Mechatronics Design Process with Energy Optimization for Industrial Machines

Aderiano Medeiros da Silva

Marquette University

Recommended Citation
https://epublications.marquette.edu/dissertations_mu/518
The need for designing industrial machines with higher energy efficiency, reliability, flexibility, and accuracy has increased to satisfy market demand for higher productivity at reduced costs in a sustainable manner. As machines become more complex, model-based design is essential to overcome the challenges in mechatronic system design. However, a well-designed mechanical system with a well-designed and tuned control system are not sufficient for machines to operate at high-performance conditions; this also heavily depends on trajectory planning and the appropriate selection of the motors controlling the axes of the machine. In this work, a model-based design approach to properly select motors for single-axes or multi-axes coordinated systems was proposed. Additionally, a trajectory planning approach was also proposed to improve performance of industrial machines. The proposed motor selection process and trajectory planning approach were demonstrated via modeling, simulation, and experimental validation for three systems: two-inertia system, planar robot, and self-balancing transporter.

Over 25% of the electric energy delivered in the U.S. in 2013 was used in the industrial sector according to the U.S. Energy Information Administration, with an estimated efficiency of 80% according to the Lawrence Livermore National Laboratory. This entails major responsibility by the industry to utilize energy efficiently and promote sustainable energy usage. To help improve the energy efficiency in the industrial sector, a novel method to optimize the energy of single-axis and multi-axis coordinated systems of industrial machines was developed. Based on trajectory boundaries and the kinetic model of the mechanism and motors, this proposed energy optimization method performs iterations to recalculate the shape of the motion profile for each motor of the system being optimized until it converges to a motion profile with optimal energy cost and within these boundaries. This method was validated by comparing the energy consumption of those three systems while commanded by the optimized motion profile and then by motion profiles typically used in industrial applications. The energy saved was between 5% and 10%. The implementation cost of this method in industrial systems resides in machine-code changes; no physical changes are needed.
ACKNOWLEDGEMENT

Aderiano M. da Silva, B.S., M.S.

First of all, I would like to express my sincere gratitude for all the support, encouragement, and guidance I received from my advisor Dr. Kevin C. Craig. With his open-mind and critical thinking, he provided me with many opportunities that highly contributed to my development and positively impacted my career. I am, therefore, highly grateful to have had the opportunity to know and work with Dr. Craig.

I would also like to extend my great appreciation to my committee members Dr. Mark L. Nagurka, Dr. Philip A. Voglewede, Dr. Anthony Bowman, and Dr. Ronald H. Brown of Marquette University for their valuable time, support, and suggestions.

I am also deeply thankful to Dr. Ronald H. Brown, Dr. Nabeel A. O. Demerdash, Dr. Philip A. Voglewede, and Dr. Kevin C. Craig for the teaching and expertise in subjects that were fundamentally important to me during the development of my work in this dissertation.

I am also very grateful to my managers and colleagues at Rockwell Automation for the support and encouragement during Ph.D. studies. Special thanks to my managers and former-managers Kurt Mathson, Michael Bayer, Dr. Robert Miklosovic, Thomas Upham, and Michael Knox for the support over the years.

I would also like to thank the dedication of professors and staff from the Department of Mechanical Engineering at Marquette University, especially Dr. James A. Rice and Annette Wolak.
I wish to acknowledge the financial support provided by Dr. Kevin Craig’s funding and also by my employer Rockwell Automation.

Most importantly, I would like to express all my appreciation to my lovely wife Marcia for the support and encouragement over the years, and also to my daughter Nicole for her contagious smile and to my son Vitor for his unique charisma that were my daily motivation to accomplish such an important step in my life. I also want to thanks to my mother Maria and my father Jorge for the unconditional love and support.

I would have many more people to thank, but it would not be possible to name all. Thank you all!

Aderiano da Silva

May 2015
TABLE OF CONTENTS

ACKNOWLEDGMENTS ........................................................................................................ ii
LIST OF TABLES .................................................................................................................. iii
LIST OF FIGURES ................................................................................................................ iv

CHAPTER 1 INTRODUCTION .................................................................................................. 1
  1.1 DEFINITION OF THE PROBLEM .............................................................................. 1
  1.2 LITERATURE REVIEW ON MECHATRONIC DESIGN PROCESSES ..................... 2
  1.3 LITERATURE REVIEW ON ENERGY UTILIZATION IN THE INDUSTRY .............. 6
  1.4 CONTRIBUTIONS ...................................................................................................... 9
  1.5 DISSERTATION ORGANIZATION ......................................................................... 14

CHAPTER 2 MOTION PROFILES .......................................................................................... 17
  2.1 CHARACTERISTICS OF MOTION PROFILES .................................................... 20
  2.2 COMPUTATION OF CAM PROFILES ....................................................................... 22
  2.3 REAL-TIME CONTROL ............................................................................................ 25
  2.4 ASYMMETRIC MOTION PROFILES .......................................................................... 26
  2.5 MOTION PROFILES WITH CONSTANT SPEED SEGMENT .................................. 27
  2.6 TYPES OF MOTION PROFILES .............................................................................. 28
    2.6.1 Trapezoidal ........................................................................................................ 29
    2.6.2 Cubic ................................................................................................................... 32
    2.6.3 Simple Harmonic Motion (SHM) ....................................................................... 34
    2.6.4 Cycloidal ............................................................................................................ 35
    2.6.5 ModSine .............................................................................................................. 36
2.6.6 5th-Order Polynomial ................................................................. 39
2.6.7 7th-Order Polynomial ................................................................. 40
2.6.8 9th-Order Polynomial ................................................................. 42
2.7 COMPARISON OF MOTION PROFILES ...................................... 43
2.8 MOTION PROFILES WITH SLOPE ............................................. 45
2.9 MULTIPLE PROFILES ................................................................. 47
2.10 SUMMARY .................................................................................. 49

CHAPTER 3  MECHATRONIC DESIGN PROCESS .............................. 50

3.1 TWO-INERTIA SYSTEM ............................................................... 55
  3.1.1 System Requirements ............................................................... 55
  3.1.2 End-point Trajectory ............................................................... 56
  3.1.3 Trajectory Planning ................................................................. 57
  3.1.4 Inverse Kinematics ................................................................. 59
  3.1.5 Kinetic and Inverse Kinetic Model .......................................... 59
  3.1.6 Open-Loop Simulation ........................................................... 62
  3.1.7 Torque vs. Speed Requirement .............................................. 63
  3.1.8 Identifying Candidate Motors .............................................. 66
  3.1.9 Augmenting the System ......................................................... 69
  3.1.10 Control System Design ......................................................... 70
  3.1.11 Tuning .................................................................................. 71
  3.1.12 Closed-Loop Simulation ....................................................... 72
  3.1.13 Final Torque vs. Speed Requirements ................................... 73

3.2 CARTESIAN TWO-AXIS PLANAR ROBOT - H-BOT ................. 75
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.1</td>
<td>System Requirements</td>
<td>76</td>
</tr>
<tr>
<td>3.2.2</td>
<td>End-point Trajectory</td>
<td>76</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Trajectory Planning</td>
<td>76</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Inverse Kinematics</td>
<td>82</td>
</tr>
<tr>
<td>3.2.5</td>
<td>Kinetic and Inverse Kinetic Model</td>
<td>83</td>
</tr>
<tr>
<td>3.2.6</td>
<td>Open-Loop Simulation</td>
<td>85</td>
</tr>
<tr>
<td>3.2.7</td>
<td>Torque vs. Speed Requirement</td>
<td>86</td>
</tr>
<tr>
<td>3.2.8</td>
<td>Identifying Candidate Motors</td>
<td>88</td>
</tr>
<tr>
<td>3.2.9</td>
<td>Augmenting the System</td>
<td>90</td>
</tr>
<tr>
<td>3.2.10</td>
<td>Control System Design</td>
<td>91</td>
</tr>
<tr>
<td>3.2.11</td>
<td>Tuning</td>
<td>91</td>
</tr>
<tr>
<td>3.2.12</td>
<td>Closed-Loop Simulation</td>
<td>91</td>
</tr>
<tr>
<td>3.2.13</td>
<td>Final Torque vs. Speed Requirements</td>
<td>92</td>
</tr>
<tr>
<td>3.2.14</td>
<td>Trajectory Planning Validation</td>
<td>93</td>
</tr>
<tr>
<td>3.3</td>
<td>SELF-BALANCING TRANSPORTER DESIGN</td>
<td>95</td>
</tr>
<tr>
<td>3.3.1</td>
<td>System Requirements</td>
<td>96</td>
</tr>
<tr>
<td>3.3.2</td>
<td>End-point Trajectory</td>
<td>97</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Trajectory Planning</td>
<td>97</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Inverse Kinematics</td>
<td>98</td>
</tr>
<tr>
<td>3.3.5</td>
<td>Kinetic and Inverse Kinetic Model</td>
<td>98</td>
</tr>
<tr>
<td>3.3.6</td>
<td>Open-Loop Simulation</td>
<td>103</td>
</tr>
<tr>
<td>3.3.7</td>
<td>Torque vs. Speed Requirements</td>
<td>104</td>
</tr>
<tr>
<td>3.3.8</td>
<td>Identifying Candidate Motors</td>
<td>106</td>
</tr>
</tbody>
</table>
Chapter 3

3.3.9 Augmenting the System ................................................................. 108
3.3.10 Control System Design ............................................................... 108
3.3.11 Tuning ...................................................................................... 112
3.3.12 Closed-Loop Simulation ............................................................. 113
3.3.13 Final Torque vs. Speed Requirements ......................................... 114

3.4 Summary .................................................................................... 115

Chapter 4

Implementation of Physical Systems .............................................. 117

4.1 Implementation of the Two-Inertia System ................................. 118

4.1.1 Controller ................................................................................ 118
4.1.2 Encoder ................................................................................... 119
4.1.3 Power Control ......................................................................... 119
4.1.4 Control System and Tuning ...................................................... 119
4.1.5 Electrical Design and Final System .......................................... 120
4.1.6 Parameter Identification .......................................................... 121

4.2 Implementation of an H-Bot ........................................................ 126

4.2.1 Processor ................................................................................ 126
4.2.2 Encoder ................................................................................... 127
4.2.3 Power Structure Design .............................................................. 131
4.2.4 Control System and tuning ......................................................... 134
4.2.5 Electrical Circuit Design ............................................................ 136
4.2.6 Final System ............................................................................ 137
4.2.7 Parameter Identification .......................................................... 138

4.3 Implementation of a Self-Balancing Transporter ......................... 141
4.3.1 Processor .......................................................... 141
4.3.2 Sensors ............................................................. 141
4.3.3 Power Structure Design ........................................ 151
4.3.4 Control System and Tuning ................................... 152
4.3.5 Electrical Circuit Design ...................................... 157
4.3.6 Final System ..................................................... 159
4.3.7 Parameter Identification ...................................... 160
4.3.8 RF Control for Maneuvering ............................... 167

4.4 SUMMARY .......................................................... 175

CHAPTER 5 ENERGY OPTIMIZATION METHODS ...................... 177

5.1 TWO INERTIA SYSTEM ........................................ 179
  5.1.1 State-Space Equation ......................................... 181
  5.1.2 Cost Function .................................................. 182
  5.1.3 Optimization Method ....................................... 184
  5.1.4 Simulation Results .......................................... 190
  5.1.5 Experimental Results ...................................... 202
  5.1.6 Discussion of Results ....................................... 204

5.2 H-BOT ............................................................... 206
  5.2.1 State Space Equation ......................................... 208
  5.2.2 Cost Function .................................................. 212
  5.2.3 Optimization Method ....................................... 213
  5.2.4 Simulation Results .......................................... 224
  5.2.5 Experimental Results ....................................... 235
5.2.6 Discussion of Results ................................................................. 238

5.3 SELF-BALANCING TRANSPORTER ............................................. 239

5.3.1 State-Space Equation ............................................................... 239

5.3.2 Cost Function ........................................................................... 240

5.3.3 Optimization Method ................................................................. 240

5.3.4 Simulation Results ................................................................. 248

5.3.5 Experimental Results ............................................................... 258

5.3.6 Discussion of Results ................................................................. 261

CHAPTER 6 CONCLUSIONS ................................................................. 263

6.1 CONCLUSIONS ........................................................................... 263

6.2 FUTURE WORK ............................................................................... 266

BIBLIOGRAPHY .................................................................................. 267

APPENDIX A  - ARDUINO CODE TO DECODIFY ENCODERS ........... 276

APPENDIX B  - ARDUINO CODE TO READ ANGULAR POSITION IN THE CONTROL BOARD 282

APPENDIX C  - SENSOR FUSION DERIVATION .................................. 285

APPENDIX D  - MATLAB CODE FOR TWO-INERTIA SYSTEM ............. 287
LIST OF TABLES

Table 2-1 – Comparison between several motion profiles for an index move of a unitary displacement in 1 second. .......................................................................................................................... 45

Table 2-2 - Parameters of a sample multiple profile........................................................................ 48

Table 3-1 - Parameters of the two-inertia system ......................................................................... 64

Table 3-2 - Selected motor for the two-inertia system ............................................................... 68

Table 3-3 - Parameters of the H-Bot .......................................................................................... 87

Table 3-4 - Selected motor for the H-bot .................................................................................. 89

Table 3-5 – Parameters for the self-balancing transporter and motors ...................................... 104

Table 3-6 – Selected motor for the self-balancing transporter ................................................ 107

Table 4-1 - Selected motor for the two-inertia system ............................................................... 121

Table 4-2 – Parameters of the two-inertia system ...................................................................... 121

Table 4-3 – Load inertia ........................................................................................................ 122

Table 4-4 – Data of motor shaft and rod .................................................................................... 123

Table 4-5 – Data of the coupling .............................................................................................. 123

Table 4-6 – Parameter of the motors for the H-Bot ................................................................. 139

Table 4-7 – Parameters of the H-Bot ....................................................................................... 139

Table 4-8 – Electrical characteristics of the gyroscope LPY503AL ........................................ 142

Table 4-9 – Electrical characteristics of the accelerometer MMA7361L ......................... 146

Table 4-10 – Motor and system parameters for the self-balancing transporter ...................... 160

Table 4-11 – Parameters of the Wireless Hi Sensitivity Receiver Module RWS-371-6 .... 168

Table 4-12 – Parameters of the Wireless Hi Power Transmitter Module TWS-BS-3... 169

Table 5-1 - Comparison of energy consumption from the simulation results for various types of motion profiles applied to the two-inertia system to move 0.1revs in 100ms with low position-following error ........................................................................................................... 196
Table 5-2 - Comparison of energy consumption from the simulation results for various types of motion profiles applied to the two-inertia system to move 0.1revs in 100ms with higher position-following error

Table 5-3 - Comparison of energy consumption from the experimental results for various types of motion profiles applied to the two-inertia system to move 0.1revs in 100ms

Table 5-4 – Results for iterative process to compute Q and P

Table 5-5 – Optimized trajectory for various initial P and Q values

Table 5-6 – Effect of P to reach target position

Table 5-7 - Comparison of energy consumption for various types of motion profiles applied to the H-Bot system to move through a circular trajectory of radius of 0.04m in 5 seconds with 2.7kg load

Table 5-8 - Comparison of energy consumption for various types of motion profiles applied to the physical H-Bot system to move through a circular trajectory of radius of 0.04m in 5 seconds with 2.7kg load

Table 5-9 – Results for iterative process to compute Q and P with random initial conditions for a move time of 5 seconds, $x_{\text{target}} = 2m$, $x_{\text{tolerance}} = 0.2m$, and $Err_{\text{max}} = 0.04m$

Table 5-10 – Resulting energy and trajectory for iterative process of computing Q and P with arbitrary initial conditions for a move time of 5 seconds, $x_{\text{target}} = 2m$, $x_{\text{tolerance}} = 0.2m$, and $Err_{\text{max}} = 0.04m$

Table 5-11 - Comparison of energy consumption for various types of motion profiles applied to the self-balancing transporter to move 2m in 5 seconds in simulation

Table 5-12 - Comparison of energy consumption for various types of motion profiles applied to the physical self-balancing transporter to move 2m in 5 seconds in $x$-direction.
LIST OF FIGURES

Fig. 1-1 – Dissertation organization.................................................................................. 16

Fig. 2-1 - Typical configuration of single axis servo systems ........................................ 18

Fig. 2-2 - Comparison of two types of motion profiles for a 1 motor revolution in 0.15 seconds. The first index move is a 5th Order Polynomial motion profile while the second move is a Trapezoidal motion profile. .............................................................................. 22

Fig. 2-3 - Servo system with controller, communication module, servo drive, motor and feedback device (encoder). ......................................................................................................................... 23

Fig. 2-4 - Cam profile constructed with eight data points. (a) Cam profile obtained via linear interpolation. (b) Cam profile obtained via second order interpolation ............................ 25

Fig. 2-5 – Example of asymmetric motion profile............................................................ 26

Fig. 2-6 – Example of a motion profile with constant speed segment ................................ 27

Fig. 2-7 – Trapezoidal profile ........................................................................................... 31

Fig. 2-8 – Example of position, velocity, acceleration, and jerk trapezoidal profile for 20% acceleration time ................................................................................................................. 32

Fig. 2-9 – Example of position, velocity, acceleration, and jerk for a Cubic (or Parabolic) profile............................................................................................................................... 33

Fig. 2-10 - Example of position, velocity, acceleration, and jerk for a Simple Harmonic Motion (SHM) profile.................................................................................................................. 35

Fig. 2-11 - Example of position, velocity, acceleration, and jerk for a Cycloidal profile 36

Fig. 2-12 - Example of position, velocity, acceleration, and jerk for a ModSine motion profile................................................................................................................................. 38

Fig. 2-13 - Example of position, velocity, acceleration, and jerk for a 5th-Order Polynomial motion profile .................................................................................................................... 40

Fig. 2-14 - Example of position, velocity, acceleration, and jerk for a 7th-Order Polynomial motion profile .................................................................................................................. 41

Fig. 2-15 - Example of position, velocity, acceleration, and jerk for a 9th-Order Polynomial motion profile ............................................................................................... 43

Fig. 2-16 - Comparison of motion profiles in terms of (a) position, (b) velocity, (c) acceleration, and (d) jerk profiles. ........................................................................................................ 44
Fig. 2-17 – Example of position, velocity, acceleration, and jerk for a 5th-order polynomial profile with $s_0 = 3$, $s_f = 7$, $v_0 = -10$, $v_f = -5$, $a_0 = -30$, and $a_f = 15$ ................. 47

Fig. 2-18 - Example of position, velocity, acceleration, and jerk for a motion profile with multiple segments. ................................................................. 48

Fig. 3-1 - Proposed motor selection process adapter from [59] ................................................................. 54

Fig. 3-2 - Physical system and model of a two-inertia system. ................................................. 56

Fig. 3-3 - Trajectory planning for the two-inertia system ................................................................. 58

Fig. 3-4 – Direct kinetic models of a two-inertia system ................................................................. 61

Fig. 3-5 - Inverse kinetic model of a two-inertia system ................................................................. 61

Fig. 3-6 - Open-loop simulation model for a two-inertia system ................................................. 63

Fig. 3-7 - Torque and velocity profile for motor a of the simulated two-inertia system .......... 64

Fig. 3-8 - Torque vs. speed curve of the two-inertia system with $rms$ torque ($\tau_{rms}$) of 0.41Nm and $rms$ velocity ($v_{rms}$) of 41.7 rpm ......................................................... 66

Fig. 3-9 - Torque vs. speed curve of the selected motor overlapped with the required torque vs. speed curve obtained from the open-loop simulation of the two-inertia system. ......................................................................................... 69

Fig. 3-10 - Cascade PI control for the two-inertia system ................................................................. 70

Fig. 3-11 - Closed-loop simulation of the two-inertia system ......................................................... 73

Fig. 3-12 – Torque and speed profiles from the closed-loop simulation ......................................... 74

Fig. 3-13 - Torque vs. speed curve of the closed-loop two-inertia system ......................................... 75

Fig. 3-14 – Typical configuration of an H-Bot .................................................................................. 75

Fig. 3-15 - Definitions of the diamond shape .................................................................................. 76

Fig. 3-16 - Master command for H-bot .......................................................................................... 79

Fig. 3-17 - Diamond profile split in X and Z axes according to presented trajectory planning ................................................................................................. 81

Fig. 3-18 - Diamond profile split in X and Z axes for a constant speed master command. ................................................................. 81

Fig. 3-19 - Profile for $\theta_1$ and $\theta_2$ obtained from the inverse kinematics of the H-bot ....... 82
Fig. 3-20 - Inverse kinetic model of the H-bot where \( G_1 = \frac{r^2}{4} [2m_1 + m_2 + (4I_a / r^2)] \), \( G_2 = \frac{r^2m_2}{4} \); \( G_3 = \frac{r^2}{4} [2m_1 + m_2 + (4I_B / r^2)] \); and \( G_4 = \frac{r^2m_2}{4} \).

Fig. 3-21 - Simulink model of H-Bot and control system.

Fig. 3-22 - Torque and speed profile for motor a of the open-loop simulation of the H-bot.

Fig. 3-23 - Torque and speed profile with \( \text{rms} \) torque and \( \text{rms} \) speed at 0.47Nm and 206rpm for the open-loop simulation of the H-bot.

Fig. 3-24 - Torque vs. speed curve of the selected motor and gearbox overlapped with the required torque vs. speed curve obtained from the open-loop simulation of the H-Bot.

Fig. 3-25 - Closed-loop model of the H-Bot.

