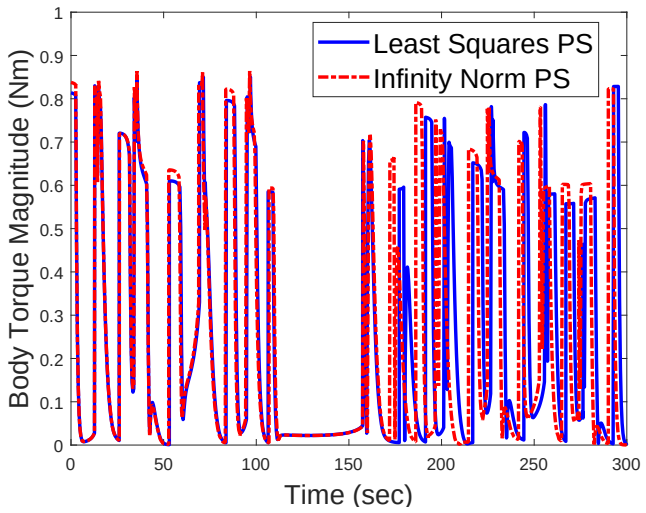


Figure 3.10. Comparison of Body Torque Magnitudes in Path Solution for Multi-Target Imaging.

In order to get a better view of what is occurring with the imaging satellite during the simulation, Figure 3.11 shows the trace of the boresight of the spacecraft as it travels from target to target. This overview of all 10 targets in Figure 3.11 does not appear to show much difference between the two allocations. The only thing to note is that the infinity norm is ahead of the least squares at the end of the 300 seconds. However, if a closer look is taken for the second and fifth targets acquired in the sequence of acquisitions, some interesting features can be seen.

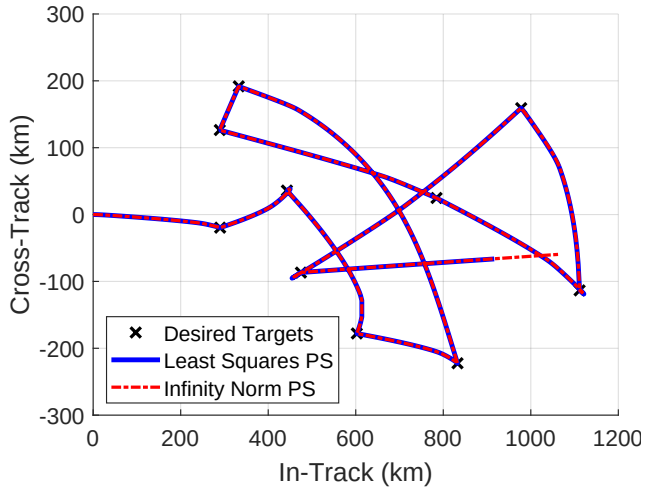


Figure 3.11. Boresight Path.

Looking closer at the acquisition of the second target in Figure 3.12a both allocations initially overshoot the target. However, using the infinity norm allocation results in a smaller overshoot than the least squares allocation. This is due to the larger torque capability that is enabled by the infinity norm allocation to slow down and reorient the spacecraft to acquire the target. The body torque magnitudes commanded during the entire maneuver from the first to second target can be seen in Figure 3.12b. The first 10 seconds show that the boresight for both allocation schemes are leaving the first target with similar angular velocities. The next portion of the maneuver shows both allocation schemes attempting to slow down the vehicle angular rate to try and start acquiring the target. It is here the infinity norm allocation commands higher torques that are available to it. This allows the infinity norm allocation to get a better inside track and overcome the overshoot sooner. This same type of overshoot situation occurs again on target eight and target ten.

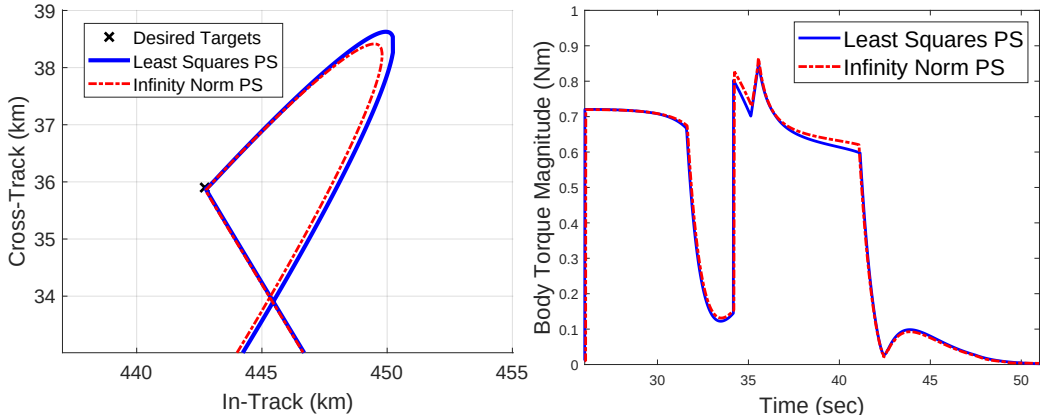


Figure 3.12. Target 2 Pointing Comparison: (a) Boresight Trace to Target 2; (b) Body Torque Magnitude to Target 2.

It can also be observed that the infinity norm performs better at target five than the least squares allocation. Using Figure 3.13a it can be seen that the boresight paths for both allocations never directly pass over the target even though they both acquire it; meaning that the boresight was still in the required pointing accuracy. However, the infinity norm allocation does not drift off as far from the target before it acquires the image, while the least squares allocation overshoots and has to turn back to reacquire the target. This can again be attributed to the larger torque envelope available to infinity norm allocation as seen

in Figure 3.13b. The little bit of extra torque extra torque authority made available by the infinity norm allocation allows the boresight trace to slow down just enough to get the 2 second lock on required so that the imaging controller can switch to the next target. Overall it is observed that the infinity norm allocation leads to better performance on the imaging task than the standard Moore-Penrose pseudoinverse.

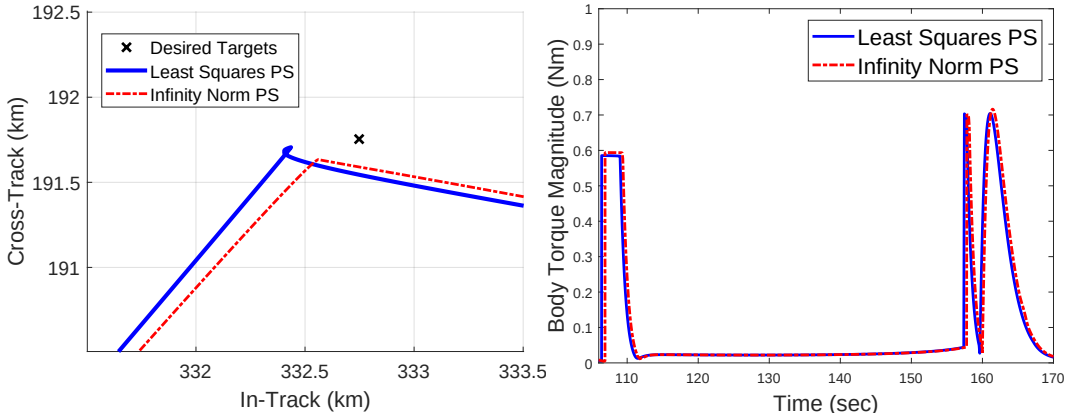


Figure 3.13. Target 5 Pointing Comparison: (a) Boresight Trace to Target 5; (b) Body Torque Magnitude to Target 5.

3.4 Control Moment Gyro and Singularity

The CMG momentum exchange system utilizes spinning wheels that create torques by gimballing the axis of rotation of the individual wheels, see Figure 3.14. Since torque is produced by moving individual CMGs the transformation matrix **Z** will change as control torques are implemented in the spacecraft.

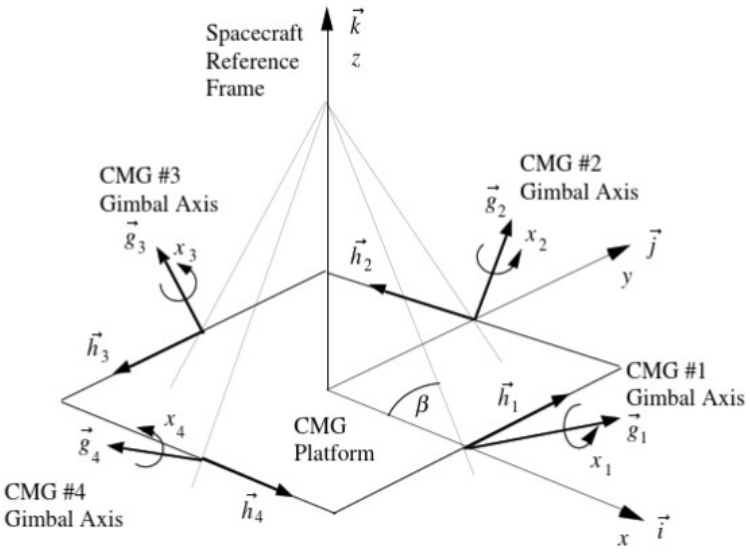


Figure 3.14. Pyramidal Single-Gimbal CMG Control System. Source: [17].