Fig. 3-26 - Torque vs. speed curve of the closed-loop system for the H-Bot.

Fig. 3-27 - Comparison between the positioning accuracy obtained with typical (dashed line) and proposed (solid line) trajectory planning for the H-Bot.

Fig. 3-28 - Comparison between torque for motor a when using proposed (solid line) and traditional (dashed line) trajectory planning.

Fig. 3-29 - Physical system of a self-balancing transporter.

Fig. 3-30 - Trajectory planning for Self-Balancing Transporter.

Fig. 3-31 - Free Body Diagram of the wheel (a) and pendulum (b) of the Balancing Transporter.

Fig. 3-32 - Direct kinetic model of a self-balancing transporter.

Fig. 3-33 – Sub-system of the direct kinetic equations of a self-balancing transporter.

Fig. 3-34 - Simulink model of the inverse kinetic equations of a self-balancing transporter.

Fig. 3-35 - Sub-system of the inverse kinetic equations of a self-balancing transporter.

Fig. 3-36 - Open-loop simulation of the self-balancing transporter.

Fig. 3-37 - Torque and speed requirements of one of the motor to perform the specified end-point trajectory.

Fig. 3-38 - Torque vs. speed requirement of one of the motors to perform the specified end-point trajectory.
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-39</td>
<td>Torque vs. speed curve of the selected motor overlapped with the required torque vs. speed curve obtained from the open-loop simulation of the self-balancing transporter</td>
</tr>
<tr>
<td>3-40</td>
<td>Closed loop model for the self-balancing transporter</td>
</tr>
<tr>
<td>3-41</td>
<td>DC motor model</td>
</tr>
<tr>
<td>3-42</td>
<td>Torque vs. speed curve of the closed-loop system for the H-Bot</td>
</tr>
<tr>
<td>3-43</td>
<td>Command and feedback</td>
</tr>
<tr>
<td>4-1</td>
<td>Physical system of the two-inertia system</td>
</tr>
<tr>
<td>4-2</td>
<td>Apparatus of the two-inertia system</td>
</tr>
<tr>
<td>4-3</td>
<td>Experimental plant Bode plot for the two-inertia system for ( mT )</td>
</tr>
<tr>
<td>4-4</td>
<td>Mechanism of a rotating encoder with 40 pulses per revolution (PPR)</td>
</tr>
<tr>
<td>4-5</td>
<td>Method to generate the signals A and B in an incremental encoder</td>
</tr>
<tr>
<td>4-6</td>
<td>Dual edge encoder decodification</td>
</tr>
<tr>
<td>4-7</td>
<td>AQB decodification of an incremental encoder</td>
</tr>
<tr>
<td>4-8</td>
<td>Diagram of an H-Bridge</td>
</tr>
<tr>
<td>4-9</td>
<td>Simulink code to control the H-Bridge TLE5206</td>
</tr>
<tr>
<td>4-10</td>
<td>H-Bridge control</td>
</tr>
<tr>
<td>4-11</td>
<td>Implementation of the control system of the H-Bot in Simulink to load into the Arduino</td>
</tr>
<tr>
<td>4-12</td>
<td>Detailed control system for the Arduino to control the H-Bot</td>
</tr>
<tr>
<td>4-13</td>
<td>Electrical diagram of the H-Bot</td>
</tr>
<tr>
<td>4-14</td>
<td>Circuit board for control of the H-Bot</td>
</tr>
<tr>
<td>4-15</td>
<td>Final assembly of the H-Bot</td>
</tr>
<tr>
<td>4-16</td>
<td>Gyro Breakout Board LPY503AL</td>
</tr>
<tr>
<td>4-17</td>
<td>Conversion of the gyroscope signal</td>
</tr>
<tr>
<td>4-18</td>
<td>Accelerometer Breakout Board MMA7361L</td>
</tr>
</tbody>
</table>
Fig. 4-19 – Behavior of the accelerometer for static acceleration [107]. ....................... 145

Fig. 4-20 – Signal conversion of the accelerometer MMA7361L................................. 147

Fig. 4-21 – Behavior of the accelerometer for tilt angle set to zero (a) and then to five degrees (b)...................................................................................................................... 148

Fig. 4-22 – Location of the gyroscope and accelerometer on the self-balancing transporter. ...................................................................................................................... 149

Fig. 4-23 – Sensor fusion with accelerometer and gyroscope to compute the tilt angle. 150

Fig. 4-24 – Implementation of the sensor fusion in Simulink. ................................. 151

Fig. 4-25 – Power structure control for the self-balancing control............................ 152

Fig. 4-26 – Arduino Code for control of the self-balancing transporter....................... 156

Fig. 4-27 – Position reference generator for the self-balancing transporter. This model is “Pos Cmd” in Fig. 4-26................................................................. 156

Fig. 4-28 – Forward, reverse, left, and right control for the self-balancing transporter. This model in the subsystem “Forward/Reverse and Left/Right Ctrl” in Fig. 4-26. ...... 157

Fig. 4-29 – Electrical diagram of the self-balancing transporter ............................... 158

Fig. 4-30 – Circuit board for control of the self-balancing transporter ....................... 159

Fig. 4-31 - Finalized self-balancing transporter................................................................ 160

Fig. 4-32 – Approach to measure the center of mass of the pendulum ..................... 162

Fig. 4-33 - Pendulum moment of inertia test ............................................................... 162

Fig. 4-34 - Measurements to calculate moment of inertia of the pendulum of the self-balancing transporter......................................................................................... 164

Fig. 4-35 - Two-axis analog joystick Parallax 27800 with independent 10kΩ potentiometers with common ground and springs to return automatically to the center position..................................................................................................................... 168

Fig. 4-36 – Wireless Hi Sensitivity Receiver Module RWS-371-6 from Wenshing [112] ................................................................................................................................. 168

Fig. 4-37 – Pinout of the Wireless Hi Sensitivity Receiver Module RWS-371-6 from Wenshing [112]........................................................................................................... 169

Fig. 4-38 - Wireless Hi Power Transmitter Module TWS-BS-3 from Wenshing [4].... 169
Fig. 4-39 – Pinout of Wireless Hi Power Transmitter Module TWS-BS-3 from Wenshing [4].................................................................................................................................................. 170

Fig. 4-40 – Electrical diagram for the transmitter (a) and receiver (b)......................... 171

Fig. 4-41 – Arduino code for the transmitter. ................................................................. 173

Fig. 4-42 – Arduino code for the receiver....................................................................... 174

Fig. 4-43 – Remote control (transmitter) for the self-balancing transporter.................. 175

Fig. 5-1 – Overview of the energy optimization method............................................. 179

Fig. 5-2 - Optimized angular motor position $\theta_m$ and optimized angular load position $\theta_l$. ........................................................................................................................................ 192

Fig. 5-3 - Optimized angular motor velocity $\omega_m$ and angular load velocity $\omega$. ........ 192

Fig. 5-4 - Optimized motor voltage $V_a$ ...................................................................... 193

Fig. 5-5 – Acceleration of the optimized load motion profile ........................................ 193

Fig. 5-6 - Closed loop system for the two-inertia system. ............................................. 195

Fig. 5-7 - Comparison of energy consumption for various types of motion profiles applied to the two-inertia system to move 0.1revs in 100ms with low position-following error 196

Fig. 5-8 - Comparison of energy consumption for various types of motion profiles applied to the two-inertia system to move 0.1revs in 100ms with higher position-following error ............................................................................................................................................... 198

Fig. 5-9 – Percentage of energy savings between the optimized motion profile and modsine profile for various torsional stiffness values ......................................................... 199

Fig. 5-10 - Effect of inertia ratio to energy consumption. ............................................... 201

Fig. 5-11 - Percentage of change in energy consumption with the energy ratio ......... 201

Fig. 5-12 - Simulation results with the same position-following error between 0.010 to 0.020 rad .............................................................................................................................. 203

Fig. 5-13 - Diagram of an H-Bot. .................................................................................. 207

Fig. 5-14 - Energy optimization algorithm ..................................................................... 218

Fig. 5-15 - Energy Optimization Method for an H-Bot. ................................................. 219

Fig. 5-16 - Method to recalculate $P$ for the energy optimization method of an H-Bot. . 221
Fig. 5-17 - Method to recalculate $Q$ for the energy optimization of an H-Bot......... 222

Fig. 5-18 - Identification of $Err_{path}$, desired trajectory, boundaries, and optimal trajectory. .................................................................................................................................................. 223

Fig. 5-19 - Optimized motion profile......................................................................................................................... 230

Fig. 5-20 - Angular position $\theta_1$ and $\theta_2$ calculated with $R=1\times I_{(2\times2)}$ and $R=10\times I_{(2\times2)}$: ....... 231

Fig. 5-21 - Model of an H-Bot, control systems, and motors .............................................................................................. 233

Fig. 5-22 - Total energy required by several motion profiles and the optimized motion profile......................................................... 234

Fig. 5-23 - Energy consumption by the optimized motion profile and eight others for the H-Bot with a 2.7kg load ........................................................................................................................................... 237

Fig. 5-24 - Unit trajectory for the self-balancing transporter ................................................................................................. 241

Fig. 5-25 - Energy Optimization Method for a Self-Balancing Transporter. .............. 242

Fig. 5-26 - Method to recalculate $P$ for the energy optimization method of a self-balancing transporter............................................................... 243

Fig. 5-27 – Detailed algorithm to iteratively recalculate $P$ to optimize the final position of $x(T)$. ................................................................. 245

Fig. 5-28 - Method to recalculate $Q$ for the energy optimization method of a self-balancing transporter ............................................................... 246

Fig. 5-29 - Pseudo-code to calculate the distance between the optimized trajectory $x(t)$ and the desired trajectory $x_d(t)$ ........................................................................................................................................... 247

Fig. 5-30 – Pseudo-code to recalculate $Q$ to obtain an optimize $x(t)$ that resides within the boundaries defined by $x_{tolerance}$. ........................................................................................................................................... 247

Fig. 5-31 – Pseudo-code to recalculate $Q$ to optimize $x(t)$ in terms of energy when the trajectory of $x(t)$ resides within the boundaries defined by $x_{tolerance}$. ........................................................................................................................................... 248

Fig. 5-32 – Linear displacement $x(t)$ calculated with $R = 1$ and $R = 10$. ...................... 254

Fig. 5-33 – Model of the self-balancing transport for validation of the energy optimization method. ........................................................................................................................................... 256

Fig. 5-34 - Total energy required by the self-balancing transporter while commanded by several motion profiles and the optimized motion profile ................................................................. 257

Fig. 5-35 - Energy consumption by the optimized motion profile and eight others for the self-balancing transporter ........................................................................................................................................... 260
Fig. B-1 - Sub-assembly in Simulink to read the angular position of motor 1 from the
digital inputs.......................................................................................................................... 282

Fig. B-2 - Content of the sub-assembly used to read the angular position of motor 1 from
the digital inputs.................................................................................................................. 282

Fig. B-3 - Content of the block “Convert Encoder Position from Cyclic to Linear” ..... 283
CHAPTER 1

Introduction

1.1 Definition of the Problem

The need to design and build industrial machines with higher energy efficiency, reliability, flexibility, and accuracy has increased to satisfy a market that is demanding higher productivity at reduced costs in a sustainable manner. Thus, engineers are challenged to deliver high-performance machines in shorter periods of time due to tighter, more complex, and challenging specifications [1, 2]. As machines become more complex, model-based design has helped the scientific and industrial communities overcome the challenges in mechatronic system design [1, 3-8]. However, well-designed and well-built mechanical systems, along with well-designed and well-configured control systems are necessary, but not enough, to create machines that can operate at high-performance levels. Well-sized motors commanded by properly-designed motion profiles are the other two characteristics of high-performance machine. Thus, methodologies to properly design motion profiles and size motors based on an integrated approach of the various disciplines involved in machine design are needed.

According to [9], over 25% of the electric energy delivered in the U.S. in 2013 was used in the industrial sector. Meanwhile, the energy-use efficiency in the industrial sector was estimated to be 80% [10]. This can be added to the estimate that the purchase cost of a machine accounts for only 2 to 3% of the total cost of ownership, since most of the cost is in energy consumption [11]. This entails major responsibility by the industry to
utilize energy very efficiently and promote sustainable energy usage, since energy conservation is of fundamental importance to preserve finite natural resources [12]. Methodologies to optimize the energy delivered to the industry are essential for a sustainable industrial sector [13].

Therefore, poor machine design leading to low performance, low accuracy, noise, vibration, and premature wear, in addition to the lack of energy optimization methods to reduce the unnecessary energy consumption of machines in the industrial sector, are areas of research of high interest by the industrial sector. These two issues are addressed in this dissertation and a novel mechatronic design method and an energy optimization method are proposed.

1.2 Literature Review on Mechatronic Design Processes

As engineers face more challenging specifications, with an increased number of unknowns during the design process of industrial machines, the amount of work to physically prototype a system and perform iterations to obtain a final product is enormous compared to digital simulations, and many times prohibitive due to cost and time [1, 14]. Problems with vibration and resonant frequencies [15], and concerns with inertia ratio, torque losses, backlash, and compliance, became more critical for machines that need to meet these more stringent requirements. To help overcome these challenges in the design process of complex machines, manufacturers have been utilizing a mechatronic-based approach, also known as model-design approach [2, 5, 7, 8, 14, 16, 17]. Mechatronic design methods are multi-disciplinary approaches that integrate mechanical, electrical, and control systems [2]. Modeling, as a mathematical method to
capture the dynamics of a system [18], and as a key element in mechatronic-design approaches, can be used to uncover design mistakes in an early design state before prototyping. Consequently, time and cost to design are reduced, product quality and performance are improved, and the design process is made more efficient [1, 5, 6]. Mechatronic design methods also allow manufacturers to move from a traditional serial-design approach to a more efficient parallel-design approach.

In a serial-design methodology, a product is developed in sequential order from product specification to mechanical, electrical, software, control design, prototype, validation, and optimization, with some eventual overlaps [1]. Since the various disciplines involved in product development are not designed in an integrated manner in serial design methodologies, extensive prototyping work is in general required to finalize a product. On the other hand, parallel design methods are based on mechatronic design approaches [1]. These approaches are leveraged by virtual prototyping and modeling which allow the integration, evaluation, and simulation of the various components, disciplines, and scenarios of a product before building. The communication among several development groups is also improved with parallel design methods. Thus, this methodology results in improved product quality, reduced prototype costs, reduced development time, and improved development efficiency. The mechatronic methods also allow the development of more complex systems due to the integrated analysis of electrical, mechanical, control, and computer systems [1, 3, 19].

Mechatronic-design approaches are still not widely used in industry. This can be attributed to several factors, including the lack of expertise in modeling, missed or unavailable machine and process data, a reduced or non-existent research and
development department (R&D), and limited time for development [2]. However, some user-stories applying mechatronic design methods have been published over the years, as shown below.

In [20], modeling via bond graphs was used to identify design misconceptions in a truck isolation system and a tympanometer. In [4], the dynamics of a motorcycle simulator were defined via Lagrange’s equations and integrated to the simulator software and control system for satisfactory realism of a riding simulation. In [21], a mechatronic design technique called design for control (DFC) was applied to control a parallel robot by simplifying the dynamic model of the robot, which allowed the use of a simpler control algorithm, while still obtaining satisfactory performance. In [22], the modeling of a dancer for tension control helped to identify a misconception of the mechanical design and determine that a PID control would not yield the required 20 Hz bandwidth which was obtained only with more sophisticated control algorithms. In [16], the mechanical and control systems of a six-degree-of-freedom motion system were developed and integrated into Matlab, Simulink and SimMechanics. This yielded a 30% development-cost reduction, two months design time reduction, and the ability to obtain the higher dynamic performance of the motion system before building it. In [17], Matlab, Simulink and SimMechanics were used to model and integrate the mechanical and control system of a multi-axis test fixture comprised of 18 hydraulic actuators for race car testing, which resulted in improvement in simulation time and the ability to evaluate the durability of mechanical components. In [5], Matlab and Simulink were used as a common modeling environment by GM in the model-based design of a hybrid power train. This accelerated the design process, the evaluation of several scenarios, and the design of the control
system before building a prototype. This approach also allowed the collaboration among engineers in US, Asia, and Europe.

Mechatronic design methodologies have also been researched over the years. In [23], a mechatronic design method based on a mechatronic design quotient (MDQ) was proposed by Saeed Behbahani and Clarence de Silva. This systematic method evaluates several mechanical configurations and actuators and grades each solution in aspects such as cost, reliability, efficiency, and match of task requirements. Those grades constitute the MDQ that is used to select the best solution to build a product. In [24], they also proposed a similar method, based on bond graphs and genetic algorithms, for selecting the best mechanical and control configurations for product design. However, this method does not take into account the motion profile controlling each motor, which highly affects the system performance. Additionally, little guidance was provided in how to properly select the actuators. From a practical perspective, design engineers are in general proficient in either mechanical, electrical, or control engineering as their own area of expertise. But, motor selection and motion-profile design constitute major challenges faced by design engineers, since these steps require the integration of mechanical and electrical engineering.

A method to select the motor and gearhead for general systems was proposed in [25], but limited guidelines were provided to the motion profile design, which is a key element in motor selection and mechatronic design. In [26], an overview of standard industrial guidelines applied to mechatronic design was provided. One of the standards described in [26] was the VDI 2221 [27] that lists the major steps in mechatronic domains (mechanics, electronics, and software) for product design. Another standard
described in [26] was the VDI 2206 [28] which defines the cross-domain interaction in product development. These standards are high-level guidelines in mechatronic design describing the integration of the various domains in product design. However, technical design methodologies are not provided in these standards.

Although some initiatives in mechatronic design approaches, such as those mentioned above, have helped manufacturers to develop better products with more aggregated value, a widely-accepted and integrated model-based design method and a systematic and more general method for machine design are still open areas of research.

1.3 Literature Review on Energy Utilization in the Industry

Energy optimization methods for industrial machines and robots have been researched over the years [29-39]. In [33], a motion-planning method was proposed to optimize path and trajectory planning for a mobile robot. The path for the mobile robot to travel among obstacles between two points was planned by the proposed algorithm and then the trajectory (motion profile) was planned to smooth out the motion. However, the energy for the proposed path was only compared to different paths, but not to the same path with various trajectory-planning approaches to validate the efficacy of the proposed method in minimizing energy.

A genetic algorithm was used to obtain a point-to-point trajectory with a minimum energy for a servomechanism in [31]. However, the proposed method was only used to investigate single-axis systems. No guidelines were provided on how to optimize energy for multi-axis coordinated systems. The only type of trajectory investigated in this work was point-to-point, without instructions on how to apply this method to more complex trajectories.
A method for minimization of energy consumption in industrial robots was proposed in [30]. Losses in the servo drive, motor, and mechanism were taken into account. A gradient-based method was used to solve this non-linear optimization problem. It was claimed that the optimized trajectories yielded energy savings of about 10%. However, the motion profile used to compare the optimized motion profiles was not defined in the paper. Since the motion profile type has a very high impact on the energy consumption in both single-axis and multi-axis coordinated systems, as in a robot, for example, the energy savings claimed in the paper may not be fully validated from the provided data.

Neuro-fuzzy methods to minimize the energy of parallel robots were proposed in [35]. This method was trained off-line to calculate online trajectory planning. However, no experimental results were provided to validate the efficiency of the proposed method in minimizing energy for a two-axis planar robot. Additionally, this is a dedicated method for two-degree-of-freedom planar robots. In [29], a genetic algorithm to search for a point-to-point trajectory with minimum energy was proposed. However, this method requires a computer-based platform which is in general not readily available in industrial applications.

Scheduling methods were proposed to reduce the energy consumption of robots in [37, 38, 40, 41]. Scheduling methods adjust the time for each segment of the motion profile within a machine cycle to create a combination that yields the lowest energy cost. The inconvenience with scheduling methods resides in the change of the time sequence for each segment of the original motion profile controlling a machine. As the motion profile is optimized by the scheduling method, the timing for each segment of the motion
profile changes in the search for a combination of a time sequence that requires lower
energy than the original profile. Since the new schedule or time sequence for each
operation or segment of the motion profile is changed from the original machine
sequence, the new time sequence or scheduling of the optimized motion profile needs to
be analyzed before being used in a machine to verify if it still complies with the machine
functional specifications.

A real-time energy optimization method for a single-axis servomotor system
performing a point-to-point trajectory was proposed in [39]. This optimized trajectory
was generated as a linear constrained optimal control problem. It was claimed that the
optimized trajectory saved about 16% energy in comparison to a trapezoidal profile.
However, in this validation, the optimized motion profile was compared to an asymmetric
trapezoidal profile. As the asymmetry of the trapezoid changes, the energy consumption
of the system commanded by this motion profile also changes. Additionally, no
comparison was performed to other motion profiles, typically used in the industrial
applications, to validate this method. The asymmetry of the trapezoidal motion profile
was also not varied to evaluate the energy savings with the optimized motion profile in
comparison to the trapezoidal profiles with various acceleration ramps (various
asymmetry values).

Several other techniques based on genetic algorithms [42, 43], fuzzy logic [44],
neural-networks [45, 46], machine learning [47], differential evolution algorithms [48],
swarm intelligence methods [49, 50], iterative search algorithms [51, 52], Newton
algorithms [53], probabilistic approaches [54], trajectory planning by avoidance of
motors in the regenerative mode [55-57], and peak-load optimization time scheduling [58] have been used to optimize energy in motor, systems, and production lines.

Energy optimization methods for industrial machines is a field of research that has gained the industrial attention in recent years due to the need of reducing production cost in an increasingly more competitive market.

1.4 Contributions

The major contributions of this dissertation are a novel mechatronic design method [59] and an energy optimization method for industrial machines. The mechatronic design method provides the missing pieces in the literature to enable machine manufactures to design machines that can reach high performance levels. Meanwhile, the energy optimization method provides a novel approach to reduce the electrical energy consumption in industrial machines by modifying solely the motion profile controlling the motor, which allows improving the energy efficiency in the industrial sector.

A mechatronic design method based on model-based design and trajectory planning to size motors of generic single axis systems and generic multi-axis coordinated systems was proposed [59]. As mentioned in Section 1.1, well-designed mechanical and control systems are necessary, but not enough, to obtain high performance machines. Two critical elements to design machines that can achieve high-performance levels are well-sized motors and trajectory planning. The proposed mechatronic design method addresses these two critical steps in machine design.

From a practical stand point, engineers are in general proficient in their domain of expertise in either mechanical, electrical, or control engineering. But, some aspects in machine design require the integration of more than one domain of expertise, as in motor
sizing and trajectory planning. Since motors are the interface between mechanical and electrical systems in a machine, it requires knowledge from both domains to properly size a motor. Similarly, trajectory planning relies on cam profiles to properly control motors. Cam profiles are derived from mechanical cam theory and programmed as an electronic cam in machine code to control a motor [59]. Thus, trajectory-planning design resides in the mechanical and control domains. Thus, the mechatronic design method in this dissertation helps to integrate the various domains involved in machine design in a way absent from the literature.

Other advantages of the proposed mechatronic design method in properly sizing a motor, controlled by motion profiles defined via trajectory planning, include the following combined benefits:

- **Reduction of vibration, noise, and wear**: In order to reduce these undesirable effects, trajectory planning employs motion profile types that avoid high jerk content, where jerk is the derivation of acceleration.

- **Smooth motion of the load**: In several applications, the motion at the load needs to be smooth, as in cases on a conveyor carrying open containers of liquids before capping. In order to avoid spills during the start and stop of the conveyor, the motion profile controlling the load needs to have smooth acceleration and deceleration profiles. This is handled in the trajectory-planning process.

- **Higher performance (lower position- and velocity-following error)**: The motor controlling a system needs to be selected to provide higher bandwidth than the frequencies of the motion profile. The selected motor also needs to be able to
provide enough torque and speed for the application, otherwise position-following error will be inevitable. The tuning of the control loop system is also crucial to achieve the required position- and velocity-following errors. These points are taken into account in the proposed mechatronic design process.

- **Higher production rate**: Similar to the requirements to achieve high performance, the system bandwidth provided by the motor, the motor torque and motor-speed utilization, and well-tuned control systems, are also necessary to achieve higher production rates. Different types of motors and different sizes of motors impact on the energy consumption to achieve a certain production rate. Thus, the type of motor and frame size need be taken into account during the motor-selection process in the proposed mechatronic design method to avoid unnecessary energy waste.

- **Systematic approach to properly select a motor**: The proposed mechatronic design method consists of a systematic approach to select motors for generic single-axis systems or for multi-axis coordinated systems.

An **energy optimization method** based on the optimization of the motion profile to optimize the electrical energy consumption of generic single-axis systems and multi-axis coordinated systems in industrial machines is proposed in this dissertation. Some of the benefits of this method are as follows:

- **Simplified implementation**: This method can be implemented in industrial machines by simply updating the machine code with the optimized motion profile built from the proposed energy optimization method. This method does
not require any mechanical or electrical changes to the machine to optimize energy, since the implementation resides in machine code only.

- **Motion profile scheduling is not changed**: In this proposed energy optimization method, the time for each segment of the motion profile is maintained as defined in the machine functional specification. This proposed method does not change the time for each segment of the motion profile while computing a motion profile to optimize energy, as in scheduling methods. Only the shape of the motion profile is changed in this energy optimization method. Thus, there is no need to verify if the optimized motion profile will comply with the machine functional specification, since the timing of the motion profile was not changed, as in scheduling methods.

- **Motor sizing augmented with the optimized motion profile method**: This energy optimization method can be part of the mechatronic design process and add the benefit of energy consumption optimization while sizing the motor.

- **Method applicable to generic systems**: This method was developed for generic single-axis systems and generic multi-axis coordinated systems, but it was demonstrated and validated for three real systems: a two-inertia system that is a single-axis system, a Cartesian two-axis parallel robot (H-Bot) that is a multi-axis coordinated system, and a self-balancing transporter that is an unbalanced system.

- **Validation aligned with practical applications**: This method was validated by comparing the energy consumption of the systems commanded by the optimized profile to the energy consumption of the systems commanded by
eight other motion profiles typically used in industrial applications. Thus, this validation procedure provides a more realistic comparison to conditions typically found in actual systems.

Some additional contributions of this dissertation are as follows:

- **Multi-disciplinary integration**: This research demonstrates how to integrate several disciplines including mechanical, electrical, and control, and how to apply fundamental concepts in the design of real systems, as demonstrated with the self-balancing transporter, H-Bot, and two-inertia system. This multi-disciplinary integration was demonstrated while designing the control system, designing the electronic power structure, decodifying the encoders, acquiring and treating sensor signals, identifying values of mechanical parameters, and modeling the mechanical system.

- **Implementation of physical systems**: The process and concepts employed in this research demonstrate a full mechatronic system design process from concept to implementation, including electrical design, programming, and parameter identification.

- **Closing the gap between theory and practice**: This research demonstrates how to close the gap between theory and practice in mechatronic designs.

- **Industrial and educational applicability**: The proposed methods presented in this research can be applied to both industrial machines and educational projects.

- **Mechatronics education**: The methods and systems developed in this research can be directly applied in mechatronics education to motivate and stimulate
student curiosity in the understanding of science and engineering applied to physical systems.

• **Methods can be applied to numerous systems:** The proposed mechatronic design and energy optimization methods can be used in a wide range of systems, not only with the self-balancing transporter, H-Bot, and two inertia system.

• **Multi-purpose control board:** The control board that was developed to host the power stage for two dc motors, decodification of two encoders, and control of single-axis or two-axis systems can be used not only with the case-study systems from this research, but with a variety of other systems for laboratory applications. This allows students to quickly and easily understand mechatronic concepts and build new systems such as two-axis Delta robots, inverted pendula, two-axis articulated-arm robots, and two-wheeled robots.

1.5 Dissertation Organization

Following the introduction, this dissertation contains five additional chapters as shown in Fig. 1-1. In Chapter 2, several types of motion profiles typically used in the industry are presented. The equations defining position, velocity, acceleration, and jerk for each one of these motion profiles are given in this chapter, in addition to a comparison of the various types of motion profiles. Guidelines for properly select motion profiles and merge consecutive segments of a motion profile are also given. This chapter provides the background for the trajectory planning approach described in the next chapter.
In Chapter 3, the proposed mechatronic design method is described and demonstrated with a single-axis two-inertia system, a Cartesian two-axis parallel robot (H-Bot), and a self-balancing transporter. In this method, the motion profiles to control each motor of an industrial machine are designed via trajectory planning. These motion profiles are then used in the mechatronic design process to size the motors.

In Chapter 4, the physical implementation details of these three systems (two-inertia system, Cartesian two-axis parallel robot, and self-balancing transporter) are provided. These three system are used in the validation of the methods described in the next chapter.

In Chapter 5, the proposed energy optimization methods are described and demonstrated with a single-axis two-inertia system, a Cartesian two-axis parallel robot, and a self-balancing transporter. The optimized motion profile designed with the energy optimization method can be used with the mechatronic design process to size the motor for an industrial machine. Thus, the motion profile is designed first with the energy optimization method. This optimized motion profile is then used with the mechatronic design process to size the motors. This can potentially result in a reduced motor and drive frame size. By merging these two methods as shown in Fig. 1-1, the motors for a machine are not only sized properly, but they are also controlled by a motion profiles that minimize the energy usage.

In Chapter 6, the conclusions of this research regarding the mechatronic design method and energy optimization method are provided, followed by a statement of suggested future work.
Fig. 1-1 – Dissertation organization
CHAPTER 2

Motion Profiles

One of the contributions in this dissertation is the mechatronic design process that is based on model-based design and trajectory planning to size motors for industrial machines. The trajectory planning is a method of designing the motion profiles that controls the motors in a machine in order to mitigate vibration, noise, wear, and stress in mechanical and electronic components. This chapter provides the background in motion profiles to properly apply trajectory planning with the proposed mechatronic design method described in the next chapter.

Motion profiles are essential elements in a high-performance mechatronic system. In industrial machines, the motion profiles are programmed in an industrial controller where the machine code resides [60, 61]. The motion profile is the position reference for the control loops located in the servo drive that powers the motor. In general, the servo drive has a cascade PI control loop that consist of an outer position loop, and an inner velocity loop with an inner current loop. More complex control systems including feedforward, observer, and filters are also typically available in servo drives [62]. The motor angular position is measured by a feedback device, typically an encoder or resolver, connected to the motor. The measured feedback signal is used to close the position and velocity loops in the servo drive. The feedback device is, in general, an encoder or a resolver. This architecture is typical for servo-systems and it is shown in Fig. 2-1.
Motion profiles are also called cam profiles or electronic cams and they are highly beneficial in industry applications to minimize vibration, reduce stress of mechanical components, increase machine productivity, and improve the overall performance of mechatronic systems [63]. The benefits obtained with electronic cams are similar to the benefits obtained with mechanical cams, since electronic cams are designed with equations derived from mechanical cams and, therefore, share similar properties.

A servo-axis system consists of a motor and servo drive, which are used to control the mechanisms responsible for a particular function in a machine, such as the joints of robots, conveyor belts, rotary knifes, and sealing jaws. The electronic cams programmatically define the position and velocity profile that a servo axis needs to follow to move from the actual position to a target position in a pre-defined amount of time. The type of motion profile to properly perform a required task depends on the system dynamics, system limitations, process, load, and machine functional specification. For example, in an application where a cart moves on a track carrying a tall pile of boxes, the motion profile to start and stop motion needs to be smooth enough to avoid the load to tip over. This can be achieved by selecting motion profiles with zero acceleration at the
beginning and the end of motion. A typical type of motion profile extensively used in motion applications is called trapezoidal profile. But, this type of profile would not be recommended for this application due to its high jerk content. However, a 5th Order Polynomial profile described next would potentially be a proper fit for this application due to the smooth acceleration profile.

Electronic cams allow great flexibility in comparison to mechanical cams in controlling mechanisms and loads because the motion of the mechanism can be modified in the machine code without any mechanical changes as it would be required with mechanical cams. Thus, various electronic cams can be tested in a physical system without any mechanical modification. The only change is in the electronic cam that resides in the machine code. This allows to evaluate the behavior of a system to different cam profiles and select the one that best perform the required task if needed.

Industrial machines are typically controlled by industrial controllers that, in general, provide easy-to-use instructions to generate trapezoidal motion profiles. Thus, the implementation of any other type of motion profile in machine code would require programming of equations that define the desired cam profile. Since the theory and implementation of electronic cams resides between the domains of mechanical and control engineering, respectively, multi-domains expert engineers are required to properly employ electronic cams to control machines. Partially due to the more complex implementation and the need for expertise in multiple areas, electronic cams are still not widely used in the industrial sector.

The proper selection of electronic cams to control industrial machines allows to reduce vibration, audible and electrical noise, wear and stress in electrical and mechanical
components, as well as reduce position and velocity following error [63-65]. Some of the main concepts to use cam profiles in industrial machine, and the equations of several motion profiles typically used in industrial applications will be provided next.

2.1 Characteristics of Motion Profiles

A typical motion profile consists of an acceleration segment followed by either a constant speed segment or simply by the deceleration segment. This defines a simple index move. More complex motion profiles consist of multiple segments merging position, velocity, and acceleration from segment to segment. The complexity, move distance, move time, and profile type of a motion profile for a given application depends on the mechanism, machine rate (machine cycles per minute), load, task, and machine constraints.

One of the main desired features in a motion profile is smoothness. Smooth motion profiles reduce vibration, noise, and stress on mechanical components, and consequently reduce the stress on motors and drives (servo drives or variable-frequency drives – VFDs), which reduces the likelihood of motor and drive failures.

The smoothness of a motion profile is identified from the acceleration profile. Motion profiles with zero initial acceleration, zero final acceleration, and no discontinuity during motion for a simple index move, characterizes smooth motion profiles that are, therefore, preferred for industrial applications. Meanwhile, motion profiles with initial and/or final acceleration different than zero for a simple index move, or profiles with a step change in acceleration during motion, should be avoided for industrial applications because infinite jerk will occur at the beginning and end of the move and at each step change in acceleration. Jerk, the derivative of acceleration, is an indicator to vibration
been introduced into the system through the motion profiles used as the reference signal that commands motion. High jerk components cause the undesirable effects aforementioned. Trapezoidal, cubic, and sine profiles are examples of motion profiles to be avoided due to the high jerk content, while cycloidal, modsine, 5th-order polynomial, 7th-order polynomial, and 9th-order polynomial are types of preferred motion profiles due to smoothness [63]. These motion profiles are defined and further described later in this chapter.

The vibration from discontinuities in acceleration may be magnified in systems with compliance and/or backlash. Backlash is present in geared systems such as gearboxs, rack-and-pinion systems, lead-screws, and gear-to-gear transmission systems. An example is shown in Fig. 2-2 for a two-inertia system. The two-inertia system used to generate this figure can be considered rigid since the load is directly connected to the motor shaft yielding very low compliance values. In this figure, two index moves are shown. The first index move is with a smooth velocity profile (5th-order polynomial profile) while the second index move is a jerky motion profile (trapezoidal profile). By comparing motor current signal during the first index move to the motor current during the second index move, it can be observed that the current signal of the second index move is more jerky (high frequency oscillations in current) than the current signal of the first index move. Additionally, the current signal of the second index move presented step changes that were not observed with the first index move. The jerky motion and the step changes in current are sources of vibration, noise, and stress to the mechanical and electrical system. It should be noticed, that these undesirable effects are avoidable via proper selection of the motion profiles controlling the load of the system. The move time
and move distance is the same in both moves in Fig. 2-2. The only difference is the motion profile. This demonstrates that the same move can be performed smoothly or aggressively, depending on the motion profile selection.

Fig. 2-2 - Comparison of two types of motion profiles for a 1 motor revolution in 0.15 seconds. The first index move is a 5th Order Polynomial motion profile while the second move is a Trapezoidal motion profile.

2.2 Computation of Cam Profiles

A servo-axis system consists of a controller, a servo drive, a motor, and a feedback device as shown in Fig. 2-3 [66]. The computation of the cam profiles occurs at the controller level as shown in Fig. 2-3. The position reference (position command) is sent from the controller to the servo drive in real-time through a communication module. Real-time control is discussed in the next section. Typical real-time communication
protocols include Ethernet, SERCOS (SErial Realtime COmmunications System), or analog signal [67-69]. Two of the elements in a servo drive is a power stage and a control system. The power stage applies a voltage signal with variable amplitude and a frequency to the motor to control torque and speed. The control system is as described earlier for Fig. 2-1. This allows the motor to follow the position command from the controller.

Fig. 2-3 - Servo system with controller, communication module, servo drive, motor and feedback device (encoder).

The equations that define the desired cam profile to control a system are, in general, implemented in the controller (see Fig. 2-3) and converted to a cam table: time vs. position. However, when multiple axes need to produce synchronized motion in a given application, the cam tables are defined as master position vs. slave position instead. In this case, all the slave axes follow the motion of the master axis according to the relationship defined in the cam table for each slave axis.

The data points in this table are interpolated to obtain the exact position reference value to be sent to the drive in real-time. If the move time is 100 milliseconds and the controller send a data-point of position reference to the servo drive every 5 milliseconds,
then 20 data-points (one at every 5 milliseconds) will be sent to the drive while commanding this move. This is how the position reference signal that was calculated in the controller is reproduced in the drive. The position command is computed in the controller instead of the drive to facilitate the programing and synchronization of multi-axis systems.

As the number of data points in this cam table increases, the processing usage of the controller to interpolate the position command increases. On the other hand, as the number of data points that define the cycle profile reduces, the ability to accurately represent the desired motion profile through data points and correctly interpolate reduces. Thus, the number of data points to correctly represent a cycle profile relies on a balance between accuracy to represent the desired motion profile and computation time.

The number of points to accurately represent a cam profile also depends on the interpolation method used to obtain the position command at every update of the position command sent to the drive. Linear, second, and third-order interpolations are common techniques used to interpolate cam tables. Linear interpolation requires less computation time from the controller than higher order interpolation methods, but it requires larger number of data points to accurately represent a cam profile.

The effect of the type of interpolation is shown in Fig. 2-4. A cam profile represented by eight data points with linear interpolation is shown in Fig. 2-4a. The same cam profile represented by the same eight data-points, but now with second-order interpolation, is shown in Fig. 2-4b. As shown in Fig. 2-4, the error between the exact motion profile and the resulting motion profile obtained via linear interpolation (Fig. 2-4a) is higher than the error obtained with second-order interpolation (Fig. 2-4b). Thus,
linear interpolation adds error to the reconstructed cam profile when a small number of data points is used. A second order interpolation helps reducing this error as shown in Fig. 2-4b, although it required more computation time.

![Fig. 2-4 - Cam profile constructed with eight data points. (a) Cam profile obtained via linear interpolation. (b) Cam profile obtained via second order interpolation](image)

2.3 Real-Time Control

Real-time control is related to the way that data is transferred between devices. Real-time control also called deterministic control is used in closed-loop servo systems to send either a position, velocity, or torque command from a controller to the servo drives. The determinism is typically obtained via two methods: time-stamping each data-point sent from the controller to the servo drive, or sending data-points at very precise rates. In the first method, the clock of the controller and servo drive are synchronized, and each time that controller updates the servo drive, one data-point of position, velocity, or torque is sent along with the respective time-stamp. Thus, the controller does not need to send
data points to the drive at exactly the same interval of time every time as in the second method where the controller send data to the drive at constant rates. A typical media for the first method is Ethernet, while SERCOS and analog signals are most commonly used for the real-time control using the second method. Both methods yield a real-time control system, which is fundamental importance for the methods described in this dissertation to allow the servo drives to accurately receive a position reference signal and then control the motor to follow it.

2.4 Asymmetric Motion Profiles

Asymmetric motion profiles are those in which the acceleration time differs from the deceleration time as in the example shown in Fig. 2-5.

![Fig. 2-5 – Example of asymmetric motion profile](image)

Asymmetric motion profiles may be beneficial in applications where the required torque to accelerate is different from the require torque to decelerate the load and the mechanism. Thus, different acceleration and deceleration times allows a balance between
the torque requirements to accelerate and decelerate the load. A better balance between acceleration torque and deceleration torque increases the available torque that can be used to increase the speed of the mechanism and consequently the number of machine cycles per minute which results in gains of productivity.

2.5 Motion Profiles with Constant Speed Segment

A long index move may require a motion profile with a constant-speed segment in order to limit the maximum speed of the move. The constant speed segment is added between the acceleration and deceleration segments of the motion profile as shown in Fig. 2-6.

![Fig. 2-6 – Example of a motion profile with constant speed segment](image)

The maximum velocity of a mechanism, load, and motor can be used to determine the need for a motion profile with a constant speed section. If any of the speed limits or
acceleration limits in the system are reached during a long index move, than a constant speed segment can be added to the motion profile as shown in Fig. 2-6 to limit the maximum velocity and/or acceleration. If necessary, asymmetric motion profiles as shown in Section 2.4 and constant speed section can be combined to generate a motion profile for long moves while balancing the motor torque during acceleration and deceleration.

2.6 Types of motion profiles

Several types of motion profiles can be used as the position reference signal to control servo systems (servo-axes) in machines. There is no single type of motion profile that is the best option for all types of applications. Each type of motion profile has pros and cons that need to be taken into account when selecting a type of motion profile for a particular application.

Typical motion profiles used in industrial machines are:

- Trapezoidal
- Cubic
- Simple Harmonic Motion (Sine)
- Cycloidal
- ModSine
- 5th-Order Polynomial
- 7th-Order Polynomial
- 9th-Order Polynomial

The equations that describe the position, velocity, acceleration, and jerk for each one of these motion profiles along with a brief description are provided next.
2.6.1 Trapezoidal

A trapezoidal motion profile is a first-order velocity profile with a constant acceleration segment and a constant deceleration segment. This results in a theoretically infinite jerk at each step change in acceleration which can cause abrupt motion and vibration in the mechanical system and motors. Despite these undesirable effects, trapezoidal profiles are very common in industrial applications due to the simplicity of implementation. Trapezoidal profiles also provide the ability to change the acceleration and/or deceleration rates for a given move by simply changing the amount of time allocated for each one of the three segments that constitute a trapezoidal motion profile: acceleration, constant speed, and deceleration. However, this profile should be avoided due to the high jerk content.