Equation (3.15) shows how the transformation matrix **Z** is defined by the positioning of the individual CMG's. The skew angle, β, describes the slope of the pyrimid face as seen in Figure 3.14. Each δ is the displacement of each h from the nominal position seen in Figure 3.14.

$$\mathbf{Z} = \begin{bmatrix} -\cos\beta\sin\delta_1 & -\cos\delta_2 & \cos\beta\sin\delta_3 & \cos\delta_4 \\ \cos\delta_1 & -\cos beta\sin\delta_2 & -\cos\delta_3 & \cos\beta\sin\delta_4 \\ \sin\beta\sin\delta_1 & \sin\beta\sin\delta_2 & \sin\beta\sin\delta_3 & \sin\beta\sin\delta_4 \end{bmatrix} \tag{3.15}$$

The torque in a CMG system must be calculated from the the rate at which each CMG is gimballing. Matrix $\mathbf{A}_{CMG}$, as defined in equation (3.16), is the transformation matrix that describes how this torque is being produced and from which the pseudoinverse will be calculated.

$$\mathbf{A}_{CMG} = \frac{\partial Z}{\partial \boldsymbol{\delta}} = \begin{bmatrix} -\cos\beta\cos\delta_1 & \sin\delta_2 & \cos\beta\cos\delta_3 & -\sin\delta_4 \\ -\sin\delta_1 & -\cos\beta\cos\delta_2 & \sin\delta_3 & \cos\beta\cos\delta_4 \\ \sin\beta\cos\delta_1 & \sin\beta\cos\delta_2 & \sin\beta\cos\delta_3 & \sin\beta\cos\delta_4 \end{bmatrix} \tag{3.16}$$

Like the robotic manipulator from Chapter 2, this type of system can encounter configurations that reach singularity. If singularity is encountered, the CMG system cannot provide a three dimensional torque in the Cartesian coordinate frame and the control authority is reduced down to a planar surface [2].

3.4.1 Problem Formation

A recurrent network is used to solve a version of the optimization problem given in [8] as seen in equation (3.17)

$$\begin{aligned} &\text{Minimize:} && \tfrac{1}{2}[\alpha||\dot{\boldsymbol{\delta}}||_2^2 + (1-\alpha)||\dot{\boldsymbol{\delta}}||_\infty^2] \\ &\text{Subject to:} && A_{CMG}\dot{\boldsymbol{\delta}} = \boldsymbol{\tau}_c \\ & && -\dot{\delta}_{max} \leq \dot{\delta} \leq \dot{\delta}_{max} \end{aligned} \tag{3.17}$$

where α represents a sliding scale between the least squares, $||\dot{\boldsymbol{\delta}}||_2$, and the infinity norm, $||\dot{\boldsymbol{\delta}}||_\infty$, for values between 1 and 0 respectively. Similar to the reaction wheel system the "max" operator is transformed using a slack variable, s_1, and to produce torque when the systems hits singularity an additional vector of slack variables, $\mathbf{s}_2$, is introduced and equation (3.17) can be rewritten into equation (3.18)

$$\begin{aligned} &\text{Minimize:} && \tfrac{1-\alpha}{2}s_1^2 + K\mathbf{s}_2 \\ &\text{Subject to:} && \begin{bmatrix} I_{N_w\times N_w} & -1_{N_w\times 1} & 0_{N_w\times N_w} \\ -I_{N_w\times N_w} & -1_{N_w\times 1} & 0_{N_w\times N_w} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\delta}} \\ s_1 \\ \mathbf{s}_2 \end{bmatrix} \leq 0_{2N_w\times 1} \\ & && \begin{bmatrix} A_{CMG} & 0_{3\times 1} & -I_{3\times 3} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\delta}} \\ s_1 \\ \mathbf{s}_2 \end{bmatrix} = \boldsymbol{\tau}_c \\ & && \begin{bmatrix} -\dot{\delta}_{max} \\ 0 \\ -\infty \end{bmatrix} \leq \begin{bmatrix} \dot{\boldsymbol{\delta}} \\ s_1 \\ \mathbf{s}_2 \end{bmatrix} \leq \begin{bmatrix} \dot{\delta}_{max} \\ \infty \\ \infty \end{bmatrix} \end{aligned} \tag{3.18}$$

where K is an artificial gain applied to the slack matrix, $\mathbf{s}_2$ to ensure a torque can be produced when singularity is encountered. Equation (3.18) can now be rewritten into the following

quadratic problem:

$$\begin{aligned}
&\text{Minimize:} && \tfrac{1}{2}x^T Q x \\
&\text{Subject to:} && Ax \leq b \\
& && Cx = d \\
& && x_{min} \leq x \leq x_{max} \\
&\text{Where:} && Q = \begin{bmatrix} \alpha I_{N_w \times N_w} & 0_{N_w \times 1} & 0_{N_w \times 3} \\ 0_{1 \times N_w} & 1-\alpha & 0_{N_w \times 3} \\ 0_{3 \times N_w} & 0_{3 \times N_w} & K I_{3 \times 3} \end{bmatrix}
\end{aligned} \tag{3.19}$$

where the variables in equation (3.19) where the variables in equation (3.10) align with the matrices and vectors seen in equation (3.18) and x is defined further in equation (3.20).

The imaging spacecraft simulation was modified to utilize a CMG system and the dynamical equations from [8], as seen in equation (3.20), were used for the recurrent neural network.

$$\begin{aligned}
\dot{u} &= \mu[g(EQ^{-1}E^T u - u) - EQ^{-1}E^T u] \\
x &= Q^{-1}E^T u \\
E &= \begin{bmatrix} A \\ C \\ I_{8\times 8} \end{bmatrix}
\end{aligned} \tag{3.20}$$

In equation (3.20), function $g(\cdot)$ is a hyperbolic tangent function, enforcing the system's limits as seen in equation (3.18).

3.4.2 Simulation Results

The dynamics of the recurrent network were propagated using through MATLAB's ode45 function. The simulation of the imaging spacecraft utilized MATLAB's ode1. Both α and K have been set to 0.5 for this simulation.

The first measurement this thesis looks at is how closely each allocation method gets to singularity and can be seen in Figure 3.15. One observation is that the least squares allocation never reaches singularity while the infinity norm allocation gets reasonably close a couple

of times. However, the infinity norm system seems to behave more stably over time.

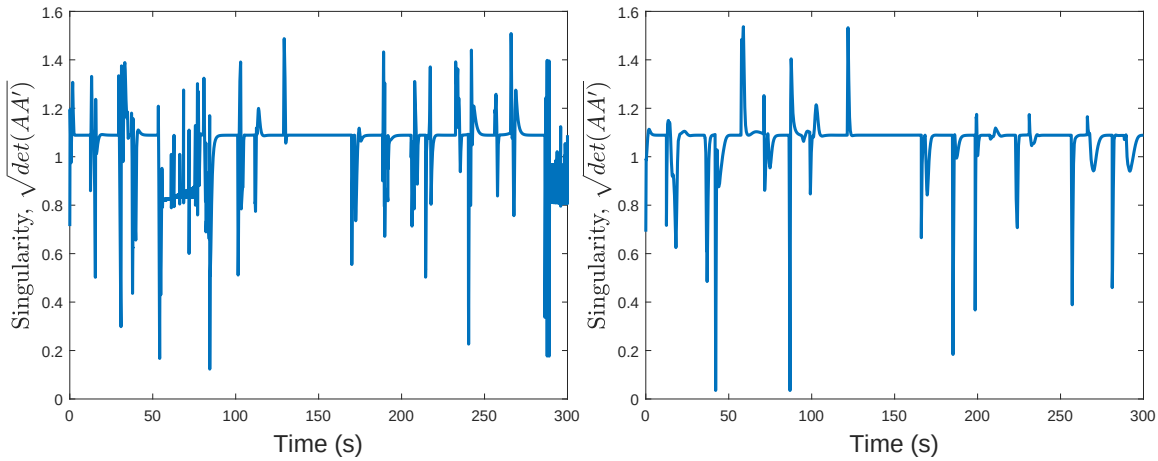


Figure 3.15. Singularity Comparison: (a) Least Squares Singularity Allocation; (b) Infinity Norm Singularity Allocation.

When observed in concert with Figure 3.16, the commanded rate each allocation produced, the stability of the infinity norm allocation becomes even more pronounced. It appears that the least squares allocation produces chatter in the commanded CMG rates whenever the system gets near singular. In contrast, even though the infinity norm gets much closer to a singular condition in a few instances, the commanded rates never chatter and remain smooth throughout the entire simulation. The lack of chatter illustrates the the recurrent neural network allocation is better suited for operating the practical CMG system.