The equation for the second order position profile \( s \) of a trapezoidal move and the respective velocity \( v \), acceleration \( a \), and jerk \( j \), are defined as follows:

\[
0 \leq t \leq t_1 \quad s = V_{\text{max}} \frac{t^2}{2t_1} \\

v = \frac{ds}{dt} = V_{\text{max}} \frac{t}{t_1} \\
a = \frac{d^2 s}{dt^2} = V_{\text{max}} \frac{1}{t_1} \\
j = \frac{d^3 s}{dt^3} = 0
\]
\[ t_1 \leq t \leq t_2 \quad s = V_{\text{max}} \left( t - \frac{t_1}{2} \right) \]

\[ v = \frac{ds}{dt} = V_{\text{max}} \]

\[ a = \frac{d^2 s}{dt^2} = 0 \]

\[ j = \frac{d^3 s}{dt^3} = 0 \]

\[ t_2 \leq t \leq t_3 \quad s = V_{\text{max}} \left( t - \frac{t_1}{2} - \frac{(t - t_2)^2}{2(t_3 - t_2)} \right) \]

\[ v = \frac{ds}{dt} = V_{\text{max}} \left( 1 - \frac{t - t_2}{t_3 - t_2} \right) \]

\[ a = \frac{d^2 s}{dt^2} = -V_{\text{max}} \left( \frac{1}{t_3 - t_2} \right) \]

\[ j = \frac{d^3 s}{dt^3} = 0 \]

Where, \( t \) is instantaneous time, \( t_1 \) is the acceleration time, \( t_2 \) the acceleration time plus the time at constant speed, \( t_3 \) is the total move time, and \( V_{\text{max}} \) is the maximum move speed defined as follows:

\[ V_{\text{max}} = \frac{2S}{t_3 + t_2 - t_1} \quad (2) \]

Where, \( S \) is the total move distance. The times \( t_1, t_2, t_3 \), and the maximum velocity \( V_{\text{max}} \) are show in Fig. 2-7. It should be pointed out that the jerk is infinite at time zero, \( t_1, t_2, \) and \( t_3 \).
For symmetric trapezoidal velocity profiles, the times $t_1$ and $t_2$ can also be defined as a percentage of the total move time $t_3$ as follows.

\[
\begin{align*}
t_1 &= t_p \times t_3 \\
t_2 &= t_1 + (100 - 2t_p) \frac{t_3}{3}
\end{align*}
\]  

(3)

Where, $t_p$ is the percentage of the move time $t_3$ used for the acceleration segment. Thus, $t_p$ can vary from 0 to 50%. When $t_p$ is zero, the acceleration time is zero, which is an impossible velocity profile to follow, and when $t_p$ is 50%, the resulting velocity profile is a triangular motion profile.

Although jerk is zero during constant acceleration and deceleration, it is infinite during the transitions in acceleration and deceleration causing mechanical stresses on the load and motors.

An example of position, velocity, acceleration, and jerk profiles for unitary move in one second with a Trapezoidal motion profile is shown in Fig. 2-8.
2.6.2 Cubic

The Cubic profile also called parabolic profile is a second order velocity profile. This motion profile has the lowest peak speed among the types of motion profiles described in this chapter. Although the lower peak speed required with cubic profiles is an advantage, this type of profile still presents infinite jerk at the beginning and at the end of the move due to the step changes in acceleration. Similarly to the trapezoidal move, the cubic motion profile should also be avoided in industrial applications due to the high jerk content.

The position, velocity, acceleration, and jerk profiles of a cubic motion profile are given as follows:
\[ s = -2S \frac{t^3}{T^3} + 3S \frac{t^2}{T^2} \]
\[ v = \frac{ds}{dt} = -6S \frac{t^2}{T^3} + 6S \frac{t}{T^2} \]
\[ a = \frac{d^2s}{dt^2} = -12S \frac{t}{T^3} + 6S \frac{1}{T^2} \]
\[ j = \frac{d^3s}{dt^3} = -12S \frac{1}{T^3} \]  

(4)

Where, \( T \) is the move time. An example of position, velocity, acceleration, and jerk profiles for unitary move in one second with a cubic motion profile is shown in Fig. 2-9:

Fig. 2-9 – Example of position, velocity, acceleration, and jerk for a Cubic (or Parabolic) profile
2.6.3 Simple Harmonic Motion (SHM)

The Simple Harmonic Motion (SHM) or Sine velocity profile is commonly used in an attempt to smooth out the motion of mechanical loads. Although, acceleration and jerk are smooth throughout the move, jerk is infinite at the beginning and end of the move which may result in undesirable effects. Thus, the sine profile is another motion profile along with the trapezoidal and cubic to be avoided in industrial applications [63].

The position, velocity, acceleration, and jerk profiles of a Sine profile are defined as follows [63]:

\[
\begin{align*}
    s &= \frac{S}{2} \left( \frac{S}{2} \cos \left( \frac{\pi t}{T} \right) \right) \\
    v &= \frac{ds}{dt} = \frac{\pi S}{2T} \sin \left( \frac{\pi t}{T} \right) \\
    a &= \frac{d^2 s}{dt^2} = \frac{\pi^2 S}{2T^2} \cos \left( \frac{\pi t}{T} \right) \\
    j &= \frac{d^3 s}{dt^3} = -\frac{\pi^3 S}{2T^3} \sin \left( \frac{\pi t}{T} \right)
\end{align*}
\]  

(5)

An example of position, velocity, acceleration, and jerk profiles for unitary move in one second with a Sine motion profile is shown in Fig. 2-10.
2.6.4 Cycloidal

The Cycloidal is a sine acceleration profile that yields zero acceleration at the ends of the profile which is a desirable feature to improve motion smoothness. This profile is known as cycloidal displacement or sinusoidal acceleration.

The position, velocity, acceleration, and jerk profiles of a Cycloidal profile are defined as follows [63]:

\[
\begin{align*}
    s &= S \left( \frac{t}{T} - \frac{1}{2\pi} \sin \left( 2\pi \frac{t}{T} \right) \right) \\
    v &= \frac{ds}{dt} = \frac{S}{T} \left[ 1 - \cos \left( 2\pi \frac{t}{T} \right) \right] \\
    a &= \frac{d^2s}{dt^2} = 2\pi \frac{S}{T^2} \sin \left( 2\pi \frac{t}{T} \right) \\
    j &= \frac{d^3s}{dt^3} = 4\pi^2 \frac{S}{T^3} \cos \left( 2\pi \frac{t}{T} \right)
\end{align*}
\]
An example of position, velocity, acceleration, and jerk profiles for unitary move in one second with a Cycloidal motion profile is shown in Fig. 2-11.

![Position, Velocity, Acceleration, and Jerk Profiles](image)

Fig. 2-11 - Example of position, velocity, acceleration, and jerk for a Cycloidal profile

2.6.5 ModSine

The ModSine is also known as Modified Sinusoidal Acceleration. The Modsine is also a smooth motion profile as the Cycloidal profile due to the zero acceleration rate at the beginning and end of the move, but with lower peak velocity and peak acceleration than the Cycloidal profile. The peak velocity and peak acceleration of the ModSine profile is approximately 13% lower than the Cycloidal. However, the jerk profile of Modsine is less smooth than the Cycloidal profile. The ModSine profile is in general an appropriate choice for high inertia applications since has the lowest acceleration rate among the recommended motion profiles, shown in this chapter, for industrial
applications. Thus, lower acceleration rates for high inertial system help reducing the motor torque to drive the load. Therefore, the Modsine motion profile provides a reasonable balance between low acceleration and smoothness.

The equations for position, velocity, acceleration, and jerk of the three segments that constitute a ModSine profile are as follows [63].

For $0 \leq t < T/8$

\[
s = S \left[ 0.43990085 \frac{t}{T} - 0.0350062 \sin \left( 4 \pi \frac{t}{T} \right) \right]
\]
\[
v = \frac{ds}{dt} = 0.43990085 \frac{S}{T} \left[ 1 - \cos \left( 4 \pi \frac{t}{T} \right) \right]
\]
\[
a = \frac{d^2s}{dt^2} = 5.5279571 \frac{S}{T^2} \sin \left( 4 \pi \frac{t}{T} \right)
\]
\[
j = \frac{d^3s}{dt^3} = 69.4663577 \frac{S}{T^3} \cos \left( 4 \pi \frac{t}{T} \right)
\]

For $T/8 \leq t < 7T/8$

\[
s = S \left[ 0.28004957 + 0.43990085 \frac{t}{T} - 0.3150577 \cos \left( \frac{4 \pi}{3} \frac{t}{T} - \frac{\pi}{6} \right) \right]
\]
\[
v = \frac{ds}{dt} = 0.43990085 \frac{S}{T} \left[ 1 + \sin \left( \frac{4 \pi}{3} \frac{t}{T} - \frac{\pi}{6} \right) \right]
\]
\[
a = \frac{d^2s}{dt^2} = 5.5279571 \frac{S}{T^2} \cos \left( \frac{4 \pi}{3} \frac{t}{T} - \frac{\pi}{6} \right)
\]
\[
j = \frac{d^3s}{dt^3} = -23.1553 \frac{S}{T^3} \sin \left( \frac{4 \pi}{3} \frac{t}{T} - \frac{\pi}{6} \right)
\]
For $7T/8 \leq t < T$

\[
s = S \left\{ 0.56009915 + 0.43990085 \frac{t}{T} - 0.0350062 \sin \left[ 2\pi \left( \frac{2}{T} t - 1 \right) \right] \right\}
\]

\[
v = \frac{ds}{dt} = 0.43990085 \frac{S}{T} \left\{ 1 - \cos \left[ 2\pi \left( \frac{2}{T} t - 1 \right) \right] \right\}
\]

\[
a = \frac{d^2 s}{dt^2} = 5.5279571 \frac{S}{T^2} \sin \left[ 2\pi \left( \frac{2}{T} t - 1 \right) \right]
\]

\[
j = \frac{d^3 s}{dt^3} = 69.4663577 \frac{S}{T^3} \cos \left[ 2\pi \left( \frac{2}{T} t - 1 \right) \right]
\]

An example of position, velocity, acceleration, and jerk profiles for unitary move in one second with a ModSine motion profile is shown in Fig. 2-12.
2.6.6 5\textsuperscript{th}-Order Polynomial

The 5\textsuperscript{th}-order polynomial position profile provides smooth motion due to zero acceleration rates at the ends of the motion profile and also due to the continuous acceleration profile during motion. The 5\textsuperscript{th}-order polynomial profile is also known as 3-4-5 polynomial. The acceleration of the 5\textsuperscript{th}-order polynomial profile is similar to the Cycloidal acceleration profile, but slightly lower peak acceleration. Since the jerk profile of the Cycloidal and 5\textsuperscript{th}-order polynomial profiles are not zero at both ends of the move, oscillations in velocity may still happen for high-speed applications. The improvement in smoothness provided by a 5\textsuperscript{th}-order polynomial profile in comparison to a Trapezoidal profile is shown in Fig. 2-2 where significant lower torque ripple was obtained with the 5\textsuperscript{th}-order polynomial profile for the same move.

The position, velocity, acceleration, and jerk profiles of a 5\textsuperscript{th} Order Polynomial profile are defined as follows [63, 64]:

\[
\begin{align*}
    s &= S \left[ 10 \left( \frac{t}{T} \right)^3 - 15 \left( \frac{t}{T} \right)^4 + 6 \left( \frac{t}{T} \right)^5 \right] \\
    v &= \frac{ds}{dt} = S \left[ 30 \frac{t^2}{T^3} - 60 \frac{t^3}{T^4} + 30 \frac{t^4}{T^5} \right] \\
    a &= \frac{d^2s}{dt^2} = S \left[ 60 \frac{t}{T^3} - 180 \frac{t^2}{T^4} + 120 \frac{t^3}{T^5} \right] \\
    j &= \frac{d^3s}{dt^3} = S \left[ 60 \frac{1}{T^3} - 360 \frac{t}{T^4} + 360 \frac{t^2}{T^5} \right]
\end{align*}
\]

(8)

An example of position, velocity, acceleration, and jerk profiles for unitary move in one second with a 5\textsuperscript{th} Order Polynomial motion profile is shown in Fig. 2-13.
2.6.7 7th-Order Polynomial

The 7th-order polynomial profile yields a velocity, acceleration, and jerk profile without discontinuities during motion and zero magnitude at both ends of an index move. These are highly desirable characteristics to reduce vibration. The 7th-Order Polynomial has smother jerk profile than the Cubic, modsine, 5th-order polynomial, and Cycloidal. However, the peak velocity and peak acceleration is higher than the other motion profiles aforementioned. Since the peak acceleration is higher, the required motor peak-torque is also higher, which is a limiting factor to use this profile. The 7th-Order Polynomial profile is also called 4-5-6-7 polynomial.

The position, velocity, acceleration, and jerk profiles of a 7th-Order Polynomial profile are as follows [63]:

![Graphs showing position, velocity, acceleration, and jerk profiles for 7th-Order Polynomial motion profile.](image)
\[ s = S \left[ 35 \left( \frac{t}{T} \right)^4 - 84 \left( \frac{t}{T} \right)^5 + 70 \left( \frac{t}{T} \right)^6 - 20 \left( \frac{t}{T} \right)^7 \right] \]
\[ v = \frac{ds}{dt} = S \left[ 140 \frac{t^3}{T^4} - 420 \frac{t^4}{T^5} + 420 \frac{t^5}{T^6} - 140 \frac{t^6}{T^7} \right] \]
\[ a = \frac{d^2s}{dt^2} = S \left[ 420 \frac{t^2}{T^4} - 1680 \frac{t^3}{T^5} + 2100 \frac{t^4}{T^6} - 840 \frac{t^5}{T^7} \right] \]
\[ j = \frac{d^3s}{dt^3} = S \left[ 840 \frac{t}{T^4} - 5040 \frac{t^2}{T^5} + 8400 \frac{t^3}{T^6} - 4200 \frac{t^4}{T^7} \right] \]

(9)

An example of position, velocity, acceleration, and jerk profiles for unitary move in one second with Cubic motion profile is shown in Fig. 2-14.
2.6.8 \textit{9}^{\text{th}}-\text{Order Polynomial}

The \textit{9}^{\text{th}}-order polynomial position profile is the smoothest motion profile shown in this chapter. However, the peak velocity and peak acceleration is higher than the other profiles described in this chapter. Thus, the tradeoff between smoothness and peak velocity and peak acceleration needs to be taken into account while selecting this type of motion profile. The \textit{9}^{\text{th}}-Order Polynomial position profile is also known as 5-6-7-8-9 polynomial. This type of profile has limited application due to the high acceleration values and consequently high torque requirement.

The position, velocity, acceleration, and jerk profiles of a \textit{9}^{\text{th}} Order Polynomial profile are defined as follows [63]:

\begin{align*}
    s &= S \left[ 126 \left( \frac{t}{T} \right)^5 - 420 \left( \frac{t}{T} \right)^6 + 540 \left( \frac{t}{T} \right)^7 - 315 \left( \frac{t}{T} \right)^8 + 70 \left( \frac{t}{T} \right)^9 \right] \\
    v &= \frac{ds}{dt} = S \left[ 630 \frac{t^4}{T^5} - 2520 \frac{t^5}{T^6} + 3780 \frac{t^6}{T^7} - 2520 \frac{t^7}{T^8} + 630 \frac{t^8}{T^9} \right] \\
    a &= \frac{d^2s}{dt^2} = S \left[ 2520 \frac{t^3}{T^5} - 12600 \frac{t^4}{T^6} + 22680 \frac{t^5}{T^7} - 17640 \frac{t^6}{T^8} + 5040 \frac{t^7}{T^9} \right] \\
    j &= \frac{d^3s}{dt^3} = S \left[ 7560 \frac{t^2}{T^5} - 50400 \frac{t^3}{T^6} + 113400 \frac{t^4}{T^7} - 105840 \frac{t^5}{T^8} + 35280 \frac{t^6}{T^9} \right]
\end{align*}

An example of position, velocity, acceleration, and jerk profiles for unitary move in one second with a \textit{9}^{\text{th}}-Order Polynomial motion profile is shown in Fig. 2-15.
2.7 Comparison of Motion Profiles

A comparison in terms of position, velocity, acceleration, and jerk for an index move with Trapezoidal, Cubic, Simple-Harmonic Move (SHM), Cycloidal, Modsine, 5th-Order Polynomial, 7th-Order Polynomial, and 9th-Order Polynomial motion profiles are shown in Fig. 2-16. As shown in this figure, slight variations in position profiles yield significant variation in velocity, acceleration, and jerk profiles. As aforementioned, motion profiles with acceleration different than zero at the beginning and/or end of the move should be avoided in industrial applications. The motion profiles with zero acceleration rate at the beginning and end of the move and without discontinuities or abrupt changes in acceleration are recommended for industrial applications because profiles with these characteristics provide smooth motion and reduce vibration, noise, and
stress on mechanical and electrical components. Trapezoidal, parabolic, and sine profiles yield infinite jerk components and therefore should be avoided, while the other motion profiles in Fig. 2-16 would be recommended.

Fig. 2-16 - Comparison of motion profiles in terms of (a) position, (b) velocity, (c) acceleration, and (d) jerk profiles.

A comparison of the motion profiles shown in Fig. 2-16 is given in Table 2-1 in terms of peak velocity, peak acceleration, and peak jerk. This comparison also identifies profiles with zero acceleration and jerk at both ends of the motion profile. After analyzing the machine functional-specification, characteristics of the process, and system limitations, this table can be used to select one or more types of motion profiles appropriate for the machine under investigation.
Table 2-1 – Comparison between several motion profiles for an index move of a unitary displacement in 1 second.

<table>
<thead>
<tr>
<th>Profile</th>
<th>Peak Velocity</th>
<th>Peak Acceleration</th>
<th>Peak Jerk</th>
<th>Zero acceleration at boundaries?</th>
<th>Zero jerk at boundaries?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trapezoidal</td>
<td>1.5</td>
<td>4.5</td>
<td>∞</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Cubic</td>
<td>1.5</td>
<td>6</td>
<td>∞</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SHM</td>
<td>1.57</td>
<td>4.93</td>
<td>∞</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Cycloidal</td>
<td>2</td>
<td>6.28</td>
<td>39.5</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>ModSine</td>
<td>1.76</td>
<td>5.53</td>
<td>69.5</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>5th-Order Polynomial</td>
<td>1.875</td>
<td>5.76</td>
<td>60</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>7th-Order Polynomial</td>
<td>2.187</td>
<td>7.51</td>
<td>-52.5</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>9th-Order Polynomial</td>
<td>2.46</td>
<td>9.33</td>
<td>51.4</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

2.8 Motion Profiles with Slope

The equations provided in Section 2.6 are for simple index moves where the initial and final velocity are zero. However, many applications require complex motion profiles constituted by several segments. If these segments are connected by dwells, then the equations provided in Section 2.6 are appropriate to generate such complex motion profiles. However, if this complex motion profile consists of segments that need to merge with each other at various initial and/or final velocities and various initial and/or final accelerations, then a new set of equations that take into account the initial and final velocities and accelerations different than zero are needed. This new set or equations is used to properly merge motion profile segments and avoid abrupt transitions between segments, as described in the next section.
As an example, the equations that calculate the position $s$, velocity $v$, acceleration $a$ and jerk $j$ for a 5$^{th}$-order polynomial profile for any given start position $s_0$, final position $s_f$, start velocity $v_0$, final velocity $v_f$, start acceleration $a_0$, and final acceleration $a_f$ is shown below.

$$s = C_0 + C_1 \left( \frac{t}{T} \right) + C_2 \left( \frac{t}{T} \right)^2 + C_3 \left( \frac{t}{T} \right)^3 + C_4 \left( \frac{t}{T} \right)^4 + C_5 \left( \frac{t}{T} \right)^5$$

$$v = \frac{ds}{dt} = \frac{1}{T} \left[ C_1 + 2C_2 \left( \frac{t}{T} \right) + 3C_3 \left( \frac{t}{T} \right)^2 + 4C_4 \left( \frac{t}{T} \right)^3 + 5C_5 \left( \frac{t}{T} \right)^4 \right]$$

$$a = \frac{d^2 s}{dt^2} = \frac{1}{T^2} \left[ 2C_2 + 6C_3 \left( \frac{t}{T} \right)^2 + 12C_4 \left( \frac{t}{T} \right)^3 + 20C_5 \left( \frac{t}{T} \right)^4 \right]$$

$$j = \frac{d^3 s}{dt^3} = \frac{1}{T^3} \left[ 6C_3 + 24C_4 \left( \frac{t}{T} \right) + 60C_5 \left( \frac{t}{T} \right)^2 \right].$$

Where:

$$C_0 = s_0$$
$$C_1 = v_0t$$
$$C_2 = \frac{a_0T^2}{2}$$
$$C_3 = s_f - C_0 - C_1 - C_2 - C_4 - C_5$$
$$C_4 = -a_fT^2 + 7v_fT - 15s_f + 15C_0 + 8C_1 + 3C_2$$
$$C_5 = \frac{a_fT^2}{2} - 3v_fT + 6s_f - 6C_0 - 3C_1 - C_2.$$

As an example, these equations applied to a motion profile with $s_0 = 3$, $s_f = 7$, $v_0 = -10$, $v_f = -5$, $a_0 = -30$, and $a_f = 15$, yield the profile shown in Fig. 2-17.
Fig. 2-17 – Example of position, velocity, acceleration, and jerk for a 5th-order polynomial profile with \( s_0 = 3, \ s_f = 7, \ v_0 = -10, \ v_f = -5, \ a_0 = -30, \) and \( a_f = 15 \)

### 2.9 Multiple Profiles

In many applications, a single index move as described in Section 2.6 is not enough to perform the tasks in a machine cycle. In this case, more complex moves consisting of various segments are necessary. However, these segments need to be properly merged to avoid discontinuities in position, velocity, or acceleration command signals. The main consideration when merging multiple segments to build a complex motion profile is that the next segment needs to start at the same position, velocity, and acceleration as the final position, velocity, and acceleration of the previous segment. The motion profile defined by (11) and (12) can be used to properly merge segments and build complex motion profiles. An example of complex motion profile is shown in Fig. 2-18. In this figure, two segments were merged to build a more complex trajectory. The
parameters used to build this motion profile is shown in Table 2-2, which yield a smooth transition from the first segment to the second segment. As show in this table, the final position, velocity, and acceleration of the first segment is the same as the initial position, velocity, and acceleration of the second segment.

Table 2-2 - Parameters of a sample multiple profile

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Segment 1</th>
<th>Segment 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>$s_f$</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>$v_0$</td>
<td>-10</td>
<td>10</td>
</tr>
<tr>
<td>$v_f$</td>
<td>10</td>
<td>-10</td>
</tr>
<tr>
<td>$a_0$</td>
<td>-30</td>
<td>-5</td>
</tr>
<tr>
<td>$a_f$</td>
<td>-5</td>
<td>-30</td>
</tr>
</tbody>
</table>

Fig. 2-18 - Example of position, velocity, acceleration, and jerk for a motion profile with multiple segments.
2.10 Summary

In this chapter, several types of motion profiles typically used in industrial were defined. A briefly description and the equations for position, velocity, acceleration, and jerk for each motion profile were provide.

Motion profiles are extensively used as the position reference signal applied to the control systems controlling each motor in industrials machines. Thus, the proper selection and design of the motion profiles heavily impact the behavior and performance of machines.

The types and selection of motion profiles, asymmetric motion profiles, and merging of multiple segments are some of the concepts used in the mechatronic design process presented in the next chapter to properly design motion profiles that mitigate vibration, stress on mechanical and electronic components, noise, and premature machine-wear.
CHAPTER 3

Mechatronic Design Process

Industrial machines with well-designed mechanical systems and well-designed and tuned control systems are necessary, but not enough conditions to reach high performance levels. Motion profiles properly designed via trajectory planning in addition to properly sized motors are the other two necessary conditions for machines to reach high performance levels. For this reason, a model-based mechatronic design process to properly design motion profiles through trajectory planning and then size the motors before building a prototype was proposed and demonstrated in this dissertation to improve position control accuracy of industrial machines, in addition to mitigate noise, vibration, and stress on mechanical components, motors, and drives. Trajectory planning is the method used to design motion profiles that are used as the position reference of the control systems controlling the motion of motors and loads.

This proposed mechatronic design process can be applied to single-axis systems and multi-axis coordinated systems. This proposed mechatronic design process for motor selection and trajectory planning was demonstrated and validated via simulation and experimental results for three systems: a two-inertia system which is a single-axis system, a Cartesian two-axis planar robot (H-Bot) which is a multi-axis coordinated system, and a self-balancing transporter. Although the proposed mechatronic design process was demonstrated for three particular systems, it can be applied to any industrial machine.

The proposed mechatronic design method is based on model-based design and trajectory planning to size motors. Model-based design has been of fundamental
importance for industrial development during the past few decades [1-8] and has gained even more interest from the scientific and industrial communities as the needs for energy efficiency, reliability, flexibility, and accuracy in machines have increased to satisfy a market that is demanding higher productivity at reduced costs in a sustainable manner. This results in tighter, more complex, and challenging specifications to be achieved in machine design [1, 2]. However, machine design-engineers are proficient in the design of elements that belongs to their domain of expertise (mechanical, electrical or control), but not necessarily in the design of elements that require expertise in multiple domains. Trajectory planning and motor sizing are two elements in machine design that require expertise in multiple domains to be integrated. Thus, in order to support engineers to improve machine designs, this dissertation provides a method for trajectory planning and motor sizing in an integrated manner, which is a methodology absent in the literature and of fundamental importance to design machines with the potential of achieving high-performance levels.

Trajectory planning is not only critical for the motor selection process, but also for the overall performance of the system. However, machines with motion-control systems often use jerky command motion profiles that unnecessarily stress the machine and motors, produce vibrations and noise, and wear mechanical components, resulting in poor machine performance and reduced life [70-73]. Jerky motion profiles also yield higher position and velocity following errors, which compromises the quality of the products from the machines. The use of inadequate command motion profiles is due to inappropriate motion profile selection and/or poor trajectory planning. As described in Chapter 2, trapezoidal profiles are typical in industrial applications due to easy implementation, despite producing jerky motion. Meanwhile, the process of building a
more complex and efficient profile is more involved; requiring more programming skills and knowledge of mechanical and control engineering. This results in a default choice for the easy-to-program motion profiles, although less-efficient. Similar performance issues can also occur in multi-axis coordinated systems tracing some types of trajectories in the Cartesian space, which can be particularly difficult to follow accurately, e.g., a robot tracing a square shape in the Cartesian space. In this particular example, large positioning error and peak torques can occur even with a well-tuned control systems and well-designed mechanical systems. The difficulty with square shapes, for example, resides in accurately tracing the corners, which can result in imperfections in the final product. This issue can be mitigated with trajectory planning, which is a method to calculate the time-domain position-reference signal for each axis of a mechanism to reduce position-following error, jerk, and peak torques. This trajectory planning method takes into account the geometry and inverse kinematics of the system, and the desired move path. The developed trajectory planning approach to mitigate undesirable positioning errors is described next and demonstrated with simulation results.

In the proposed mechatronic design method, the inverse kinematics of the system in conjunction with trajectory planning are used to compute the motion profiles for each motor. These motion profiles are applied to the inverse kinetic model of the system to compute the required motor torque and speed curves. These curves are then used to select candidate motors. After selecting candidate motors, a closed-loop simulation is used to validate the solution and stability of the system. This approach allows simulating not only the mechanical system, but also the control system approach in an integrated manner. This approach can also help uncover issues before building prototypes, in addition of narrowing the gap between the mechanical and control designs.
The proposed mechatronic design process is shown in Fig. 3-1 and it was published in [59]. This model-based design process allows the designer to analyze the system before building it, identify possible design mistakes, and implement the necessary correction still in the design stage avoiding expensive redesign costs in a more advanced stage of the prototyping process. This procedure will be demonstrated next by sizing the motors for a single axis system (two-inertia system), a two-axis coordinated motion system (Cartesian two-axis planar robot – H-Bot), and an unstable system (self-balancing transporter). A multi-axis coordinated system is defined here as two or more mechanisms mechanically linked, where each mechanism is individually controlled by a motor or actuator and the motion of each motor affects the process or motion of the other motor in the coordinated system. Multi-axis coordinated systems require synchronization of position reference signals applied to each motor in order to result in the desired motion in the Cartesian space. Typical examples of multi-axis coordinated system are robots. Industrial machines consists of single axes, multi-axis coordinated systems, or a combination of both. Thus, the mechatronic design process shown in Fig. 3-1 is demonstrated next to all cases in industrial machines.
Fig. 3-1 - Proposed motor selection process adapter from [59]
3.1 Two-Inertia System

Many types of mechanisms controlled by motors in industrial machines are or can be simplified to a two-inertia system. In a two-inertia system, one of the inertias is the rotor inertia while the other one is the load inertia. Both inertias are connected by a compliant link modeled as a torsional spring. For highly rigid systems, the compliance may be ignored. The proposed mechatronic design process in Fig. 3-1 is demonstrated next for a two-inertia system.

3.1.1 System Requirements

The system requirements define the minimum functionalities and level of performance that the system needs to achieve. This may include the envelope (workspace) of motion, trajectory, positioning accuracy, machine cycle time, pay-load, and maximum allowable level of vibration and noise. The system requirements are, in general, contained in the machine functional specifications.

The two-inertia system consists of a motor driving a constant load inertia coupled by a compliant link modeled as a torsional spring $k$ as shown in Fig. 3-2. The two independent variables in this system are the motor angular velocity $\dot{\theta}_m$ and the load angular velocity $\dot{\theta}_l$ as shown in Fig. 3-2.
The system requirements for this demonstration of the mechatronic design process with a two-inertia system were chosen as typical values for motion control applications as follows:

- Incremental move distance: 0.1 revolution
- Incremental move time: 100 milliseconds
- Dwell time: 200 milliseconds
- Load inertia: $2.5 \times 10^{-3}$ kg-m^2

3.1.2 End-point Trajectory

The end-point trajectory is defined here as the motion of the end-effector or load. Depending on the type of mechanism, the end-point trajectory can be defined in reference to different points on a machine, e.g., at the end-effector for robots, at the platen for presses, or at the load on conveyors.

In this mechatronic design process, the worst-case trajectory of the end-point or end-effector that yields the highest torque and speed requirements of the motors need to be identified. This worst-case trajectory can be identified from the functional...
specifications of the machine. The worst-case trajectory is, in general, defined as a function of the application requirements and not in terms of the maximum capability of the mechanism. This may allow selecting smaller motor sizes. It is essential to define the end-point trajectory first, because the required torque vs. speed curve used to size the motor heavily depends on it.

If the system needs to perform multiple types of trajectories, designing motors and drives for the worst case trajectory is enough to account for all cases. If it is not obvious which trajectory is the worst case, either all or the most likely worst-cases must be evaluated to identify which trajectories require the highest motor and servo-drive utilization. In some cases, the mechanism can perform an infinite number of different trajectories. A typical case is a robot in a pick-and-place application with a vision-system that locates products in random orientations and locations to be picked up by the robot. In this case, either the longest or the fastest pick-and-place move are in general good candidates to be used as the worst-case trajectory in the motor selection process.

The end-point trajectory for this two-inertia system was defined based on the system requirement as an index move of 0.1 motor revolution in 100 milliseconds with a dwell time of 200 milliseconds.

3.1.3 Trajectory Planning

Trajectory planning is the computation of motion profiles used as the position reference signal to command motion of each actuators in automatic machines, e.g., packaging machines, machine tools, assembly machines, metal-forming machines, and industrial robots. Each motion profile needs to be designed to avoid or reduce mechanical vibration, stress on mechanical and electronic components, electrical and
audible noise, stress on motors and actuators, as well as to reduce overshoot response and excessive position-following error during motion.

The information necessary to compute the trajectory planning is the end-point trajectory and the inverse kinematics of the mechanism. Since kinetic models are not necessary at this point, the trajectory planning can be defined at the early stages of the design from sketches with the main dimensions of the moving mechanism. Thus, there is no need for information about masses and moments of inertia of the system for trajectory planning.

In order to obtain a balance between peak speed and motion smoothness, a 5th-order polynomial profile was selected to perform the required end-point trajectory. The resulting motion profile calculated from (8) is shown in Fig. 3-3.

If the peak speed is too high for the motor under consideration, then a motion profile with lower peak speed can be selected from Table 2-1.

Fig. 3-3 - Trajectory planning for the two-inertia system
3.1.4 Inverse Kinematics

The inverse kinematics of the mechanism converts the end-point trajectory described in the Cartesian space into the motion profiles that control the motors. In the case of a mechanism with redundant degrees of freedom, e.g., a 7-degree-of-freedom manipulator, infinite possibilities of motion profiles of the motors can be defined to yield the same end-point trajectory [74, 75]. In this case, one of the possible inverse kinematic solutions needs to be chosen [75] to perform the motor selection.

The two-inertia system does not require inverse kinematics in this mechatronic design process.

3.1.5 Kinetic and Inverse Kinetic Model

The kinetic and inverse kinetic models of the mechanism are used to estimate the amount of torque required to control the system. This kinetic and inverse kinetic model should also include the rotor inertia of the motor being investigated as a candidate motor since the rotor inertia may require significant acceleration torque in comparison to the load torque.

The kinetic and inverse kinetic models can be derived via methods such as D’Alembert’s Principle, Newton-Euler, or Lagrange [76, 77]. Many machines may have complex mechanisms and also parts with complex geometries. In these cases, the derivation of the kinetic equations via Newton-Euler and Lagrange methods becomes a time-consuming task. Some alternatives are as follows. For parts with complex geometry, the inertia can be obtained from 3D drawings when available. For complex mechanisms, SimMechanics™ from MathWorks can be used to develop the kinetic model of the system [78]. If the mechanism was designed in a 3D Software package, it may be
possible to automatically generate the SimMechanics model via SimMechanics™ Link [79].

The kinetic model for the two-inertia system was derived from the diagram shown in Fig. 3-2. In this diagram, $T_m$ is the motor torque, $T_l$ is the load torque, $k$ is the stiffness or spring constant of the rod connecting the motor to the load, $b$ is the damping factor, $\dot{\theta}_m$ is the motor angular velocity, and $\dot{\theta}_l$ is the load angular velocity.

The sum of torques from the motor side of the free-body diagram in Fig. 3-2 yields the first equation of motion:

$$J_m \ddot{\theta}_m + k(\theta_m - \theta_l) + b(\dot{\theta}_m - \dot{\theta}_l) = T_m$$

$$\dot{\theta}_m = \frac{1}{J_m} [Tm - k(\theta_m - \theta_l) - b(\dot{\theta}_m - \dot{\theta}_l)]$$

(13)

While the sum of torques from the load side yields the second equation of motion:

$$J_l \ddot{\theta}_l - k(\theta_m - \theta_l) - b(\dot{\theta}_m - \dot{\theta}_l) = 0$$

$$\dot{\theta}_l = \frac{1}{J_l} [k(\theta_m - \theta_l) + b(\dot{\theta}_m - \dot{\theta}_l)]$$

(14)

The direct kinetic model defined by (13) and (14) can be implemented in Simulink as shown in Fig. 3-4.
Based on (13) and (14), the inverse kinetic model can be obtained and modeled in Simulink as shown in Fig. 3-5.
3.1.6 Open-Loop Simulation

Open-loop simulation is used to estimate the torque required from the motor to drive the system and its own rotor inertia according to the motion profile defined by the Trajectory Planning. The open-loop simulation is performed for each motor under consideration.

The open-loop simulation consists of the kinetic model fed by the inverse kinetic model which is commanded by the position reference signal from the Trajectory Planning. Using this configuration, no tuning is necessary while performing the open loop simulation for each motor being tested to identify candidate motors that can provide enough torque and speed to drive the load. Tuning is avoided by using the inverse kinetic model in place of a closed-loop control system. This configuration is necessary for multi-axis coordinated systems (e.g. H-Bot). Otherwise, the inverse kinetic only is sufficient to calculate the required torque to drive the system. In order to use the same open-loop configuration for all system, the single axes systems also employ this open-loop configuration in this dissertation.

The open-loop model for this two-inertia system is shown in Fig. 3-6 and it consists of the inverse kinetic model shown in Fig. 3-5 feeding the direct kinetic model shown in Fig. 3-4. The additional summation block in this figure, although not always necessary, can help reduce the position error and more accurately estimate the motor torque for multi-axis coordinated systems. As aforementioned, this configuration is, in general, necessary for multi-axis coordinated systems, but it is also been used with single axis systems in this dissertation in order to apply the same solution to all systems. The calculated motor torque with the inverse kinetic model only or with the configuration
shown in Fig. 3-6 for the two-inertia system is the same. The “Position Command” block in Fig. 3-6 generates the motion profile shown in Fig. 3-3.

Next, the open-loop simulation will be used to compute the required torque and speed curves to drive the system. The torque and speed curves will then be used to select candidate motors. If a motor database is available, this search for candidate motors can be automated.

3.1.7 Torque vs. Speed Requirement

The open-loop simulation is used to determine the torque vs. speed curve required by each motor of the system to follow the motion profile designed by trajectory planning for the worst case scenario of the end-point trajectory. The selection of candidate motors in the next step relies on this estimated torque vs. speed curve.
The torque vs. speed curve of the two-inertia system was obtained from the simulation of the open-loop model in Fig. 3-6 for the motion profile shown in Fig. 3-3 and parameters listed in Table 3-1. More details about the identification of these parameters is given later in Section 4.1.6.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load inertia</td>
<td>$J_l$</td>
<td>$2.5 \times 10^{-3}$</td>
<td>kg-m$^2$</td>
</tr>
<tr>
<td>Motor inertia</td>
<td>$J_m$</td>
<td>$4.4 \times 10^{-5}$</td>
<td>kg-m$^2$</td>
</tr>
<tr>
<td>Total torsional stiffness</td>
<td>$k$</td>
<td>1281</td>
<td>Nm/rad</td>
</tr>
<tr>
<td>Viscous Damping</td>
<td>$b$</td>
<td>$2.5 \times 10^{-6}$</td>
<td>N-m-s</td>
</tr>
</tbody>
</table>

The resulting torque and speed for this two-inertia system curves are shown in Fig. 3-7. As shown in this figure, peak angular motor speed was 113 rpm while the peak torque was ±1.05Nm.

Fig. 3-7 - Torque and velocity profile for motor a of the simulated two-inertia system
The torque and velocity time-domain data shown in Fig. 3-7 can then be plotted as a torque vs. speed curve, as shown in Fig. 3-8, which is the typical format to represent the data used for motor sizing. There are two conditions that a motor must satisfy to qualify as a candidate motor to drive a system. One condition is that the peak torque vs. speed curve of the candidate motors encloses the required torque vs. speed curve of the system. The other condition to be satisfied is that the \textit{rms} torque at the \textit{rms} speed of the application resides below the continuous torque vs. speed curve of the motor. Otherwise, the motor will overheat and potentially be damaged. The \textit{rms} torque ($\tau_{rms}$) and the \textit{rms} velocity ($v_{rms}$) are calculated as follows:

\begin{align}
\tau_{rms} &= \sqrt{\frac{1}{T} \int_0^T \tau^2(t)dt} \\
v_{rms} &= \sqrt{\frac{1}{T} \int_0^T v^2(t)dt}
\end{align}

Where, $\tau$ is the instantaneous torque, $v$ is the instantaneous velocity, and $T$ is the time of a machine cycle. For the two-inertia system, the resulting \textit{rms} torque was calculated at 0.41 Nm and the \textit{rms} speed at 41.7 rpm, as shown by the solid dot in Fig. 3-8.
3.1.8 Identifying Candidate Motors

The candidate motors to drive a system are selected based on several conditions. These conditions also vary from application to application, and the same condition may hold different weight in the decision for a motor. For example, for low cost machine, the cost of the motor is more important than the motor efficiency, which might be the opposite for a high-efficiency machine.

As mentioned in the previous section, one of the conditions to select a motor is that the peak (also called intermediate) torque vs. speed curve of the candidate motor encloses the torque vs. speed curve of the system. The other condition is that the point defined by the \( \tau_{rms} \) at the \( v_{rms} \) needs to be located below the continuous torque vs. speed region of the motor. This allows the motor to operate below the maximum operating temperature.

Another parameter to account for during the selection of a motor is the inertia ratio between the load and motor. The inertia ratio is in general preferred to be kept low.
(typically less than 10:1 as a rule-of-thumb) to obtain higher system bandwidth. To keep the inertia ratio low, a gearbox or a transmission system may be necessary. Additional requirements such as cost, motor voltage, energy efficiency, motor size, and mounting orientation, as well as preferences for particular motor manufacturers will shorten the list of candidate motors.

The motor technology types is also part of the motor selection process. Torque density, rotor inertia, power, maximum speed, voltage, torque loses, frame type, operating temperature, are some of the characteristics that need to be taken into account while deciding upon a particular motor technology.

Common types of motor technologies used in industrial applications include induction motors, permanent magnet synchronous motors (PMSM), dc motors, and steppers [80]:

- **Induction motors** are used in general applications including pumps, conveyors, compressors, mixers, hoists, and cranes. This type of motor is low cost, robust, easy to maintain, and is available from fractions to hundreds of a horse-power. Induction motors can be powered directly from the power line, or via variable frequency drives (VFD) and servo drives [81, 82]. When powered via VFDs or servo drives, motors can be controlled from nearly zero to the rated speed and develop rated torque in this speed range, which is not possible when powered directly from the power line.

- **Permanent magnet synchronous motors** (PMSM) can have low rotor inertia and high torque density, which make them a suitable choice for motion-control applications, where fast transitions in speed, positioning accuracy, high bandwidth, and hold torque at zero speed are required. Typical applications for PMSM motors include packaging machines, web handling, cartoners, fillers, robots, and capping machines. DC motors
are relatively easy to model and can have simple speed control methods and develop high torque [80, 83]. This allows this type of motor to be used in a variety of applications, including mobile devices, mining equipment, and robots.

- **Stepper motors** are commanded in incremental steps. Each step yields a fixed angular displacement which can be in the forward or reverse direction. As long as no steps are missed, there is a synchronous relationship between the input command and the angular position of the motor shaft, which can yield an accurate open-loop positioning system. Thus, typical applications for stepper motors include printers, disk drives, and machine tools.

Since the two-inertia system described in this chapter characterizes a motion-control application, a PMSM was chosen as the motor technology. By comparing the required torque vs. speed curve in Fig. 3-8 to the torque speed curve of several PMSM, the motor shown in Table 3-2 was selected. If an electronic motor database of the motors under consideration is available, this selection process can be automated.