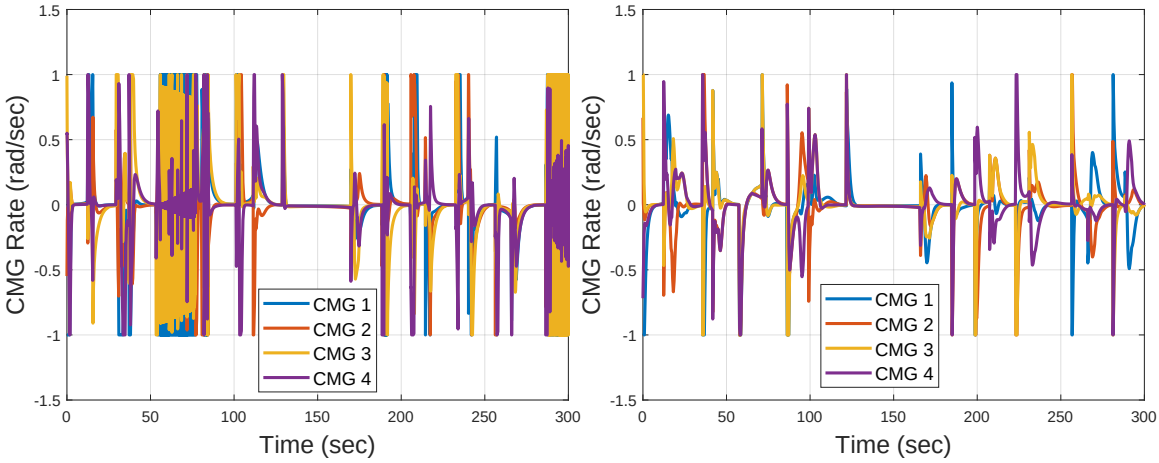
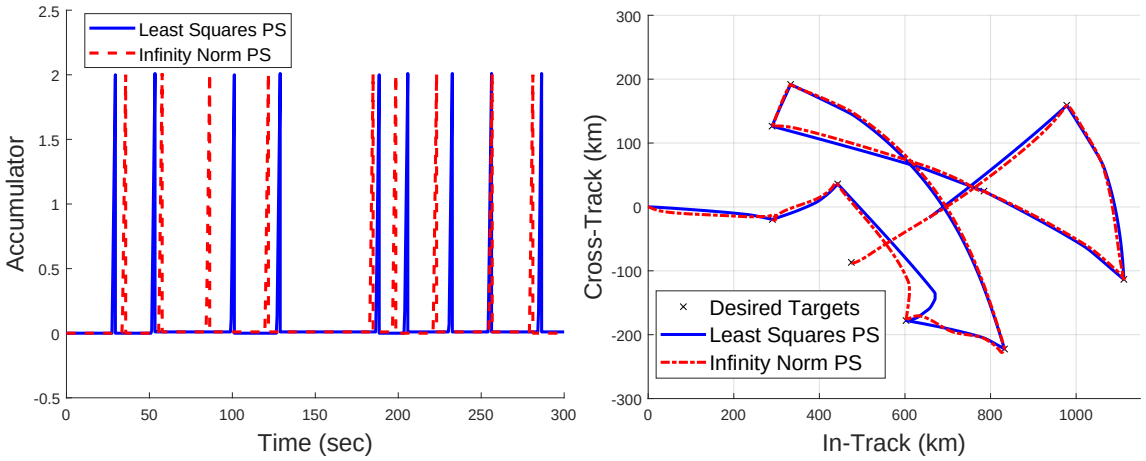


Figure 3.16. Requested Command Comparison: (a) Least Squares Commanded Rate; (b) Infinity Norm Commanded Rate.

The accumulator plot, which depicts the status of the image acquisition process, and the boresight trace of each allocation can be seen in Figure 3.17 to see which system performed better overall in image acquisition. Here it is observed that infinity norm allocation tends to outperform the least squares pseudoinverse by capturing all but three targets more quickly than the least squares scheme. In a world where time is money, capturing more images in a similar amount of time gives attitude control using infinity norm allocation an edge. It is interesting to note that the least squares allocation catches up to the infinity norm on the eighth target. The least squares' ability to catch up could be likely because the seventh target lies almost directly between the sixth and eight, which would allow the least squares method more time to build up torque to catch up. It could also be due to both torque solutions that exist on a common edge of both torque envelopes.



(a) (b)

Figure 3.17. CMG Simulation Comparison: (a) Accumulator; (b) Boresight Trace.

In Figure 3.17b shows that each allocation took slightly different paths compared to the reaction wheel case in Figure 3.11, where both solutions were right on top of each other. The path's differences could be due to the different dynamics employed by this recurrent neural network and due to the specific values for α and K chosen. This implementation of the recurrent neural network used a single artificial gain that was not not chosen by the DoE method to speed up convergence to a solution. The lack of a specifically developed gain could account for why the infinity norm allocation tends to wobble a little bit compared to the least squares allocation. Furthermore, the infinity norm allocation tended to overshoot a target and circle back while never losing lock. With the integrator gain in the control loop feedback not being fine-tuned to the CMG system on top of the error introduced from single artificial gain employed could be the cause of this wandering.

Further work to tune the problem formulation for singularity avoidance and the recurrent neural network and integrator gains should alleviate the issues illustrated here.

3.5 Summary

The infinity norm allocation to the reaction wheel attitude control problem provides for a larger torque envelope that allows a spacecraft to better use the available torque capability.

Alternatively, designers could use an infinity norm allocation scheme to select smaller devices to accomplish the same design goals, which then turns the problem into a optimal linear programming problem. Recurrent neural networks can be implemented as a analog electrical circuit to perform the calculations so as to avoid the iterative approach associated with a linear programming solver.

Since the target positions and sequence of acquisitions was randomly generated, the simulation results presented here may not fully illustrate the true differences between the infinity norm and least squares allocation schemes. However, when it comes to imaging missions, every second can make a difference. In light of this, using an infinity norm allocation could lead to the collection of a another target before the boresight passes out of reach or give a better resolution due to a closer proximity and smaller off nadir angle. This chapter has thus shown that a recurrent neural network can be used to solve an infinity norm control allocation problem for satellite attitude control and that the allocation can outperform the conventional least squares allocation.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Utilizing Infinity Norm Control Allocation with Optimized Spacecraft Control Trajectories

The previous chapter focused on applying recurrent neural networks in the control loop feedback during the simulation of an imaging spacecraft. This chapter explores how the choice of allocation scheme impacts the ability of a spacecraft to reproduce a time-optimal reorientation path. The optimal controls are developed and inserted into the control loop feedback simulation of Chapter 2 and then compare the results.

4.1 Minimum-Time Reorientation Problems

4.2 Problem Formulation

The trajectory optimization problem that will define this chapter's scope can be seen in equation (4.1). The minimum-time reorientation problem [22] cost functional is simply minimizing the final time. The dynamics are defined in equation (4.1) as $\dot{q}$, $\dot{\omega}$, and $\dot{h}_{RW}$. Simplification of $\dot{\omega}$ has been made in the form of the conservation of angular momentum by canceling out the inner terms of the cross product. This is consistent with a zero net-bias attitude control system. The maneuver to be performed is a 180 degree rotation about the $\hat{\mathbf{e}}^T = \begin{bmatrix} 1/\sqrt{3} & -1/\sqrt{3} & 1/\sqrt{3} \end{bmatrix}^T$ eigenaxis from a coincident Cartesian reference frame. Equation (4.1) illustrates the state-space dynamics, initial conditions, and the endpoint function.

$$
\begin{cases}
\text{Minimize:} & J[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f] = t_f \\
\text{Subject to:} & \dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \begin{bmatrix} \boldsymbol{\omega} \\ 0 \end{bmatrix} \\
& \dot{\boldsymbol{\omega}} = -I^{-1}(\boldsymbol{\omega} \times (I\boldsymbol{\omega} + Z\mathbf{h}_{RW}) + Z\dot{\mathbf{h}}_{RW}) = -I^{-1}Z\dot{\mathbf{h}}_{RW} \\
& \dot{\mathbf{h}}_{RW} = \boldsymbol{\tau}_{RW} \\
& t_0 = 0 \\
& \mathbf{q}(0) = [0, 0, 0, 1]^T \\
& \boldsymbol{\omega}(0) = [0, 0, 0]^T \\
& \mathbf{h}_{RW}(0) = [0, 0, 0, 0]^T \\
& \mathbf{e}(\mathbf{x}(t_f)) = [\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, 0, 0, 0, 0, 0, 0, 0, 0]^T \\
& |\bar{h}_{\boldsymbol{\omega}}| = \omega_1^2 + \omega_2^2 + \omega_3^2 \leq 3.97^\circ/\text{sec} \\
& |\boldsymbol{\tau}_{RW}|) \leq 0.6 \text{ Nm}
\end{cases}
\tag{4.1}
$$

where $\mathbf{e}(\mathbf{x}t_f)$ are the end point for all states, t_0 is the initial time, t_f is the final time, and all other variables are as defined in Chapter 3.

This chapter will focus on two different problem sets; both problems will execute the same maneuver, but the control will be defined differently in each. The first problem assume the reaction wheels can be controlled directly, meaning there will be four independent control variables, equation (4.2), hereafter refered to as $Problem\ A$. The other method is to take the three-dimensional torques acting on the spacecraft as the control and use the pseudoinverse to calculate the torques that should be applied to the wheels, hereafter referred to as $Problem\ B$. This problem formulation means the three controls are the three independent body torques as in equation (4.3).

$$
\boldsymbol{\tau_{RW}} = \mathbf{u}_{RW} \in \mathbb{R}^{4x1} \tag{4.2}
$$

$$
\boldsymbol{\tau_{RW}} = Z^{-1}\mathbf{u}_{3D} \text{ where } \mathbf{u}_{3D} \in \mathbb{R}^{3x1} \tag{4.3}
$$

The properties of the spacecraft to be simulated are the same as in Table 3.2.