**Table 3-2 - Selected motor for the two-inertia system**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brand</td>
<td>-</td>
<td>Allen-Bradley</td>
<td>-</td>
</tr>
<tr>
<td>Model</td>
<td>-</td>
<td>MPL-B310P-M</td>
<td></td>
</tr>
<tr>
<td>Torque Constant</td>
<td>$K_t$</td>
<td>0.573</td>
<td>Nm/A</td>
</tr>
<tr>
<td>Peak Stall Current</td>
<td>$I_m$</td>
<td>1.7</td>
<td>A</td>
</tr>
<tr>
<td>Rated Voltage</td>
<td>$V$</td>
<td>460</td>
<td>V</td>
</tr>
<tr>
<td>Resistance</td>
<td>$R_a$</td>
<td>18.9</td>
<td>Ω</td>
</tr>
<tr>
<td>Inductance</td>
<td>$L_a$</td>
<td>92</td>
<td>mH</td>
</tr>
<tr>
<td>Rotor Inertia</td>
<td>$J_m$</td>
<td>4.4x10^{-5}</td>
<td>kg-m(^2)</td>
</tr>
<tr>
<td>Back-EMF Constant</td>
<td>$K_e$</td>
<td>0.936</td>
<td>V/rad/s</td>
</tr>
<tr>
<td>Viscous Damping</td>
<td>$b$</td>
<td>2.5x10^{-6}</td>
<td>N-m-s</td>
</tr>
<tr>
<td>Rated Speed</td>
<td>$\omega_{max}$</td>
<td>5000</td>
<td>rpm</td>
</tr>
</tbody>
</table>
The torque vs. speed curve of the selected motor (see Table 3-2) overlapped with the required torque vs. speed curve is shown in Fig. 3-9. As shown in this figure, there are two torque vs. speed regions. The continuous torque vs. speed region is where the motor can operate continuously while the intermittent torque vs. speed region is where the motor can operate for short periods of time. The amount of time in this intermediate region is dictated by the motor temperature.

Fig. 3-9 - Torque vs. speed curve of the selected motor overlapped with the required torque vs. speed curve obtained from the open-loop simulation of the two-inertia system.

3.1.9 Augmenting the System

Once a motor is selected, additional motor properties and the complete model of the motor can be added to the simulation model. Any flexible couplings or gearing contained in the system should also be included in the model. If a gearbox or a transmission mechanism is added to the system to reduce the inertia ratio between the load and motor or better balance the available torque and speed, not only the gear ratio, but also the losses and inertia of this additional component, must be included in the simulation of the system.
For the two-inertia system, the motor model can be added to the simulation system as shown next in Section 3.1.12.

3.1.10 Control System Design

A typical control system approach for motor control consists of a cascaded control system with an inner velocity PI (Proportional-Integral) control loop and an outer position PI control loop, as shown in Fig. 3-10. This configuration is also typically found in industrial drives [62]. A feedforward velocity loop (FFv) can be used to reduce the position-following error, while the acceleration feedforward loop (FFa) can be used to reduce the velocity following error. The velocity loop feeds the inner current loop that controls the current delivered to the motor to move according to the commanded position profile. Two or more filters can be placed between the velocity and current loops. A low-pass filter (LPF) can be used to reduce high frequency noise feeding the current loop while one or more notch filters (NF) can be used to attenuate resonant frequencies.

Fig. 3-10 - Cascade PI control for the two-inertia system.
3.1.11 Tuning

Several tuning methods can be used to tune the cascade control loop shown in Fig. 3-29 [62]. When the motor is powered by industrial servo drives, automatic tuning features can be used to tune the control loops if available. An alternative is to manually tune the controls loops starting with the PI gains from the velocity loop and then the PI gains from the position loop. This manual tuning process extensively used with industrial machines consists on the following main steps:

- Command index moves with acceleration rates equivalent to the acceleration rates during normal machine operation.
- Reset the PI gains from the control loop to bypass the position loop, and set the velocity feedforward gain to unitary gain. This will apply a velocity reference signal to the velocity loop control without interference from the position loop (see Fig. 2-1). Alternatively, the drive can be set to velocity mode, which bypasses the position loop as well.
- Increase the proportional gain from the PI velocity loop until the system become unstable. Then, cut this gain to half. Add integral gain to the velocity loop if the velocity following error needs to be reduced. The unstable condition is identified while monitoring the velocity error or the motor current signal. When the oscillations in either or both signals start to increase asymptotically, the unstable condition was identified.
- Increase the proportional gain from the position loop until the system become unstable. Then, cut this gain to half. Add integral gain to the position loop if the position-following error needs to be reduced.
• Set the low-pass filter (LPF) to attenuate high frequency noise if needed and set one or more notch filters (NF) to attenuate resonant frequencies if needed during any time of this tuning process.

The PI gains of the position and velocity loops are tuned to make the system robust to load disturbances, while the velocity feedforward and acceleration feedforward gains can be set to reduce position and velocity following errors, respectively.

If frequency response tools are available in the drive that powers the motor, the control loop gains can be set to reach a particular gain margin and phase margin [84-88].

3.1.12 Closed-Loop Simulation

The closed-loop simulation contains the motion position profile defined via trajectory planning, the kinetic model of the system, the motor model, and the control system. The resulting position feedback from the kinetic model can be compared to the position command to evaluate the position-following error as shown in Fig. 3-11. Similarly, velocity following error can be analyzed by comparing the velocity command and the motor velocity feedback as shown in this figure as well. This closed-loop simulation is used to verify if the system requirements are met.

The closed-loop model for the two-inertia system is shown in Fig. 3-11. The motor model used in this figure is as follows:

\[
v_a = R_a i_a + L_a i_a + K_e \omega_m
\]  

(16)

Where, \(v_a\) is the motor voltage, \(R_a\) is the motor resistance, \(L_a\) is the motor inductance, \(K_e\) is the back-emf constant, \(i_a\) is the motor current, and \(\omega_m\) is the motor angular velocity.
Fig. 3-11 - Closed-loop simulation of the two-inertia system

3.1.13 Final Torque vs. Speed Requirements

Based on the closed-loop simulation with the system properly tuned as defined in Section 3.1.11, the final required torque vs. speed curve of the system is calculated and compared to the torque vs. speed curve of the motor. If the following requirements are met, the selected motor is a valid solution:

- The required torque vs. speed curve must reside within the peak torque vs. speed curve of the motor,
- The $rms$ torque at the $rms$ speed must reside within the continuous torque vs. speed curve of the motor
- The system needs to develop a stable behavior.

If any of these requirements are not met with the selected motor, iterations are needed, i.e., different motors and/or gearboxes need to be evaluated until the requirements listed above are met. However, in applications where the torque vs. speed
requirement is too high, a valid motor/gear box solution may not exist. In this case, the mechanism or the application requirements need to be reevaluated.

The torque and speed curves obtained from the closed-loop simulation of the two-inertia system are shown in Fig. 3-12.

![Fig. 3-12 – Torque and speed profiles from the closed-loop simulation.](image)

The final torque vs. speed curve requirement of the two-inertia system shown in Fig. 3-11 is shown in Fig. 3-13. These results were obtained after properly tuning the control system of the two-inertia system. The torque vs. speed curve obtained from closed-loop simulation shown in Fig. 3-13 is similar to the open loop simulation shown in Fig. 3-9.
3.2 Cartesian Two-axis Planar Robot - H-Bot

The proposed mechatronic design process was also demonstrated for a multi-axis coordinated system by selecting the motors of a two-axis Cartesian planar robot typically called H-Bot, which consists of two motors, a timing belt, and two rails mounted perpendicular to each other. A typical H-Bot configuration is shown in Fig. 3-14. An H-bot is a two degree of freedom robot extensively used in industry in applications such as pick-and-place, sorting, gluing, and inspection. This type of robot is particularly attractive for machine builders due to the relatively ease of manufacturing.

Fig. 3-14 – Typical configuration of an H-Bot
3.2.1 System Requirements

The system requirements for the H-Bot consist in performing the diamond shape shown in Fig. 3-15 with diagonal of 0.4 meters in 2 seconds with a maximum position-following error of 1mm. As described in Section 3.1.1, the system requirements are obtained from the machine functional specification. Additionally, the H-Bot needs to be mounted vertically.

![Diagram of diamond shape](image)

Fig. 3-15 - Definitions of the diamond shape

3.2.2 End-point Trajectory

Since the system requirements defined a single task, the diamond path with a diagonal of 0.4 meters traced in 2 seconds was defined as the worst case scenario of the end-point trajectory for this system.

3.2.3 Trajectory Planning

Trajectory planning is particularly important to reduce position tracking errors in multi-axis coordinated system, such as machine tools, while performing certain types of profiles in the Cartesian space [89]. Trajectories in the Cartesian space that require particular attention while designing the motion profiles for each motors via trajectory
planning include those with sharp corners or high dynamics. A typical example of such profile is a diamond shape as shown in Fig. 3-15. The difficulty with diamond shapes resides in accurately tracing the corners without overshoots or distortions that can cause imperfections to the final product.

The proposed trajectory planning approach that can be applied to generic types of industrial machines with multi-axis coordinate systems is described and demonstrated next with an H-Bot tracing a diamond shape.

After defining the end-point trajectory, the proposed trajectory planning approach for multi-axis coordinated systems consists of the following steps.

3.2.3.1 Identify points in the end-point trajectory in which the velocity changes direction

These are the points in which at least one Cartesian axis from the same coordinated motion system changes the direction of motion. In the case of the diamond shape shown in Fig. 3, this occurs at the corners. At corners A and C, the Z-axis changes direction, while at corners B and D, the X-axis changes direction.

3.2.3.2 Define the master command.

The master command or master reference is used to synchronize the motion of the axes in the multi-axis coordinated system. In industrial applications, this master reference is, in general, generated from a virtual servo axis, which is a servo axis that only exists in the machine code without any hardware (e.g., drives, motors) associated with it. The master reference is a signal to which the motion profile controlling the physical axes in a multi-axis coordinated system are synchronized to. Thus, instead of generating a time-domain motion profile for each axis in a multi-axis coordinated system, the motion
profiles are generated in respect to this master reference signal. This allows to synchronize the motion of all axes as shown next.

The master reference signal is also used to synchronize independent axes in a machine, i.e., motors that are not part of coordinated systems. In this case, the master reference signal is set to a constant speed value that corresponds to the desired machine production rate. This allows to easily change the production rate by simply changing the speed of the virtual axis that generates the master reference signal. Since all independent servo axes are linked (synchronized) to this master reference signal, all axes will move according to the production rate defined by the master reference signal in a synchronized manner. This practice of setting the master reference signal to a constant speed is appropriate for synchronizing independent axes in a machine, but it is not necessarily appropriate for multi-axis coordinated system as demonstrated in the next section, because it can cause high peak torques, vibration, noise, etc.

Thus, instead of setting the master reference signal to a constant speed, a motion profile can be designed for the master reference signal to reduce vibration, mechanical and electrical stresses, and noise on the physical axes. The master reference signal can be defined as a simple index move between each two consecutive points of the end-point trajectory where the Cartesian axes change the direction of motion. The type of motion profile for these index moves of the master reference signal is chosen by taking into account the considerations in Section 2.7.

The polynomial profile shown in Fig. 3-16 is in general an appropriate choice for the master reference signal because of the zero acceleration at the beginning and end of the move which yields smoother motion for physical systems. Although various types of profiles could be used to build the master reference, the 5th-order polynomial profile
yields a good tradeoff between smoothness and peak velocity. The master reference will consist of segments located between every two consecutive points of the end-effector trajectory that has a change in polarity on the velocity profile. Thus, the master position command \( S_M \) can be defined for each one of these segments as a 5\(^{th}\) order polynomial profile as follows:

\[
S_M = 10\left(\frac{t}{T}\right)^3 - 15\left(\frac{t}{T}\right)^4 + 6\left(\frac{t}{T}\right)^5
\]  

(17)

Where, \( T \) is the desired time to complete each segment and \( t \) is the instantaneous time. Thus, for the end-point trajectory in Fig. 3-15, the master position command for the H-Bot contains four segments \( (\overline{AB}, \overline{BC}, \overline{CD}, \overline{DA}) \) located between each corner of the diamond shape as shown in Fig. 3-16. The master position command has unitary position increments from segment to segment. These unitary increments continue until it completes the entire end-point trajectory. Then, the master command can either continue the unitary increment into the next machine cycle or be reset back to zero. The duration of each segment is given by \( T \). The desired time \( T \) depends on the time to perform each segment, which may not be the same for all segments. In Fig. 3-16, the master reference signal was defined with \( T \) set to 0.5 sec.

![Fig. 3-16 - Master command for H-bot.](image-url)
When the master reference signal in Fig. 3-16 pass through the point where speed and acceleration are zero, the physical axes are also commanded to zero speed. This is key for accurate positioning and smooth motion at the corners of the end-point trajectory.

The master reference signal will be used with the geometric equations of the end-effector trajectory to calculate the command position of the motors, as described next.

3.2.3.3 Define geometric equations of the end-point trajectory

The geometric equations define the end-point trajectory in the Cartesian space as a function of the master reference signal. The number of geometric equations is given by the number of coordinates to describe the motion of the end-effector. In the case of the H-Bot, the end-point trajectory is in the XZ plane. Thus, there are two geometric equations: one to describe the motion in X and one in Z. The geometric equations of the H-Bot performing the diamond shape shown in Fig. 3-15 are as follows:

\[
X = \begin{cases} 
\frac{((X_{\text{MAX}} - X_{\text{MIN}})/2)(1 - S_M)}{2} + X_{\text{MIN}} & AB \\
\frac{((X_{\text{MAX}} - X_{\text{MIN}})/2)(-1 + S_M)}{2} + X_{\text{MIN}} & BC \\
\frac{((X_{\text{MAX}} - X_{\text{MIN}})/2)(-1 + S_M)}{2} + X_{\text{MIN}} & CD \\
\frac{((X_{\text{MAX}} - X_{\text{MIN}})/2)(5 - S_M)}{2} + X_{\text{MIN}} & DA 
\end{cases}
\]

\[
Z = \begin{cases} 
\frac{((Z_{\text{MAX}} - Z_{\text{MIN}})/2)S_M}{2} + Z_{\text{MIN}} & AB, BC \\
\frac{((Z_{\text{MAX}} - Z_{\text{MIN}})/2)(4 - S_M)}{2} + Z_{\text{MIN}} & CD, DA 
\end{cases}
\]

Where, \(X_{\text{MAX}}, X_{\text{MIN}}, Z_{\text{MAX}}\) and \(Z_{\text{MIN}}\) are the maximum and minimum displacements in the X and Z directions, respectively. The resulting profiles of \(X\) and \(Z\) obtained from (18) for the H-Bot tracing a diamond shape with diagonal of 0.4 meters in 2 seconds is shown in Fig. 3-17. In this example, \(X_{\text{MIN}}\) and \(Z_{\text{MIN}}\) were set to zero, \(X_{\text{MAX}}\) and \(Z_{\text{MAX}}\) were 0.4m as defined in Section 3.2.3.2, and the master command \(S_M\) was defined by (17) (see Fig. 3-16 also).
Fig. 3-17 - Diamond profile split in X and Z axes according to presented trajectory planning.

For comparison purposes, if the master command is set to a constant speed instead, as in typical trajectory-planning approaches, the resulting profiles for X and Y are triangular, as shown in Fig. 3-18. Consequently, high acceleration rates and jerk occur every 0.5 seconds which yields spikes of acceleration torque that causes undesirable effects such as abrupt motion, vibration, noise, and stress on mechanical and electrical components. This effect is shown later in Section 3.2.13. Smoother motion profiles were obtained with the proposed trajectory planning method.

Fig. 3-18 - Diamond profile split in X and Z axes for a constant speed master command.

The values of X and Z obtained from (18) with the master command $S_M$ defined in (17) can then be applied to the inverse kinematics of the system to compute the motor motion profiles, as described next.
3.2.4 Inverse Kinematics

The diagram of the two-axis H-Bot is shown in Fig. 3-14. In this system, if one motor stays stationary while the other one rotates, the end effector moves diagonally. If both motors rotate at the same speed in the same direction, the end effector moves either left or right, depending on the direction that the motors are rotating. If both motors rotate at the same speed in opposite directions, the end effector moves either up or down, depending on the direction that the motors are rotating.

The inverse kinematics equations of the H-Bot are as follows.

\[
\begin{align*}
\theta_1 &= \frac{Z - X}{r} \\
\theta_2 &= \frac{-Z - X}{r}
\end{align*}
\] (19)

Where, \(\theta_1\) is the angular position of motor shaft \(M_1\), \(\theta_2\) is the angular position of motor shaft \(M_2\), \(X\) and \(Z\) define the position of the end effector in the Cartesian space given by the trajectory planning in (18), and \(r\) is the radius of the driving pulley connected to each motor as shown in Fig. 3-14. The resulting profile from the inverse kinematics in (19) with \(X\) and \(Z\) computed from the trajectory planning defined in (17) and (18) is shown in Fig. 3-19.

![Graph showing \(\theta_1\) and \(\theta_2\) profiles](image_url)

Fig. 3-19 - Profile for \(\theta_1\) and \(\theta_2\) obtained from the inverse kinematics of the H-bot
3.2.5 Kinetic and Inverse Kinetic Model

The kinetic model of the H-Bot was developed with the following assumptions: vertically mounted, massless belt, rigid bodies, frictionless joints, and inertialess idler-pulley. Lagrange’s method [77] were used to obtain the equations of motion of the system. Lagrange’s Equations are as follows [77]:

\[
\frac{d}{dt} \left( \frac{\partial T}{\partial \dot{q}_i} \right) - \frac{\partial T}{\partial q_i} + \frac{\partial V}{\partial \dot{q}_i} = Q_i, \tag{20}
\]

Where, \( T \) is kinetic energy, \( V \) is the potential energy, and \( Q_i \) are the generalized forces/torques for each generalized coordinate, \( q_i \). The generalized coordinates are \( q_1 = \theta_1 \) and \( q_2 = \theta_2 \). The kinetic energy \((T)\) of the system is \( T = T_{m1} + T_{m2} + T_{M1} + T_{M2} \), while the potential energy \((U)\) is \( U = U_{m1} + U_{m2} + U_{M1} + U_{M2} \) with \( U_{m2} = U_{M1} = U_{M2} = 0 \). \( T_{m1}, T_{m2}, T_{M1}, T_{M2} \) and \( U_{m1}, U_{m2}, U_{M1}, U_{M2} \) are the kinetic energy and potential energy of body 1 (vertical moving element), body 2 (horizontal moving element), driving pulley 1 and driving pulley 2, respectively (see Fig. 3-14). For the H-Bot, the kinetic energy is as follows:

\[
T = \frac{1}{4} r^2 m_1 (\dot{\theta}_1^2 + \dot{\theta}_2^2) + \frac{1}{8} r^2 m_2 (\dot{\theta}_1^2 + \dot{\theta}_2^2 + 2\dot{\theta}_1 \dot{\theta}_2) + \frac{1}{2} J_{M1} \dot{\theta}_1^2 + \frac{1}{2} J_{M2} \dot{\theta}_2^2
\]

\[
U = \frac{1}{2} rm_1 g (\theta_1 - \theta_2) \tag{21}
\]

Where, \( \dot{\theta}_1 \) and \( \dot{\theta}_2 \) are the angular velocity of the driving pulleys, \( m_1 \) is the mass of the vertical moving element, \( m_2 \) is the mass of the horizontal moving element, \( g \) is the acceleration of gravity, and \( J_{M1} \) and \( J_{M2} \) are the moment of inertia at driving pulley 1 and
2 respectively (see Fig. 3-14). The inertia $J_M$ is a function of the inertia of the driving pulley 1, $J_1$, and the motor inertia, $J_m$, reflected through the gearbox ratio, $GR$, to the pulley side. Thus:

$$J_{M1} = J_1 + J_m GR^2$$  \hspace{1cm} (22)

Similarly, the moment of inertia at the driving pulley 2, $J_{M2}$, is a function of the inertia of the driving pulley 2, $J_2$, and the motor inertia, $J_m$, reflected through the gearbox ratio, $GR$, to the pulley side. Thus:

$$J_{M2} = J_2 + J_m GR^2$$  \hspace{1cm} (23)

The generalized torques are $Q_1 = \tau_{M1}$ and $Q_2 = \tau_{M2}$, where $\tau_{M1}$ and $\tau_{M2}$ are the torques developed by motor $M_1$ and $M_2$, respectively, at the output of the gearbox if used, or it is simply the developed motor torque if no gearbox is needed. Applying Lagrange’s equations to $T$, $U$ and $Q_i$, the following equations of motion are obtained:

$$\ddot{\theta}_1 \frac{r^2}{4} \left(2m_1 + m_2 + \frac{4}{r^2} J_{M1}\right) + \ddot{\theta}_2 \frac{r^2}{4} m_2 + \frac{1}{2} r m_1 g = \tau_{M1}$$  \hspace{1cm} (24)

$$\ddot{\theta}_2 \frac{r^2}{4} \left(2m_1 + m_2 + \frac{4}{r^2} J_{M2}\right) + \ddot{\theta}_2 \frac{r^2}{4} m_2 - \frac{1}{2} r m_1 g = \tau_{M2}$$

The inverse kinetic model obtained from (31) was implemented in Simulink as shown in Fig. 3-20. For higher fidelity of the results, the effects of belt stiffness, idler-pulley inertia, friction, load mass, and any external forces and torques can be added to the
to the model of the system. The level of complexity included in the model of a mechanism is in general associated with the risk of the design. High risk designs require more complete and accurate models. The risk can be measured in terms of the experience with similar mechanisms, importance to the overall process or machine, amount of innovation in the design, safety concerns, etc. In this particular case-study, the model defined in (31) was considered enough for the sake of simplifications in this case-study.

Fig. 3-20 - Inverse kinetic model of the H-bot where $G_1 = \frac{r^2}{4}[m_1 + m_2 + (4I_A/r^2)]$, $G_2 = \frac{r^2m_2}{4}$; $G_3 = \frac{r^2}{4}[2m_1 + m_2 + (4I_B/r^2)]$; and $G_4 = \frac{r^2m_2}{4}$.