4.2.1 Development of the Necessary Conditions

Pontryagin's principle [23] will be used to develop any relationships that will define the optimal trajectory. The first step in any problem is to develop the Hamiltonian, $\bar{H}$. The Hamiltonian for a minimum time reorientation of a spacecraft can be seen in equation (4.4).

$$\begin{aligned} \bar{H} &= F(\mathbf{x},\mathbf{u}) + \boldsymbol{\lambda}^T f(\mathbf{x},\mathbf{u}) + \boldsymbol{\mu}^T \bar{h} \\ &= \boldsymbol{\lambda}_{\mathbf{q}}^T (\tfrac{1}{2}\mathbf{q} \otimes \begin{bmatrix} \boldsymbol{\omega} \\ 0 \end{bmatrix}) + \boldsymbol{\lambda}_{\boldsymbol{\omega}}^T (-I^{-1} Z \boldsymbol{\tau}_{RW}) + \boldsymbol{\lambda}_{\mathbf{h}_{RW}}^T \boldsymbol{\tau}_{RW} + \boldsymbol{\mu}^T \begin{bmatrix} \bar{h}_{\boldsymbol{\omega}} \\ \bar{\mathbf{h}}_{RW} \end{bmatrix} \end{aligned} \tag{4.4}$$

where $F(\mathbf{x},\mathbf{u})$ are any running costs of the problem, $f(\mathbf{x},\mathbf{u})$ refers to the state equations, $\bar{h}$ are the state constraints as found in equation (4.1). Additionally $\boldsymbol{\lambda}$ refers the to costate pairs of the associated states and $\boldsymbol{\mu}$ are similar to the costates but are instead tied to the state constraints.

Developing the necessary conditions for optimality involves taking the Hamiltonian's partial derivative with respect to each state; this reveals how each of the costates should behave in time. MATLAB symbolic toolbox was used extensively to simplify the number of calculations performed since 11 states involve some significant algebra. In this case, MATLAB's symbolic toolbox reveals that the only costate that has easily verified behaviors are the reaction wheel costates, which should have a constant slope, which is true using either formulution of $Problem\ A$ or $Problem\ B$, as seen in equation (4.5).

$$\dot{\boldsymbol{\lambda}}_{\mathbf{h}_{RW}} = -\frac{\partial \bar{H}}{\partial \mathbf{h}_{RW}} = \mathbf{0} \tag{4.5}$$

To minimize the Hamiltonian, its partial derivative is taken with respect to each control and set equal to 0. The same is done with the Hamiltonian with respect to each of the controls in order to minimize the Hamiltonian and determine the optimal control trajectories. However, since the Hamiltonian is linear in the control, the optimal controls must be determined based on the behavior of the switching functions. Even though there are no closed forms of the solution, the relationships resulting from this condition must equal 0; this is called the stationarity condition.

To try and determine how the Hamiltonian will behave over time the partial of $\bar{H}$ with

respect to time is taken. The result is called the Hamiltonian evolution equation [23] and reveals any relationship that can be utilized to verify optimality after running it through an optimal control solver. Since there is no running cost and no states or path constraints are explicit functions of time, the Hamiltonian should remain a constant value. Equation (4.6) shows the Hamiltonian evolution equation and the result. $\mathcal{H}$ is the minimized Hamiltonian.

$$\frac{d\mathcal{H}}{dt} = \frac{\partial \bar{H}}{\partial t} = 0 \tag{4.6}$$

Next, in the application of Pontryagin's principle is to develop the Endpoint Lagrangian, $\bar{E}$, as seen in equation (4.7).

$$\begin{aligned} \bar{E} &= E(\mathbf{x}(t_f)) + \boldsymbol{\nu}^T e(\mathbf{x}(t_f)) \\ &= t_f + \boldsymbol{\nu}_{\mathbf{q}}^T e(\mathbf{q}(t_f)) + \boldsymbol{\nu}_{\boldsymbol{\omega}}^T e(\boldsymbol{\omega}(t_f)) + \boldsymbol{\nu}_{\mathbf{h}_{RW}}^T e(\mathbf{h}_{RW}(t_f)) \end{aligned} \tag{4.7}$$

where $E(\mathbf{x}(t_f))$ is the end point cost of the function and $\boldsymbol{\nu}$ are similar to the costate vectors but associated to only the defined end points assigned to each state.

Using the formula from equation (4.7) and MATLAB's symbolic toolbox, the last few verifiable relationships, which, if they exist, should appear with a few manipulations. The first to examine is the Hamiltonian's final value is obtained by taking the partial derivative of the negative of equation (4.7) with respect to t_f as seen in equation (4.8). Examining equation (4.8) with the knowledge that the Hamiltonian is a constant value as determined previously, the optimal solution will have a Hamiltonian of a constant negative one.

$$\bar{H}(t_f) = -\frac{\partial \bar{E}}{\partial t_f} = -1 \tag{4.8}$$

The last necessary conditions result from transversality analysis, which is developed for the case of missing boundary conditions. This analysis is done by taking the partial of the endpoint Lagrangian with respect to each of the final states; however, this is done to fill in gaps for missing boundary conditions. As mentioned previously, MATLAB's symbolic toolbox was used extensively in the application of Pontryagin's principal, and since the boundary conditions for all states have been defined, this was used to verify that no new

information was obtained in this step. These relationships were derived for individually for both $Problem\ A$ and $Problem\ B$ and found the same relationships apply to both.

The last optimality condition is related to the KKT conditions. Simply put, the KKT conditions dictates the value the path covectors can take along the optimal trajectory. The KKT condition is illustrated in equation (4.9). If those conditions are met, then it verified that the solution is an optimal solution to the minimum time reorientation problem. These results are also often referred to as the complementarity condition.

$$\begin{aligned} \boldsymbol{\mu} &\geq 0 \quad \text{if} \quad \bar{h} = \bar{h}^U \\ \boldsymbol{\mu} &= 0 \quad \text{if} \quad \bar{h}^L < \bar{h} < \bar{h}^U \\ \boldsymbol{\mu} &\leq 0 \quad \text{if} \quad \bar{h} = \bar{h}^L \end{aligned} \tag{4.9}$$

$Problem\ A$ and $Problem\ B$ were both shown to have the same optimality conditions.

4.2.2 Problem Solutions

With the optimal control analysis developed, there is only one more step to take before attempting to solve the trajectory optimization problems, scaling and balancing. Scaling the state-space equations must be done to facilitate well-conditioned computation. Scaling ensures that the states' variations are similar to the respective costates and vice-versa; without scaling, a state's motion can outweigh its costate counterpart and result in a sub-optimal solution or no solution at all. Scaling can be achieved by creating an individual scaling variable for each state and costate pair. An example of scaling for $\dot{q}_1$ can be seen in equation (4.10), where each variable on the right hand side with a $\tilde{}$ has been scaled according to equation (4.11). The method of scaling variables individually using equation (4.11), where x is the variable to be scaled and X is a constant scalar, allows for better tailoring of each state and costate pair. T is a scalar constant used to scale time in the derivative.

$$\begin{aligned} &\text{Unscaled:} \quad \dot{q}_1 = \tfrac{1}{2}(q_3\omega_2 - q_4\omega_3 + q_2\omega_1) \\ &\text{Scaled:} \quad \dot{\tilde{q}}_1 = \tfrac{T}{2Q_1}(\tilde{q}_3\tilde{\omega}_2(Q_3\Omega_2) - \tilde{q}_4\tilde{\omega}_4(Q_3\Omega_3) + \tilde{q}_2\tilde{\omega}_1(Q_2\Omega_1)) \end{aligned} \tag{4.10}$$

$$\tilde{x} = \frac{x}{X} \tag{4.11}$$

Figure 4.1 depicts the typical solution to the optimal control problem, equation (4.1), to illustrate the scaling between a state and costate pair for the variable q_1. An ideal scaling results in a one to one magnitude between a state and costate pair, as seen in Figure 4.1b. Whereas in Figure 4.1a, the state appears to have almost no motion compared to the costate. Through trial and error, the values in Table 4.1 were obtained as resulting in the best simulation output and state and costate scaling.