3.2.6 Open-Loop Simulation

The Simulink model for the open-loop simulation of the H-Bot to estimate the required torque and speed from the motors to drive the system through the desired end-point trajectory is shown in Fig. 3-21. In this figure, “Torque_1”, “Torque_2”,
“Theta1_dot” and “Theta2_dot” are the torque and speed at the output of the gearbox connected to motor M1 and M2, respectively. The subsystem “H-Bot Position Reference” consists of equations (17) and (18) from the trajectory planning and (19) from the inverse kinematics. The motion profile obtained from this subsystem to command the motion of each motor is shown in Fig. 3-19. Meanwhile, the subsystem called “H-Bot Inverse Kinetic” in Fig. 3-21 contains the inverse kinetic model shown in Fig. 3-20, and the subsystem called “H-Bot Kinetic Model” in Fig. 3-21 contains the implementation of (24). The additional summation blocks in this figure, although not always necessary, can help reduce the position error and more accurately estimate the motor torque. The position error \((e_A\) and \(e_B\)) can be estimated, as shown in Fig. 3-21, by comparing the error between the reference signal and the motor position from the kinetic model.

Fig. 3-21 - Simulink model of H-Bot and control system.

3.2.7 Torque vs. Speed Requirement

The torque and speed curves of the H-Bot obtained from the open-loop simulation for the position reference shown in Fig. 3-19 to perform a diamond move with a diagonal of 0.4m in the Cartesian space in 2 seconds is shown in Fig. 3-22. The parameters of the
H-Bot used in this open-loop simulation are shown in Table 3-3. The open-loop simulation of the H-Bot model yielded peak angular speeds between 440 and -440 rpm for at the output of the gearbox connected to both motors and peak torque at the output of the gearbox of +1.1 to -0.40 Nm for motor M₁ and +0.40 to -1.10Nm for motor M₂. The torque and speed curves for at the output of the gearbox connected to motor M₁ of the H-Bot are shown in Fig. 3-22.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radius of the pulleys</td>
<td>r</td>
<td>0.03</td>
<td>m</td>
</tr>
<tr>
<td>Vertical moving mass</td>
<td>m₁</td>
<td>2.4</td>
<td>kg</td>
</tr>
<tr>
<td>Horizontal moving mass</td>
<td>m₂</td>
<td>3.9</td>
<td>kg</td>
</tr>
<tr>
<td>Inertia of driving pulley</td>
<td>J₁ and J₂</td>
<td>3.1x10⁻⁴</td>
<td>kg-m²</td>
</tr>
</tbody>
</table>

Fig. 3-22 - Torque and speed profile for motor a of the open-loop simulation of the H-bot
The data shown in Fig. 3-22 was then plotted as a torque vs. speed curve, as shown in Fig. 3-23, which is the typical format to represent the data for motor sizing. The \( \text{rms} \) torque and \( \text{rms} \) speed were calculated by (15). The calculated \( \text{rms} \) torque was 0.479 Nm at the \( \text{rms} \) speed of 206 rpm, as shown by the solid dot in Fig. 3-23.

![Fig. 3-23 - Torque and speed profile with \( \text{rms} \) torque and \( \text{rms} \) speed at 0.47Nm and 206rpm for the open-loop simulation of the H-bot](image)

3.2.8 Identifying Candidate Motors

Since the control of an H-Bot characterizes a motion-control application, a PMSM was chosen as the motor type. The torque vs. speed curve as the one in Fig. 3-23 was generated for a variety PMSM motors and compared to the torque vs. speed curve of each motor and gearbox (if any) tested. The motor or gearmotor that provides a torque vs. speed curve that fits the conditions defined in Section 3.1.8 can be selected as a candidate motor. The motor shown in Table 3-4 was then selected in conjunction with a 10:1 gearbox 95% efficient as a candidate motor. The torque and speed curves shown in Fig. 3-22 and Fig. 3-23 were obtained for the motor in Table 3-4. The same analysis was repeated
for motor B of the H-Bot and the same motor and gearbox were selected. If an electronic
database of the motors under consideration is available, this selection process can be
automated.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>460</td>
<td>Vac</td>
</tr>
<tr>
<td>Continuous Torque</td>
<td>0.26</td>
<td>Nm</td>
</tr>
<tr>
<td>Peak Speed</td>
<td>8000</td>
<td>rpm</td>
</tr>
<tr>
<td>Rotor Inertia</td>
<td>7.41x10^{-6}</td>
<td>kg-m^2</td>
</tr>
<tr>
<td>Torque Constant</td>
<td>0.3</td>
<td>Nm/A</td>
</tr>
<tr>
<td>Part Number</td>
<td>MPL-B1510V</td>
<td>-</td>
</tr>
<tr>
<td>Manufacturer</td>
<td>Allen-Bradley</td>
<td>-</td>
</tr>
</tbody>
</table>

The required torque vs. speed curve overlapped with the torque vs. speed curve of
the selected motor (see Table 3-4) connected to a gearbox with ratio of 10:1 and
efficiency of 95% is shown in Fig. 3-24. The required torque vs. speed curve shown in
Fig. 3-24 includes the selected motor and gearbox in terms of rotor inertia, gearbox ratio,
and gearbox efficiency. The peak and continuous torque vs. speed curves shown in Fig.
3-24 is the motor torque vs. speed curves multiplied by the gearbox ratio and efficiency.
As shown in this figure, the required torque vs. speed curve resides inside the peak torque
vs. speed region of the motor/gearbox, and the \textit{rms} torque at the \textit{rms} speed resides inside
the continuous torque vs. speed curve of the motor/gearbox, which makes this selected
motor and gearbox a candidate solution.
3.2.9 Augmenting the System

Any other losses on motor, transmissions, and mechanism not accounted in the open-loop simulation can be included at this point of the proposed mechatronic design process. However, all losses were already included in the open-loop simulation of the H-Bot.
3.2.10 Control System Design

The same control design approach used for the two-inertia system was applied with the H-Bot (see Section 3.1.10).

3.2.11 Tuning

The same tuning method used for the two-inertia system was applied with the H-Bot (see Section 3.1.11).

3.2.12 Closed-Loop Simulation

The closed-loop model for the H-Bot is shown in Fig. 3-25.

Fig. 3-25 - Closed-loop model of the H-Bot
3.2.13 Final Torque vs. Speed Requirements

The final torque vs. speed curve requirement of the H-Bot after properly tuning the closed-loop system (see Fig. 3-25) is shown in Fig. 3-26. The torque vs. speed curve obtained from closed-loop simulation as shown in Fig. 3-26 is similar to the open loop simulation results shown in Fig. 3-24.

Fig. 3-26 - Torque vs. speed curve of the closed-loop system for the H-Bot
3.2.14 Trajectory Planning Validation

In order to demonstrate the gain in positioning accuracy with the proposed trajectory planning presented in this dissertation, a comparison between the typical and the proposed trajectory planning was performed. The typical trajectory planning here consists of a master reference running at constant speed, which yields the motion profile command shown in Fig. 3-18. The results of this comparison are shown in Fig. 3-27. The position error obtained with the proposed trajectory planning was 0.5 mm, which is within the 1 mm maximum error from the system requirements. However, when the position commands ($\theta_1$ and $\theta_2$) applied to the motors were calculated with the typical trajectory planning as described for Fig. 3-18, the position error increased to 3.3 mm which is higher than the system requirements. In addition to meet the position error requirement, the presented trajectory planning yielded an end-point trajectory profile closer to a diamond shape than the end-point trajectory obtained with the typical trajectory planning. From Fig. 3-27, it can be noticed that the accuracy to trace the corners with the proposed trajectory planning was improved by simply shaping the profile of the command reference signal ($\theta_1$ and $\theta_2$) applied to the motor for the same end-point trajectory executed in the same amount of time. Consequently, lower product imperfections would occur. The tuning of the control system was kept exactly the same for this test.
When the typical and proposed trajectory planning are compared in terms of torque profile applied to the motors, the typical method yielded peak-to-peak torque of 12.46 Nm, while the presented method yielded peak-to-peak torque of 1.38 Nm as shown in Fig. 3-28. Thus, the typical method yielded a peak-to-peak torque nine times higher than the proposed trajectory planning method. Since these large peak torques have higher jerk content than the torque profile obtained with the proposed trajectory planning; mechanical vibrations, stress on mechanical and electronic components, electrical and audible noise, and stress on motors and drives can occur.
Thus, a systematic motor selection process associated with an appropriate trajectory planning method helps to obtain a stable system with low jerk content and accurate positioning throughout the end-point trajectory, including the corners.

3.3 Self-Balancing Transporter Design

Self-balance transporters have gained great importance in recent years as an option for human transportation, robotics navigation, walking assistance, and office assistants [90, 91]. The self-balancing transporter consists of two independent wheels connected to an inverted pendulum via a pivot point. This configuration results in an unstable system in open loop that is challenging to control. One motor in each wheel simultaneously controls the tilt angle of the pendulum and the navigation of the system. Stability is only obtained in closed loop when a feedback device monitors the tilt angle. A typical approach is to use a sensor fusion constituted by a gyroscope and an accelerometer as the feedback device.
3.3.1 System Requirements

The pendulum of the self-balancing transporter is connected to two parallel wheels through two motors. The motor frames are attached to the pendulum while each motor shaft is connected to each wheel. There are two independent variables in this unstable system: the linear displacement $x$ and the tilt angle $\theta$ as shown in Fig. 3-29.

![Fig. 3-29 - Physical system of a self-balancing transporter.](image)

The system requirements were chosen as follows:

- Maximum linear speed: 2 mph
- Acceleration time to maximum speed: 3 seconds
- Motion profile: 5th order polynomial profile
- Maximum pay load: 10 lb

For comparison purpose, a commercial Segway has maximum speed of 12.5mph [92]. Since this is a small version for laboratory use, the maximum speed was selected significantly lower than the Segway.
3.3.2 End-point Trajectory

The end-point trajectory was defined as a function of the linear displacement, $x$. The test-case trajectory was defined as a move of 4 meters, in 6 seconds, reaching the maximum speed 2mph (0.88 m/sec).

3.3.3 Trajectory Planning

Since the self-balancing transporter requires smooth motion, a 5th-order polynomial profile was selected to perform the required end-point trajectory. Thus, the linear motion of the system is planned as shown in Fig. 3-30. It consists of an acceleration time of 1.6 seconds, maximum velocity of 0.88 m/s during 2.8 seconds, and deceleration time of 1.6 seconds as well. This motion profile completes a move of four meters in six seconds as defined in Section 3.3.2.

![Fig. 3-30 - Trajectory planning for Self-Balancing Transporter](image-url)
3.3.4 Inverse Kinematics

The only necessary inverse kinematics equation in a self-balancing transporter is to define the relationship between motor angular position ($\phi$) and linear displacement ($x$) as a function of the radius $r$ of the wheels as shown below:

$$r\phi = x$$  \hspace{1cm} (25)

3.3.5 Kinetic and Inverse Kinetic Model

The kinetic and inverse kinetic model allows to estimate the amount of torque required to control the system and perform the desired end-point trajectory designed by trajectory planning. Assumptions were made to constrain the system and simplify the modeling. The following assumptions were used to define the equations of motion of the self-balancing transporter:

- Only 2D motion is possible
- The rotating structure is a rigid body
- Motors are identical
- Sensors for $x$ and $\theta$ give instantaneous response
- Wheels do not slip
- Tilt angle $\theta$ and tilt velocity $\dot{\theta}$ are small.

The free-body diagram of the system was defined as shown in Fig. 3-31 to identify the forces and torques acting on the system, where $F_f$ is the friction force between wheel and ground, $N$ is the normal force between wheel and ground, $T_d$ is the drive torque, $T_f$ is the Coulomb friction torque at the pivot, $m_w$ is the mass of the wheels,
\( m_p \) is the pendulum mass, \( \phi \) is the angular position of the wheels, \( B \dot{\phi} \) is viscous friction, \( g \) is gravity acceleration, \( \theta \) is tilt angle, \( G \) is the center mass of the pendulum, \( r \) is the radius of the wheels, and \( \bar{r} \) is the distance between the pivot and \( G \).

\[
\begin{align*}
T_w &= \frac{1}{2} m_w \ddot{x}^2 + \frac{1}{2} J_w \left( \frac{\ddot{x}}{r} \right)^2 \\
T_p &= \frac{1}{2} m_p \ddot{\theta}^2 + \frac{1}{2} J_p \dot{\theta}^2,
\end{align*}
\]  
(26)

Where, \( J_w \) is the inertia of the wheel, \( J_p \) is the inertia of the pendulum, and \( \bar{v} \) is the absolute velocity of the center of mass of the pendulum. From Fig. 3-31b, \( \bar{v} \) can be
derived as \( \ddot{v}^2 = (\dot{x} + \ddot{\theta} \cos \theta)^2 + (\ddot{\theta} \sin \theta)^2 \). Since, the total kinetic energy \( T \) is given as \( T = T_p + T_w \), then \( T \) can be rewritten as follows based on (26):

\[
T = \frac{1}{2} m_u \dot{x}^2 + \frac{1}{2} J_w \left( \frac{\dot{x}}{r} \right)^2 + \frac{1}{2} m_p \left[ (\dot{x} + \ddot{\theta} \cos \theta)^2 + (\ddot{\theta} \sin \theta)^2 \right] + \frac{1}{2} J_p \dot{\theta}^2
\]  

(27)

The potential energy \( V \) is only on the pendulum and is given as:

\[
V = -mg \left( r - \ddot{r} \cos \theta \right)
\]  

(28)

The generalized force \( Q_x \) is the resulting force that does work when \( \theta \) is static and \( x \) is free to move. Thus:

\[
Q_x = \left( T_d - T_f - B \dot{x} \right) \frac{1}{r}
\]  

(29)

The friction force, \( F_f \), does not do any work because there is no slip. The generalized force \( Q_\theta \) is the resulting force that does work when \( x \) is static while \( \theta \) is free to move. Thus:

\[
Q_\theta = -T_d + T_f + B \dot{x} \quad \frac{1}{r}
\]  

(30)

The equations of motion can then be derived by evaluating (27) and (28) according to (20) and substituting (29) and (30) into (20). This yields the following two equations of motion:

\[
\left( J_p + m_p r^2 \right) \ddot{\theta} + \left( m_p \ddot{r} \cos \theta \right) \ddot{x} - B \dddot{x} - m_p g \ddot{r} \sin \theta = T_f - T_d
\]  

\[
\left( m_p \ddot{r} \cos \theta \right) \ddot{x} + \left( m_p + m_w + \frac{J_w}{r^2} \right) \dddot{x} + B \dddot{x} - m_p \ddot{r} \ddot{\theta} \sin \theta = \frac{1}{r} \left( T_d - T_f \right)
\]  

(31)
The equations of motion can be linearized about the operating point $\theta = 0$ and then be implemented in Simulink as a state-space mode. Alternatively, these equations of motion in (31) can be directly implemented in Simulink as shown in Fig. 3-32. In this figure, “$r_w$” is the radius of the wheel $r$, while “$r_p$” is the center of mass $\bar{r}$.

**Fig. 3-32 - Direct kinetic model of a self-balancing transporter**

The model shown in Fig. 3-32 was embedded into a sub-system as shown in Fig. 3-33.

**Fig. 3-33 – Sub-system of the direct kinetic equations of a self-balancing transporter**

Based on the equation of motion shown above, the inverse kinetic model of the self-balancing transporter can be derived. In this case, tilt angle and linear displacement
are the input to the system, while drive torque $T_d$ is the output of the model. The inverse kinetic model of the self-balancing transporter is shown in Fig. 3-34.

Fig. 3-34 - Simulink model of the inverse kinetic equations of a self-balancing transporter

The model shown in Fig. 3-34 was embedded into a sub-system as shown in Fig. 3-35.

Fig. 3-35 - Sub-system of the inverse kinetic equations of a self-balancing transporter
3.3.6 Open-Loop Simulation

The open-loop simulation of the self-balancing transporter was used to estimate the required torque and speed from the motors to drive the system through the desired end-point trajectory. The approach used with stable systems such as the two-inertia system and the H-Bot consists in feeding the direct kinetic model with the inverse kinetic model to compute the required torque and speed to drive the system through the end-point trajectory. However, this approach is less effective with unstable systems such as the self-balancing transporter. One alternative is to rely on the inverse kinetic model only, commanded by the motion profile designed via trajectory planning method to estimate the load torque $T_d$ as shown in Fig. 3-36, although this approach may estimate higher torque than necessary. Another alternative is to use the closed-loop model to compute the required torque vs. speed curve since the control loop used with the self-balancing transporter can be a state-space control that does not required manual tuning.

Fig. 3-36 - Open-loop simulation of the self-balancing transporter
3.3.7 Torque vs. Speed Requirement

The open-loop simulation in Fig. 3-36 was used to compute the torque and speed requirements. The drive torque and speed requirements to be developed by a single motor of the self-balancing transporter performing the motion profile defined in Fig. 3-30 is shown in Fig. 3-37, while the corresponding torque vs. speed requirement is shown in Fig. 3-38. The open-loop simulation in Fig. 3-36 estimated the drive torque developed by the two motors. Thus, the drive torque requirement shown in Fig. 3-37 and Fig. 3-38 is half of the drive torque estimated from the model in Fig. 3-36. The drive torque is the torque at the wheels. Thus, if a gearmotor is used, the drive torque is at the output of the gearbox.

The parameters of the self-balancing transporter used in this open-loop simulation are shown in Table 3-5. The open-loop simulation of the self-balancing transporter model yielded peak angular speeds of 56 rpm at the wheels and peak torque between -0.80 and +0.91Nm at each wheel.

Table 3-5 – Parameters for the self-balancing transporter and motors

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radius of wheels</td>
<td>$r$</td>
<td>0.15m</td>
</tr>
<tr>
<td>Mass of wheels</td>
<td>$m_w$</td>
<td>1.166 kg</td>
</tr>
<tr>
<td>Inertia of wheels</td>
<td>$J_w$</td>
<td>0.0216 kgm$^2$</td>
</tr>
<tr>
<td>Mass of pendulum</td>
<td>$m_p$</td>
<td>6.314 kg</td>
</tr>
<tr>
<td>Center of mass</td>
<td>$\bar{r}$</td>
<td>0.115 m</td>
</tr>
<tr>
<td>Pendulum inertia</td>
<td>$J_p$</td>
<td>0.35kgm$^2$</td>
</tr>
<tr>
<td>Damping</td>
<td>$B$</td>
<td>0.024Nm/rad/s</td>
</tr>
<tr>
<td>Friction torque</td>
<td>$T_f$</td>
<td>0.11Nm</td>
</tr>
<tr>
<td>Rotor Inertia</td>
<td>$J_m$</td>
<td>7.1x10$^{-6}$ kgm$^2$</td>
</tr>
<tr>
<td>Gearbox ratio</td>
<td>$GR$</td>
<td>19.7</td>
</tr>
<tr>
<td>Gearbox efficiency</td>
<td>$\eta$</td>
<td>0.84</td>
</tr>
</tbody>
</table>
The curves shown in Fig. 3-37 was then plotted as a torque vs. speed curve, as shown in Fig. 3-38, since it is the typical format to represent the data for motor sizing. The \textit{rms} torque and \textit{rms} speed were calculated according to (15) and the resulting \textit{rms} torque was 0.45 Nm at an \textit{rms} speed of 46.7 rpm at each wheel as shown in Fig. 3-38 by the solid dot.
3.3.8 Identifying Candidate Motors

Since the self-balancing transporter is powered by batteries, brushed DC gearmotors were chosen as the motor technology. By generating the torque vs. speed curve shown in Fig. 3-38 for a variety of brushed DC motors and comparing to the torque vs. speed curve of the motor, the motor that provides a torque vs. speed curve that matches the conditions defined in Section 3.1.8 can be selected as a candidate motor. The brushed DC gearmotor shown in Table 3-6 was selected as a candidate motor. The torque and speed curves shown in Fig. 3-37 and Fig. 3-38 were generated with the rotor inertia and gearbox of the motor in Table 3-6.
Table 3-6 – Selected motor for the self-balancing transporter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part Number</td>
<td>-</td>
<td>GM9236S021</td>
</tr>
<tr>
<td>Manufacturer</td>
<td>-</td>
<td>Pittman</td>
</tr>
<tr>
<td>Torque Constant</td>
<td>$K_t$</td>
<td>0.0379 Nm/A</td>
</tr>
<tr>
<td>Motor Resistance</td>
<td>$R_a$</td>
<td>2.49Ω</td>
</tr>
<tr>
<td>Motor Inductance</td>
<td>$L_a$</td>
<td>2.63 mH</td>
</tr>
<tr>
<td>Rotor Inertia</td>
<td>$J_m$</td>
<td>7.1 x 10^{-6} kgm²</td>
</tr>
<tr>
<td>Back EMF constant</td>
<td>$K_e$</td>
<td>0.0458 V/rad/s</td>
</tr>
<tr>
<td>Friction torque</td>
<td>$T_f$</td>
<td>0.055 Nm</td>
</tr>
<tr>
<td>Gear ratio</td>
<td>$GR$</td>
<td>19.7</td>
</tr>
<tr>
<td>Gearbox efficiency</td>
<td>$\eta$</td>
<td>0.84</td>
</tr>
</tbody>
</table>

The required torque vs. speed curve from Fig. 3-38 overlapped with the torque vs. speed curve of the selected gearmotor in Table 3-6 is shown in Fig. 3-39. The intermittent (peak) and continuous torque vs. speed curves shown in Fig. 3-39 are at the output of the gearbox of the selected gearmotor. As shown in this figure, the required torque vs. speed curve resides inside the peak torque vs. speed region of the selected gearmotor, while the rms torque at the rms speed point is inside the continuous torque vs. speed curve of the motor.