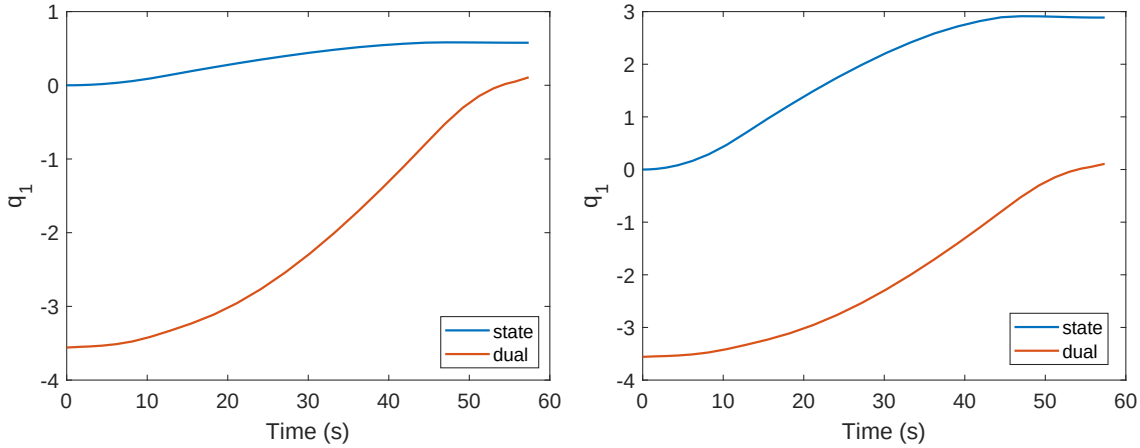


Figure 4.1. Effect of Scaling: (a) q_1 Unscaled; (b) q_1 Scaled.

Table 4.1. Scales used for Solving Minimum Time Reorientation Problem.

State	Scale
q_1	0.2
q_2	0.1
q_3	0.1
q_4	2
ω_1	0.01
ω_2	0.1
ω_3	0.1
$h_{RW,1}$	0.1
$h_{RW,2}$	0.1
$h_{RW,3}$	0.1
$h_{RW,4}$	0.1

The scaling obtained in Table 4.1 was developed for the *Problem A*. The same scaling was used for both cases to make direct comparisons between *Problem A* and *Problem B*. This scaling worked similarly for each problem as illustrated in Figure 4.2.

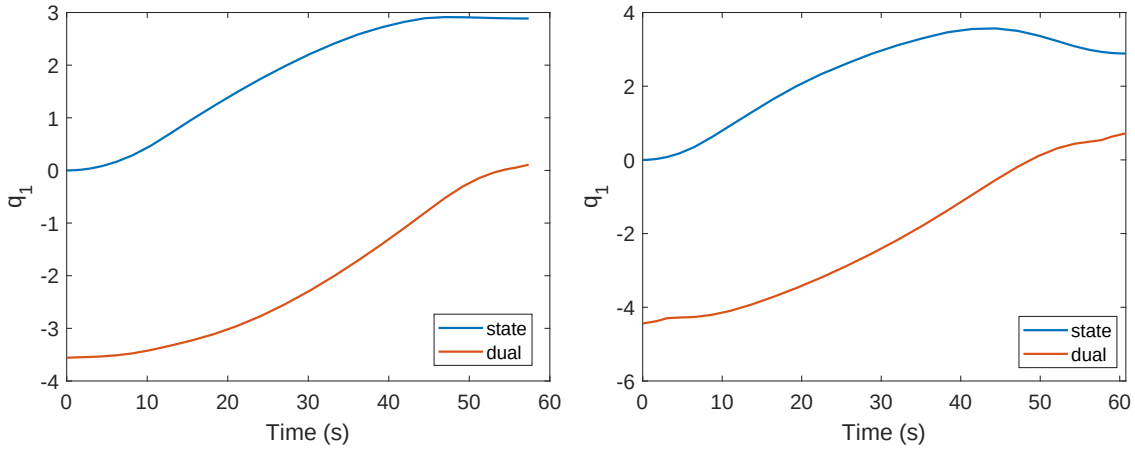


Figure 4.2. Effect of Pseudoinverse on Scaling: (a) q_1 *Problem A* Scaled; (b) q_1 *Problem B* Scaled.

Once the optimal controls were obtained, they were then used to propagate the plant dynamics function, which utilized MATLAB's ode45 integrator. Table 4.2 details all verification

of boundary conditions of the optimal solution and the plant dynamics simulation as well as the final minimum times that resulted from each control method.

Table 4.2. Results of Feasibility Test.

State	Desired End State	*Problem A* Control	*Problem A* Verification	*Problem B* Control	*Problem B* Verification
q_1	0.557	0.557	0.574	0.577	0.579
q_2	-0.557	-0.557	-0.585	-0.577	-0.572
q_3	0.557	0.577	0.573	0.577	0.581
q_4	0	-2.9e-14	0.012	-6.3e-15	-8.1e-4
ω_1	0	1.3e-16	8.1e-5	-3.1e-20	1.8e-4
ω_2	0	4.2e-14	1.3e-4	-1.8e-16	3.7e-4
ω_3	0	7.3e-15	7.8e-5	1.2e-16	1.1e-4
$h_{RW,1}$	0	-2.6e-15	0.050	2.5e-14	0.014
$h_{RW,2}$	0	-1.2e-20	-0.078	-3.1e-17	0.066
$h_{RW,3}$	0	0	0.037	2.0e16	-0.033
$h_{RW,4}$	0	0	-0.035	2.6e-14	0.047
t_f	N/A	57.3733	N/A	60.7673	N/A

The majority of the results show the difference of the optimal solution vs propagated solution are numerically zero. Neither the *Problem A* or the *Problem B* control method seem to result in a decidedly better end boundary conditions, but *Problem A*'s solution did result in a 3 second faster maneuver, as expected.

Figure 4.3 verifies the equations used to simulate the plant dynamics. The momentum balance, $|I\omega + Zh_{RW}|$, was taken throughout the entire maneuver duration to ensure the conservation of angular momentum. Both scenarios perform near numerical zero, validating that the dynamic simulation is accurate.

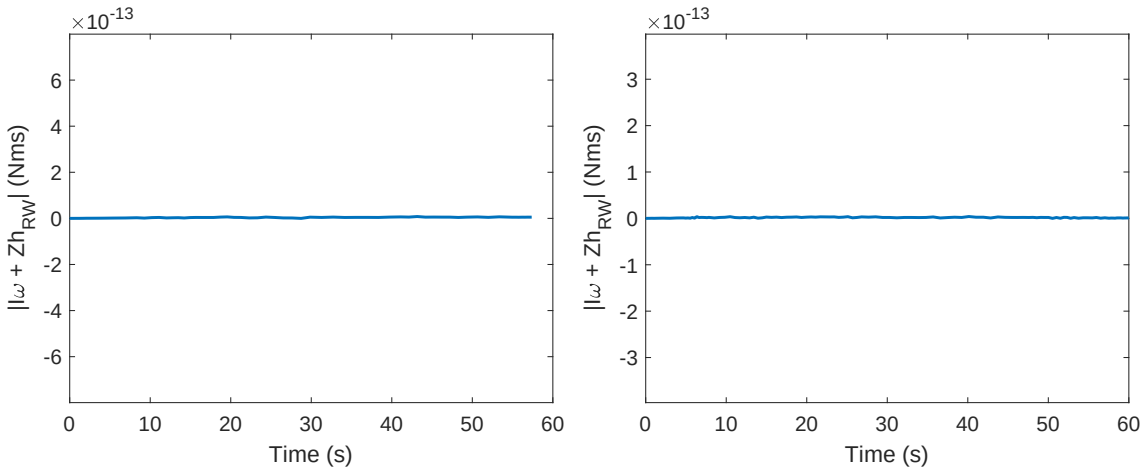


Figure 4.3. Variation in Total Angular Momentum: (a) *Problem A* Momentum Conservation; (b) *Problem B* Momentum Conservation.

Figure 4.4 also shows that the norm of the quaternions falls within an acceptable range of 1 with some numerical error.

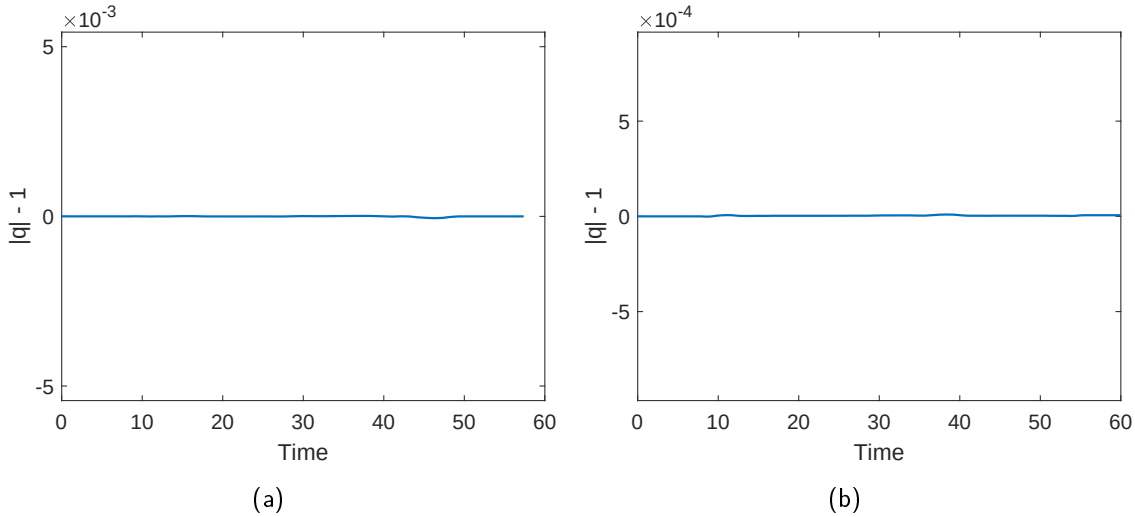
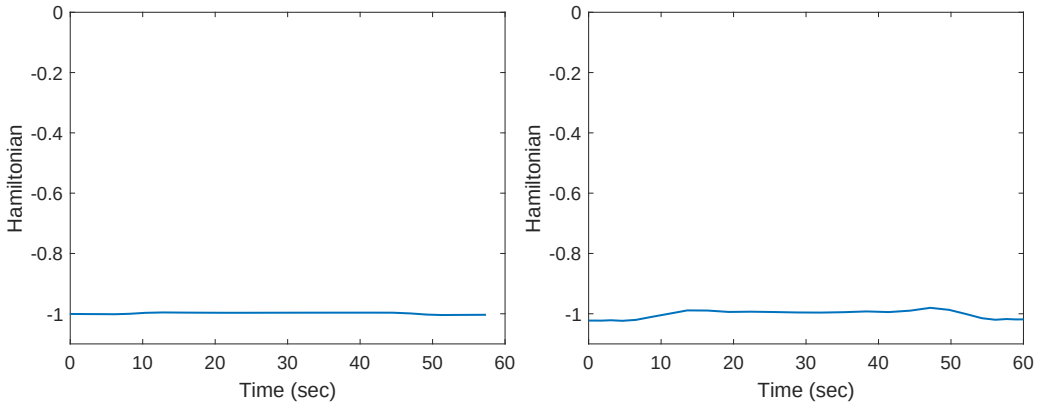


Figure 4.4. Check of Quaternion Norm: (a) $|\mathbf{q}| - 1$ *Problem A*; (b) $|\mathbf{q}| - 1$ *Problem B*.

When comparing the Hamiltonians of Figure 4.5, the data shows that both scenarios are

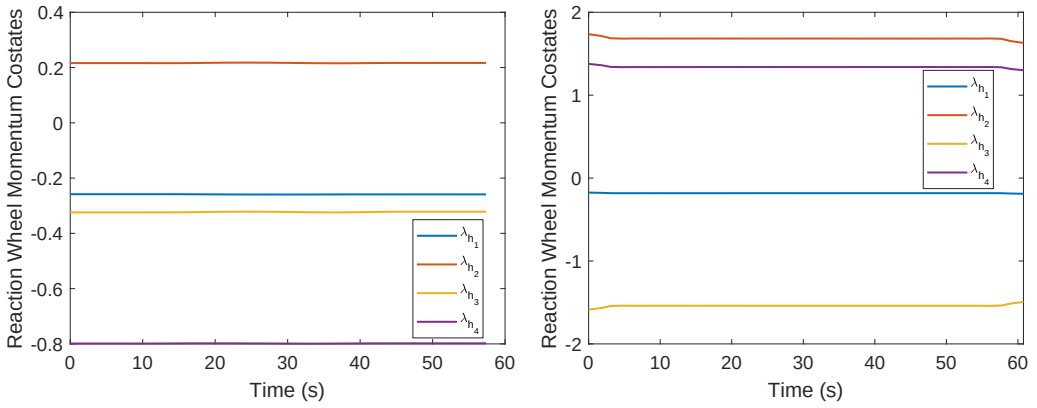
centered roughly around negative 1 with $Problem\ A$ having a smoother solution. Overall, this confirms the solution's extremal nature of the solutions.



(a) (b)

Figure 4.5. Hamiltonian of System: (a) $\bar{H}$ Hamiltonion $Problem\ A$; (b) $\bar{H}$ Hamiltonian $Problem\ B$.

Figure 4.6 shows the scaled angular momentum costates for both scenarios, of which all are relatively flat: confirming the optimality of the solution. There are some minor variations to each of them, but this could be attributed to the simulation solver's numerical nature.



(a) (b)

Figure 4.6. Reaction Wheel Costates of System: (a) $\lambda_{h_{RW}}$ $Problem\ A$; (b) $\lambda_{h_{RW}}$ $Problem\ B$.

The pseudoinverse was applied to the three-dimensional control torques to determine the

individual wheel torques in Figure 4.7b to facilitate comparison. Using the pseudoinverse also allows for the easy verification of the path constraints, holding to the plus or minus 0.6 Nm torque limit of the given spacecraft's reaction wheels. In *Problem A*, the switch between the commanded torques happens much faster than what was observed for *Problem B*; however, *Problem B* seems to cause the controls to converge to near-zero values in the center more smoothly than seen in *Problem A*. *Problem A* torques could be smoother by solving for addition nodes along the path, but it was kept to 30 to speed up the solution process.

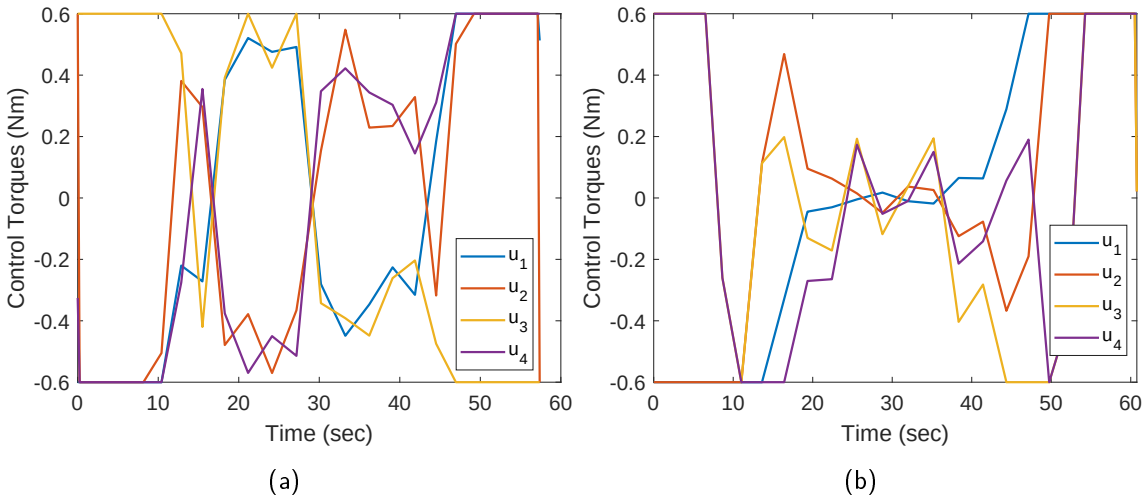


Figure 4.7. Controls of System: (a) *Problem A* ($\mathbf{u}_{RW}$); (b) *Problem B* ($Z^{-1}\mathbf{u}_{3D}$).

Figure 4.8 depicts the state constraint applied to the overall spacecraft angular velocity, and many of the same comparisons can be made about this constraint as was made for controls in Figure 4.7. While both methods adhere to the state constraint, *Problem A* has almost a constant slope until angular velocity hits the constraint and *Problem B* has the angular velocity bend as it approaches the constraint. This result suggests the *Problem B* tends to limit the performance.

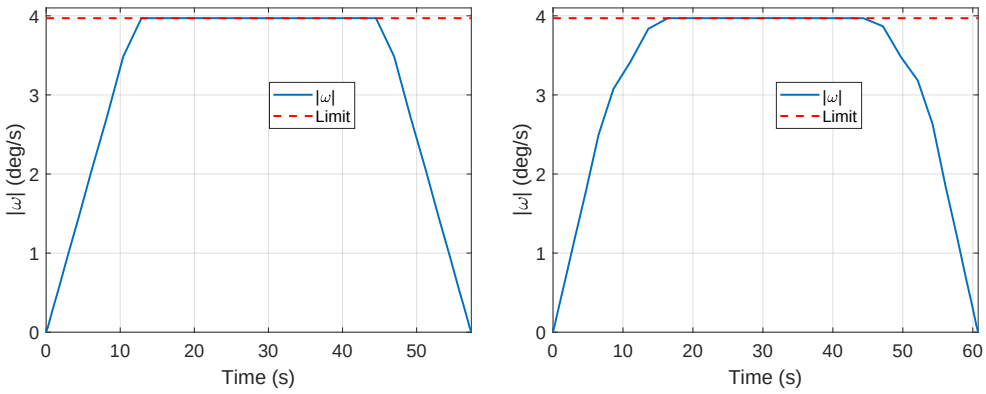


Figure 4.8. $|\boldsymbol{\omega}|$ Constraint Verification: (a) $|\boldsymbol{\omega}|$ Constraint *Problem A* ($\mathbf{u}_{RW}$); (b) $|\boldsymbol{\omega}|$ Constraint *Problem B* ($Z^{-1}\mathbf{u}_{3D}$).

Figure 4.9 depicts the KKT conditions that are expected to result from equation (4.9). Additionally, Figure 4.9 illustrates that the optimality conditions are met with *Problem A* showing more pronounced results. The other controls are not shown as the results are similar. The reaction wheel control torques produced from *Problem B* are shown in Figure 4.9b.

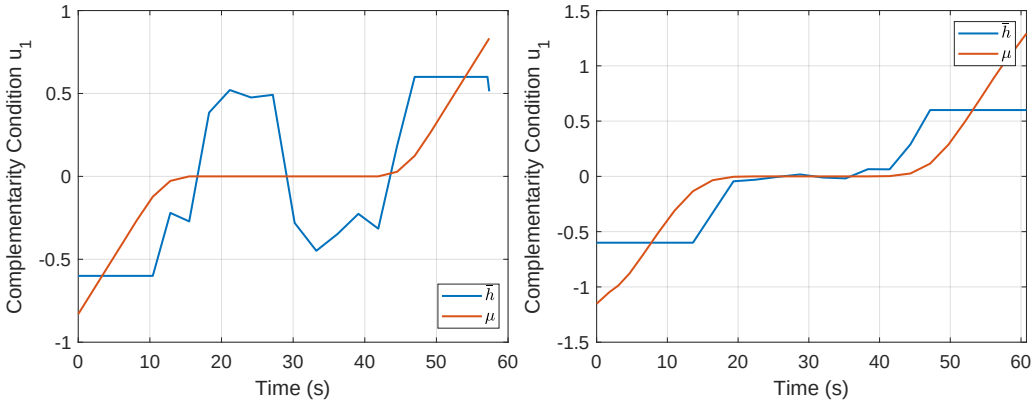


Figure 4.9. KKT Verification: (a) KKT Verification *Problem A*; (b) KKT Verification *Problem B*.

Figure 4.10 shows that the stationarity check passes for both problems; extra figures were

left off for all the other controls because they all showed similar results.

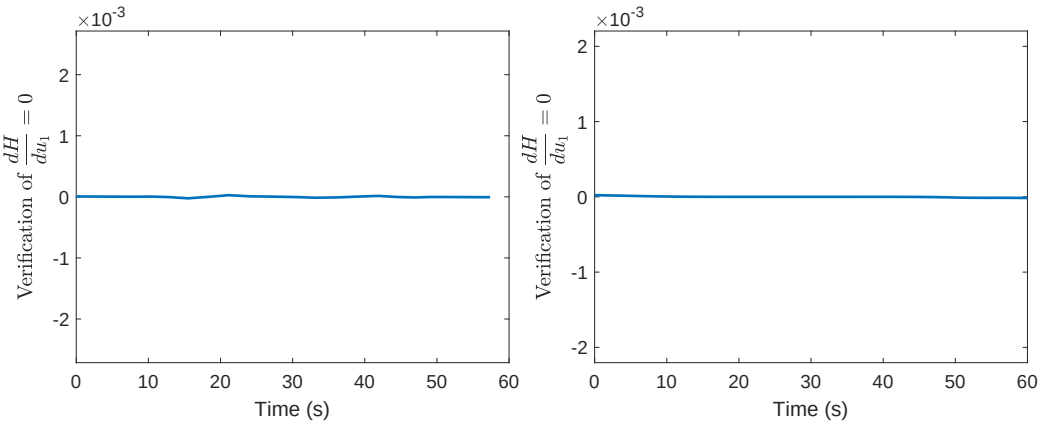
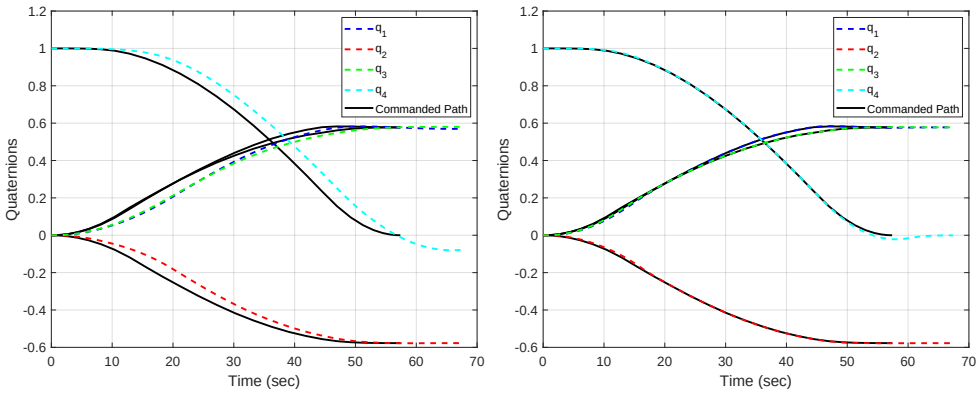


Figure 4.10. Stationarity Verification: (a) Stationarity Verification *Problem A*; (b) Stationarity Verification *Problem B*.

4.3 Implementing Optimal Trajectories under Feedback Control

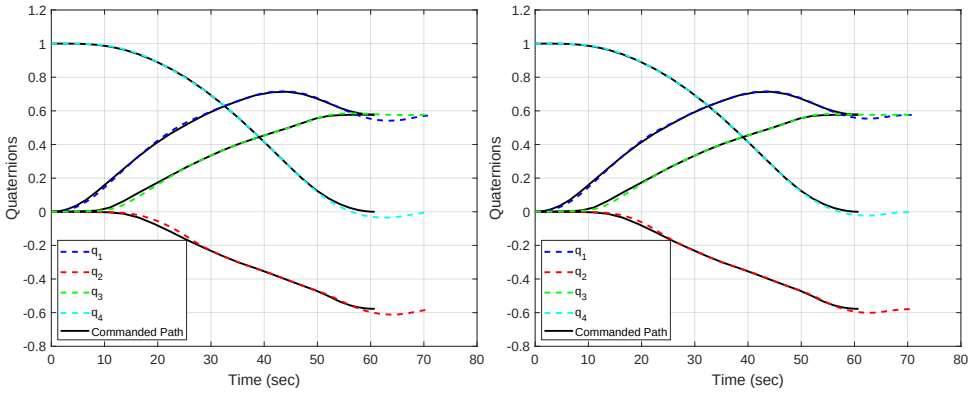
To ascertain how the optimal trajectories for both the cases could be reproduced using a standard attitude control loop, they were introduced into the spacecraft simulation of chapter 3. Figure 4.11 shows the spacecraft attitude response as well as the commanded optimal path from *Problem A*.



(a) (b)

Figure 4.11. Spacecraft Orientation Using *Problem A* Reaction Wheel Control: (a) Orientation Using Least Squares; (b) Orientation Using Infinity Norm.

It is immediately evident that using the least squares pseudoinverse causes a lag in Figure 4.11a while the infinity norm pseudoinverse in Figure 4.11b can better reproduce the path with only a little bit of overshoot at the end. The infinity norm outperforms again due to the larger torque envelope that it has available to it. Next, in Figure 4.12, spacecraft attitude response as well as the commanded optimal path is shown for the optimal control developed *Problem B*.



(a) (b)

Figure 4.12. Spacecraft Orientation Using *Problem B* Reaction Wheel Control: (a) Orientation Using Least Squares; (b) Orientation Using Infinity Norm.

In this particular instance, with the control developed in *Problem B* having been made specifically for a least squares pseudoinverse, both allocation methods have no issue adhering to the commanded profile that is being fed to the control loop. The only difference that can be seen is that the infinity norm seems to have slightly less overshoot at the end as compared to the least squares allocation. The difference between Figures 4.11 and 4.12 illustrates that if optimal control is being developed for a system, it must take into account the specific control allocation that is implemented on the spacecraft.

4.4 Summary

In optimal control development for a system, the control to be implemented must be modeled in the state-space equations. As seen in the first portion of this chapter, the pseudoinverse utilized in *Problem B* resulted in a slightly slower maneuver time for the spacecraft. The slower maneuver time is because less torque is available for the pseudoinverse allocation. If a designer did not account for how the control is implemented, the spacecraft could miss or waste time reacquiring an image. While the difference in optimal paths between the problems only accounted for a couple of seconds, it was observed in Figure 4.11 that the adherence to the desired profile can be lost when implemented in a control loop resulting in even more time lost. However, even without accounting for how this control is being developed, if the infinity norm allocation could be utilized in the control loop, there would be no need to distinguish between commanded paths and what the system can realize because the infinity norm can access the full torque envelope.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusion and Future Work

In control loop feedback for spacecraft attitude control, the torque commands in the body frame must be mapped to each of the control actuators used by the system. The typical least squares solution is easily calculated, but the solution is conservative and does not make use of the full torque envelope. The infinity norm pseudoinverse has the capability to utilize all of the torque envelope of a spacecraft's momentum exchange devices. The larger torque envelope provided by the infinity norm can be used to outperform a similar system that employs the least squares pseudoinverse. The downside to the infinity norm solution is that it is not trivial to calculate and can require time for a solution to converge. This convergence issue can be countered by implementing a recurrent network in an analog circuit such as a FPAA or VLSI inline with the control loop.

It is essential to ensure that how the control is fed back into the system is accounted for in optimal control development. However, since the infinity norm has access to the entire torque envelope, it can keep up the requested torques if the controls were not developed to account for that system requiring a pseudoinverse to get the optimal controls.

This thesis only took a small look at the application of the recurrent neural network with CMG devices. The recurrent neural network seems to have an inherent advantage over the least squares pseudoinverse regarding the system's stability as it approaches singularity. The recurrent neural network could be optimized for a spacecraft that utilizes CMGs. It could further be improved by including additional constraints to prevent the system from ever reaching singular configuration.

The fabrication of the recurrent neural network in hardware could only be looked at through simulation and modeling in software. Additional work would start by fabricating a recurrent neural network in a circuit. The circuit would then have to be tested on working momentum exchange devices for additional comparisons to the typical least squares pseudoinverse approach utilized by industry.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX: Determining the Error Quaternion from Two Vectors

Equation (A.1) is the definition of a quaternion.

$$q = \begin{bmatrix} \hat{e}_1 \sin\frac{\theta}{2} \\ \hat{e}_2 \sin\frac{\theta}{2} \\ \hat{e}_3 \sin\frac{\theta}{2} \\ \cos\frac{\theta}{2} \end{bmatrix} \tag{A.1}$$

where $\hat{\mathbf{e}}$ is the unit vector describing the axis of rotation to go from one Cartesian orientation to another and θ describes the total rotation to perform. Equation (A.2) describes the change in the quaternion to go from vector 1, $\mathbf{v}_1$, to vector 2, $\mathbf{v}_2$.

$$\Delta q = \frac{\begin{bmatrix} \mathbf{v}_1\mathbf{v}_2 \\ 1 + \mathbf{v}_1 \cdot \mathbf{v}_2 \end{bmatrix}}{\sqrt{(\mathbf{v}_2 \times \mathbf{v}_1)^2 + (1 + \mathbf{v}_1 \cdot \mathbf{v}_2)^2}} = \frac{|\mathbf{v}_1||\mathbf{v}_2| \begin{bmatrix} \hat{e}_1 \sin\theta \\ \hat{e}_2 \sin\theta \\ \hat{e}_3 \sin\theta \\ \frac{1}{|\mathbf{v}_1||\mathbf{v}_2|} + \cos\theta \end{bmatrix}}{\sqrt{(\hat{e}_1 \sin\theta + \hat{e}_2 \sin\theta + \hat{e}_3 \sin\theta)^2 + (\frac{1}{|\mathbf{v}_1||\mathbf{v}_2|} + \cos\theta)^2}} \tag{A.2}$$

$\mathbf{v}_1$ and $\mathbf{v}_2$ are defined to be unit vectors so equation A.2 simplifies to the following in

equation (A.3).

$$\Delta q = \frac{\begin{bmatrix} \hat{e}_1 \sin\theta \\ \hat{e}_2 \sin\theta \\ \hat{e}_3 \sin\theta \\ 1+\cos\theta \end{bmatrix}}{\sqrt{(\hat{e}_1 \sin\theta + \hat{e}_2 \sin\theta + \hat{e}_3 \sin\theta)^2 + (1+\cos\theta)^2}} = \frac{\begin{bmatrix} \hat{e}_1 \sin\theta \\ \hat{e}_2 \sin\theta \\ \hat{e}_3 \sin\theta \\ 1+\cos\theta \end{bmatrix}}{\sqrt{((\hat{e}_1+\hat{e}_2+\hat{e}_3) \sin\theta)^2 + (1+\cos\theta)^2}} = \frac{\begin{bmatrix} \hat{e}_1 \sin\theta \\ \hat{e}_2 \sin\theta \\ \hat{e}_3 \sin\theta \\ 1+\cos\theta \end{bmatrix}}{\sqrt{(\hat{e}_1+\hat{e}_2+\hat{e}_3)^2 \sin^2\theta + 1 + 2\cos\theta + \cos^2\theta}} \tag{A.3}$$

Since $\hat{\mathbf{e}}$ is defined to be a unit vector it simplifies to 1, and $\sin^2\theta + cos^2\theta$ is also equal to 1, equation (A.3) simplifies further into equation (A.4).

$$\Delta q = \frac{\begin{bmatrix} \hat{e}_1 \sin\theta \\ \hat{e}_2 \sin\theta \\ \hat{e}_3 \sin\theta \\ 1+\cos\theta \end{bmatrix}}{\sqrt{2+2\cos\theta}} \tag{A.4}$$

To show that equations (A.2) through (A.4) do not violate the defined quaternion relationships, Δq_1 will be rewritten using algebra and trigonometric identities as seen in equation (A.5).

$$\begin{aligned} \Delta q_1 = \frac{\hat{e}_1 \sin\theta}{\sqrt{2+2\cos\theta}} = \hat{e}_1 \sqrt{\frac{1-\cos^2\theta}{2+2\cos\theta}} = \hat{e}_1 \sqrt{\frac{(1-\cos\theta)(1+\cos\theta)}{2(1+\cos\theta)}} = \hat{e}_1 \sqrt{\frac{1-\cos\theta}{2}} \\ = \hat{e}_1 \sin\frac{\theta}{2} = q_1 \end{aligned} \tag{A.5}$$

This relationship can then be applied in the exact same format to Δq_2 and Δq_3. Δq_4 can also be rewritten using algebra and trigonometric identities as seen in equation (A.6).

$$\Delta q_4 = \frac{1+\cos\theta}{\sqrt{2+2\cos\theta}} = \frac{1+\cos\theta}{\sqrt{2+2\cos\theta}} = \sqrt{\frac{1+\cos\theta}{2}} = \cos\frac{\theta}{2} = q_4 \tag{A.6}$$

List of References

[1] F. L. Markley, R. G. Reynolds, F. X. Liu, and K. L. Lebsock, "Maximum torque and momentum envelopes for reaction wheel arrays," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 5, pp. 1606–1614, 2010.

[2] B. Wie, *Space Vehicle Dynamics and Control*, 2nd ed. Reston, VA, USA: American Institute of Aeronautics and Astronautics, 2008.

[3] M. Romano, "Linear control of spacecraft attitude," class notes for Spacecraft Attitude, Determination, and Control, Dept. of Mechanical and Aerospace Engineering, Naval Postgraduate School, Monterey, CA, USA, fall 2020.

[4] M. Karpenko, J. T. King, C. J. Dennehy, and I. Michael Ross, "Agility analysis of the james webb space telescope," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 4, pp. 810–821, 2018.

[5] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, 3rd ed. New York, NY, USA: Springer, 2008.

[6] C. C. Aggarwal, *Neural Networks and Deep Learning*. Cham, Switzerland: Springer, 2018.

[7] W. S. Tang and J. Wang, "A recurrent neural network for minimum infinity-norm kinematic control of redundant manipulators with an improved problem formulation and reduced architecture complexity," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 31, no. 1, pp. 98–105, 2001.

[8] Y. Zhang, J. Wang, and Y. Xu, "A dual neural network for bi-criteria kinematic control of redundant manipulators," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 6, pp. 923–931, 2002.

[9] J. Lee, "A structured algorithm for minimum l∞-norm solutions and its application to a robot velocity workspace analysis," *Robotica*, vol. 19, no. 3, pp. 343–352, 2001.

[10] Z. Mao and T. C. Hsia, "Obstacle avoidance inverse kinematics solution of redundant robots by neural networks," *Robotica*, vol. 15, no. 1, pp. 3–10, 1997.

[11] X. Hu and B. Zhang, "A new recurrent neural network for solving convex quadratic programming problems with an application to the k-winners-take-all problem," *IEEE Transactions on Neural Networks*, vol. 20, no. 4, pp. 654–664, 2009.

[12] T. H. Lee and C. J. Harris, *Adaptive Neural Network Control of Robotic Manipulators*. Singapore: World Scientific, 1998, vol. 19.

[13] H. Ding and S. K. Tso, "A fully neural-network-based planning scheme for torque minimization of redundant manipulators," *IEEE Transactions on Industrial Electronics*, vol. 46, no. 1, pp. 199–206, 1999.

[14] H. Ding and J. Wang, "Recurrent neural networks for minimum infinity-norm kinematic control of redundant manipulators," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 29, no. 3, pp. 269–276, 1999.

[15] Y. Xia, "A new neural network for solving linear and quadratic programming problems," *IEEE transactions on neural networks*, vol. 7, no. 6, pp. 1544–1548, 1996.

[16] A. S. Deo and I. D. Walker, "Minimum effort inverse kinematics for redundant manipulators," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 5, pp. 767–775, 1997.

[17] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. London, England: Springer, 2010.

[18] R. Penrose, "A generalized inverse for matrices," in *Mathematical Proceedings of the Cambridge Philosophical Society*, no. 3. Cambridge University Press, 1955, vol. 51, pp. 406–413.

[19] A. Giunta, S. Wojtkiewicz, and M. Eldred, "Overview of modern design of experiments methods for computational simulations," in *41st Aerospace Sciences Meeting and Exhibit*, 2003, p. 649.

[20] B. Diehl and M. Karpenko, "A recurrent network for infinity norm control allocation in spacecraft attitude control." Lake Tahoe, CA, USA: 2020 AAS/AIAA Astrodynamics Specialist Conference, August 2020, paper AAS 20-713.

[21] P. C. Calhoun and J. C. Garrick, "Observing mode attitude controller for the lunar reconnaissance orbiter." Annapolis, MD, USA: 20th International Symposium on Space Flight Dynamics, 2007.

[22] M. Karpenko, S. Bhatt, N. Bedrossian, and I. M. Ross, "Flight implementation of shortest-time maneuvers for imaging satellites," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 4, pp. 1069–1079, 2014.

[23] I. M. Ross, *A Primer on Pontryagin's Principle in Optimal Control*, 2nd ed. Carmel, CA, USA: Collegiate Publishers, 2015.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California