![Fig. 3-39 - Torque vs. speed curve of the selected motor overlapped with the required torque vs. speed curve obtained from the open-loop simulation of the self-balancing transporter](image-url)
3.3.9 Augmenting the System

Any other losses on motor, transmissions, and mechanism not accounted in the open-loop simulation can be included at this point of the proposed mechatronic design process. However, all losses were already included in the open-loop simulation of the self-balancing transporter.

3.3.10 Control System Design

Some of the options to control a self-balancing transporter include PD control and state-space control. In this dissertation, state-space control was selected as the control system for the self-balancing transporter. In order to compute the control loop gains, the state-space model of the self-balancing transporter needs to be linearized as shown next.

3.3.10.1 State-space equation

The equations of motion for the case-study self-balancing transporter given in (31) can be rewritten as follows:

\[
\ddot{\theta} = \frac{1}{\left(I_p + m_p \bar{r}^2\right)} \left[ T_f - T_d - (m_p \bar{r} \cos \theta) \dot{x} + B \frac{\dot{x}}{r} + m_p g \bar{r} \sin \theta \right]
\]

\[
\dot{x} = \frac{1}{\left(m_p + m_w + \frac{I_w}{r^2}\right)} \left[ \frac{1}{r} \left( T_d - T_f \right) - (m_p \bar{r} \cos \theta) \dot{\theta} - B \frac{\dot{x}}{r^2} + m_p \bar{r} \dot{\theta}^2 \sin \theta \right]
\]

(32)

And, by defining the following coefficients:
The non-linear equations in (33) can then be simplified as follows:

\[
\begin{align*}
C_1 &= I_p + m_p \ddot{r}^2 \\
C_2 &= m_p + m_w + \frac{I_w}{r^2} \\
C_3 &= m_p g \ddot{r} \\
C_4 &= m_p \ddot{r} \\
C_5 &= \frac{B}{r} \\
C_6 &= \frac{B}{r^2}
\end{align*}
\]

(33)

Because the tilt angle \( \theta \) is about zero during operation, the following approximation can be used to linearize these equations: \( \cos \theta \cong 0, \sin \theta \cong \theta, \) and \( \dot{\theta}^2 \cong 0. \)

Thus, the linear equations can be derived from (34) after some manipulations as:

\[
\begin{align*}
\ddot{\theta} &= \frac{1}{C_1} \left( T_f - T_d - C_4 \cos(\theta) \ddot{r} + C_5 \ddot{r} + C_3 \sin \theta \right) \\
\ddot{x} &= \frac{1}{C_2} \left( \frac{1}{r} \left( T_d - T_f \right) - C_4 \cos(\theta) \ddot{\theta} - C_6 \ddot{r} + C_4 \ddot{r}^2 \sin \theta \right)
\end{align*}
\]

(34)

\[
\begin{align*}
\ddot{\theta} &= \frac{C_2}{C_1 C_2 - C_4^2} \left( T_f - T_d \right) \left( 1 + \frac{C_4}{r C_2} \right) + \left( C_5 + \frac{C_4 C_6}{C_2} \right) \ddot{r} + C_3 \theta \\
\ddot{x} &= \frac{C_1}{C_1 C_2 - C_4^2} \left( T_d - T_f \right) \left( \frac{C_1 + r C_4}{r C_1} \right) - \left( \frac{C_4 C_5 + C_1 C_6}{C_1} \right) \ddot{r} - \frac{C_3 C_4}{C_1} \theta
\end{align*}
\]

(35)
The drive torque \( T_d \) (at the output of the gearbox of the motor) is calculated as follows:

\[
T_d = 2K_i i_a GR \eta \tag{36}
\]

Where, \( K_i \) is motor torque constant, \( i_a \) is the motor current, \( GR \) is the gearbox ratio, and \( \eta \) is the efficiency of the gearbox. The factor 2 is to account for two motors driving the wheels of the self-balancing transporter. The motor current \( i_a \) is defined as follows:

\[
i_a' = \frac{v_a}{L_a} - \frac{R_a}{L_a} i_a - \frac{K_e}{L_a} \dot{\phi} GR
\]

or:

\[
i_a' = \frac{v_a}{L_a} - \frac{R_a}{L_a} i_a - \frac{K_e}{L_a} \dot{x} GR
\]

Defining the five states as \( \theta, \dot{\theta}, x, \dot{x}, \) and \( i_a \), where \( \theta \) and \( \dot{\theta} \) are the tilt angle and tilt velocity, and \( x \) and \( \dot{x} \) are the linear displacement and linear velocity, the state-space equation can be derived from (35), (36) and (116) as follows:

\[
\begin{bmatrix}
\dot{\theta} \\
\dot{x} \\
\dot{x} \\
i_a'
\end{bmatrix} =
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
\frac{C_2 C_3}{C_1 C_2 - C_4^2} & 0 & \frac{C_2 C_5 + C_4 C_6}{C_1 C_2 - C_4^2 C_2} & \frac{2K_i GR \eta (r C_2 + C_4)}{r C_1 C_2 - r C_4^2} & 0 \\
0 & 0 & 1 & 0 & 0 \\
\frac{\dot{C}_4 C_4}{C_1 C_2 - C_4^2} & 0 & -\frac{C_4 C_5 + C_1 C_6}{C_1 C_2 - C_4^2} & \frac{2K_i GR \eta (r C_4 + C_1)}{r C_1 C_2 - r C_4^2} & 0 \\
0 & 0 & 0 & -\frac{K_i \dot{\phi} GR}{L_a r} & 0
\end{bmatrix}
\begin{bmatrix}
\theta \\
\dot{\theta} \\
x \\
\dot{x} \\
i_a'
\end{bmatrix} +
\begin{bmatrix}
0 \\
0 \\
0 \\
1/L_a
\end{bmatrix}
\begin{bmatrix}
v_a \\
i_a
\end{bmatrix}
\tag{38}
\]

Which can be written in short-hand as follows:

\[
\dot{x} = Ax + Bu
\]

\( (39) \)
The output equation is defined as follows:

\[
y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \\ i_a \\ \end{bmatrix} + 0v_a
\]  
(40)

Which can be written as follows:

\[
y = Cx + Du
\]  
(41)

3.3.10.2 Optimal Control

Optimal feedback control was used to balance the self-balancing transporter, by minimizing the following cost function [93]:

\[
J = \frac{1}{2} x^T(T)S(T)x(T) + \frac{1}{2} \int_{t_0}^{T} (x^TQ(t)x + u^TR(t)u)dt
\]  
(42)

Where, \( S \) and \( Q \) are symmetric positive semi-definite matrices, and \( R \) is a symmetric positive definite matrix. The matrix \( Q \) is the cost penalty to the states and it is used to minimize error; \( R \) is the cost penalty to the inputs and it is used to minimize energy; and \( S \) is the penalty to the final state. A function is positive semi-definite if it is positive in all states of a given region, except at the origin and in states where its value is zero. A function is positive definite in a given region including at the origin if it is positive for all non-zero states \((x)\).

For the linear system in (39), the control input can be defined as follows:
Where, the Kalman gain $K$ can be defined as follows:

$$K = -R^{-1}B^T S$$  \hspace{1cm} (44)$$

And, the control input can be rewritten as follows:

$$u(t) = -Kx(t)$$  \hspace{1cm} (45)$$

Where, $R$ is defined in the cost function, $B$ from the system model, and the intermediate equation $S$ can be computed via the *Riccati equation* defined as follows [93]:

$$-\dot{S} = A^T S + SA - SBR^{-1}B^T S + Q$$  \hspace{1cm} (46)$$

The algebraic Riccati equation is defined as $t \to \infty$ which makes $\dot{S} = 0$, can be solved in Matlab with commands *lqr* (Linear-Quadratic Regulator (LQR) design) or *care* (Continuous-time algebraic Riccati equation solution) if $(A,B)$ is stabilizable. The solution of the Riccati equation yields $S$ which can then be applied in (44) to find the optimal control gain $K$ for (45). The state-space equations $A$ and $B$ defined in (38) are controllable, while $A$ and $C$ defined in (38) and (40) are observable [93].

### 3.3.11 Tuning

With the $A$ and $B$ from the system model, and with selected matrices $Q$ and $R$, the Riccati equation can be solved with the command *lqr* in Matlab to compute the optimal control loop gain $K$.

The matrix $R$ was set to $[1]_{1x1}$ while the matrix $Q$ was set as follows to obtain the desired response:
\[
Q = \begin{bmatrix}
10 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 5000 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\] (47)

3.3.12 Closed-Loop Simulation

The closed-loop model for the self-balancing transporter using the optimal control model defined in Section 3.3.10 is shown in Fig. 3-40. The block called “Command Position \( x \)” generates the motion profile in Fig. 3-30. The subsystem “DC motors” in Fig. 3-40 contains the model in Fig. 3-41 of the two motors that drive the self-balancing transporter.

Fig. 3-40 – Closed loop model for the self-balancing transporter
3.3.13 Final Torque vs. Speed Requirements

The final torque vs. speed requirement obtained from closed-loop simulation for a single motor of the self-balancing transporter is shown in Fig. 3-42 which is similar to the open-loop simulation results shown in Fig. 3-39.

Fig. 3-42 - Torque vs. speed curve of the closed-loop system for the H-Bot
The command and feedback displacement $x$ obtained from the closed-loop model of the self-balancing transporter in Fig. 3-40 tuned according to Sections 3.3.10 and 3.3.11, is shown in Fig. 3-43.

![Graph](image-url)

**Fig. 3-43 – Command and feedback $x$**

Since the required torque vs. speed curve resides in the peak torque vs. speed region of the selected gearmotor and the $rms$ torque at the $rms$ resides in the continuous torque vs. speed region as shown in Fig. 3-42, and the system present stable behavior as shown in Fig. 3-43, the selected motor is a valid solution.

### 3.4 Summary

A mechatronic design process was developed to size motors of single axis and multi-axis coordinated systems for industrial applications. In this process, a trajectory planning method was proposed to design the motion profiles used as the reference signals...
to the control systems controlling each motor in a system. The trajectory planning uses the concepts introduced in Chapter 2 in regards to types and selection of motion profiles, asymmetry of motion profiles, and merging of multiple segments of motion profiles. This mechatronic design process was demonstrated for three distinct systems: a two-inertia system as a single-axis system, a Cartesian two-axis planar robot as a multi-axis coordinated system, and a self-balancing transporter as an unbalanced system.

This chapter concludes the first major section of this dissertation. The next chapter describes the implementation of a two-inertia system, a Cartesian two-axis planar robot, and a self-balancing transporter that were used as the validation systems for the energy optimization method described in Chapter 5.
CHAPTER 4

Implementation of Physical Systems

In this chapter, the implementation of the two-inertia system, the Cartesian two-axis parallel robot – H-Bot, and the self-balancing transporter is described. Sensors, processors, electronic power control board, electrical circuit, programming, and parameter identification are some of the points described next to build each system. The two-inertia system was built with industrial devices due to the high dynamics to be tested with this system, while the H-Bot and the self-balancing transporter were built with lab devices.

Although these system were implemented with the intent of validating the method proposed in this dissertation, they can be used beyond this dissertation in STEM education and by practicing engineers.

The automation system used with the two-inertia system demonstrates how to integrate sensors, controllers, servo drives, motors, and mechanical components using industrial electronic equipment. It also demonstrates how to control, program, install, and configure a motion control system for industrial machines using state-of-the-art technology extensively employed in industrial automation.

Meanwhile, the design and implementation of each piece of hardware and program used to control the H-Bot and the self-balancing transporter were based on fundamental concepts, including the methods to decodify encoders, control of power delivered to motors via H-Bridges and PWM control, design of control systems, and
conditioning of the sensor signals via digital signal processing. These fundamental concepts are embedded in industrial equipment used in industrial automation. But, although these concepts are embedded in industrial hardware, they are essential for engineers to better design, configure, implement, program, and specify industrial automation system.

4.1 Implementation of the two-inertia system

The two-inertia system consists of a permanent-magnet synchronous motor (PMSM) connected to a fly-wheel through a rod and a coupling as shown in Fig. 4-1.

![Fig. 4-1 – Physical system of the two-inertia system](image)

4.1.1 Controller

The controller used with the two-inertia system is an industrial controller called programmable automation controller (PAC) that can be used for motion control applications. The main function of this controller used with the two-inertia system is to
generate the position reference signal to the servo drive controlling the motor. The program of the controller was developed in Ladder Logic [94, 95].

4.1.2 Encoder

The feedback device used in this system is a high resolution absolute encoder with 21 bits of resolution which yields 2097152 encoder counts per revolution. This is equivalent to a resolution of 0.000171 degrees, which is suitable for motion control applications.

4.1.3 Power Control

The power control device of this system is provided by a high-performance industrial servo drive [96] powered by a three-phase 460V line, which matches the voltage of the motor. This drive is controlled via Ethernet by the controller described in Section 4.1.1. This drive can regulate position, which means that it can be used to command a motor to follow the position command signal obtained from the controller.

4.1.4 Control System and Tuning

The control system for the two-inertia system is the cascade PI control system described in Section 3.1.10 which is embedded in the high-performance industrial servo drive mentioned in the previous section. Thus, the position reference computed in the controller and sent to the drive is the position reference signal for the cascade PI control loop running in the servo drive. This enables the drive to command the motor to follow the position reference defined in the controller.
This cascade control system was tuned according to the procedure described in Section 3.1.11.

4.1.5 Electrical Design and Final System

An overview of the system with the controller, servo drive, motor, encoder, and the two-inertia system is shown in Fig. 4-2. The program of the controller contains the position motion profile that is sent to the servo drives via Ethernet in real-time to provide the position reference for the cascade control system in the drive. This drive has a position, velocity, and current loop in a cascade configuration as shown in Fig. 3-10. The feedback signal for the position loop is the signal from the encoder, while the feedback signal for the velocity loop is the derivative of the encoder signal.

Fig. 4-2 – Apparatus of the two-inertia system
4.1.6 Parameter Identification

The parameters to be identified in this system are the load inertia, the spring constant, the damping, and the rotor inertia as shown in Fig. 3-2. The motor used with the two-inertia system is shown in Table 3-2 and also shown again below in Table 4-1 for convenience. The parameters of the two-inertia system are shown in Table 4-2 and the procedures to obtain each parameter are given next.

Table 4-1 - Selected motor for the two-inertia system

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brand</td>
<td>-</td>
<td>Allen-Bradley</td>
<td>-</td>
</tr>
<tr>
<td>Model</td>
<td>-</td>
<td>MPL-B310P-M</td>
<td></td>
</tr>
<tr>
<td>Torque Constant</td>
<td>$K_t$</td>
<td>0.573</td>
<td>Nm/A</td>
</tr>
<tr>
<td>Peak Stall Current</td>
<td>$I_m$</td>
<td>1.7</td>
<td>A</td>
</tr>
<tr>
<td>Rated Voltage</td>
<td>$V$</td>
<td>460</td>
<td>V</td>
</tr>
<tr>
<td>Resistance</td>
<td>$R_a$</td>
<td>18.9</td>
<td>Ω</td>
</tr>
<tr>
<td>Inductance</td>
<td>$L_a$</td>
<td>92</td>
<td>mH</td>
</tr>
<tr>
<td>Rotor Inertia</td>
<td>$J_m$</td>
<td>$4.4 \times 10^{-5}$</td>
<td>kg-m$^2$</td>
</tr>
<tr>
<td>Back-EMF Constant</td>
<td>$K_e$</td>
<td>0.936</td>
<td>V/rad/s</td>
</tr>
<tr>
<td>Viscous Damping</td>
<td>$b$</td>
<td>$2.5 \times 10^{-6}$</td>
<td>N-m-s</td>
</tr>
</tbody>
</table>

Table 4-2 – Parameters of the two-inertia system

<table>
<thead>
<tr>
<th>Element</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Inertia</td>
<td>$J_l$</td>
<td>$2.52 \times 10^{-3}$</td>
<td>kg-m$^2$</td>
</tr>
<tr>
<td>Motor Inertia</td>
<td>$J_m$</td>
<td>$4.4 \times 10^{-5}$</td>
<td>kg-m$^2$</td>
</tr>
<tr>
<td>Damping</td>
<td>$b$</td>
<td>$2.5 \times 10^{-6}$</td>
<td>N-m-s</td>
</tr>
<tr>
<td>Total torsional stiffness</td>
<td>$k$</td>
<td>1281</td>
<td>Nm/rad</td>
</tr>
</tbody>
</table>
4.1.6.1 Load inertia \((J_l)\)

The load inertia was calculated from the measurements of the dimensions of the fly-wheel used as the load. The dimensions and calculated inertia value are shown in Table 4-3.

Table 4-3 – Load inertia

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outer diameter</td>
<td>127</td>
<td>mm</td>
</tr>
<tr>
<td>Inner diameter</td>
<td>25</td>
<td>mm</td>
</tr>
<tr>
<td>Width</td>
<td>12.5</td>
<td>mm</td>
</tr>
<tr>
<td>Material</td>
<td>Steel</td>
<td>-</td>
</tr>
<tr>
<td>Load Inertia (J_l)</td>
<td>2.52x10^{-3}</td>
<td>kg-m²</td>
</tr>
</tbody>
</table>

4.1.6.2 Motor inertia \((J_m)\)

The motor inertia was obtained from the motor datasheet as shown in Table 4-1.

4.1.6.3 Damping \((b)\)

The damping of the system was assumed to be the damping of the motor given in Table 4-1.

4.1.6.4 Spring constant \((k)\)

The compliant components in this two-inertia system are the motor shaft, coupling, and rod as shown in Fig. 4-1. The total torsional stiffness of the system can be obtained from the calculated torsional stiffness of each compliant components, or via a Bode plot of the plant. Both methods were employed and compared next.

The procedure to obtain the calculated torsional stiffness is as follows. The dimensions of the motor shaft and rod were measured to compute the torsional stiffness of these two elements and the results are show in Table 4-4.
Table 4-4 – Data of motor shaft and rod

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rod length</td>
<td>203.2</td>
<td>mm</td>
</tr>
<tr>
<td>Motor shaft length</td>
<td>72</td>
<td>mm</td>
</tr>
<tr>
<td>Diameter of rod and motor shaft</td>
<td>15.6</td>
<td>mm</td>
</tr>
<tr>
<td>Material</td>
<td>Steel</td>
<td>-</td>
</tr>
<tr>
<td>Spring constant of the rotor shaft and rod</td>
<td>1894</td>
<td>Nm/rad</td>
</tr>
</tbody>
</table>

The torsional stiffness of the motor shaft and rod was calculated as follows:

\[ K = \frac{\pi G}{32L} (OD^4 - ID^4) \quad (48) \]

Where, \( G \) is the shear module in [GPa], \( L \) is the length of the rod and motor shaft in [m], \( OD \) and \( ID \) are the outer and inner diameter of the rod and motor shaft, respectively, in [m], and \( K \) is the torsional stiffness in [Nm/rad].

Meanwhile, the torsional stiffness of the coupling was obtained from the coupling datasheet as shown in Table 4-5.

Table 4-5 – Data of the coupling

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brand</td>
<td>Ruland</td>
<td>-</td>
</tr>
<tr>
<td>Model</td>
<td>JD26/41</td>
<td>-</td>
</tr>
<tr>
<td>Torsional stiffness</td>
<td>540</td>
<td>Nm/rad</td>
</tr>
</tbody>
</table>

From the torsional stiffness of the motor shaft and rod shown in Table 4-4 and the torsional stiffness of the coupling shown in Table 4-5, the total torsional stiffness of the system was calculated as 420 Nm/rad.
Although the estimation of stiffness of the rod based on (48) is relatively accurate, variation can exist between the torsional stiffness of the coupling specified in the datasheet and the actual stiffness. These variations can be due to mechanical parts tolerances, misalignment of the parts constituting the coupling, damaged parts, and misalignments between rod and motor shaft.

In order to identify the actual torsional stiffness $k$ of the system, a Bode plot experimentally obtained from the plant can be used [97]. The torsional stiffness can be estimated from the anti-resonance frequency $\omega_{AR}$ as shown below.

Converting (13) and (14) to the s-domain and reorganizing the term, the following is obtained:

$$\theta_m(jm^2s^2 + bs + k) = T_m + \theta_l(bs + k)$$  \hspace{1cm} (49)  

$$\theta_l(jm^2s^2 + bs + k) = \theta_m(bs + k)$$ \hspace{1cm} (50)

By substituting (50) into (49) and manipulating the terms, the relationship between motor position $\theta_m$ and motor torque $T_m$ can be defined as follows.

$$\frac{\theta_m}{T_m} = \frac{jm^2s^2 + bs + k}{s^4[jmJ_l^2 + b(J_m + J_l) + k(J_m + J_l)]}$$ \hspace{1cm} (51)

Which can also be written as follows:

$$\frac{\theta_m}{T_m} = \frac{1}{(J_m + J_l)s^2} \left[ \frac{jm^2s^2 + bs + k}{J_mJ_l^2s^2 + bs + k} \right]$$ \hspace{1cm} (52)
Where the term outside the square bracket represents a rigid load, while the terms inside the square brackets represent the compliance in the system. If the numerator in (52) is compared to the standard form \((s^2 + \zeta \omega_{AR}s + \omega_{AR}^2)\), then the anti-resonance frequency \(\omega_{AR}\) can be defined as follows:

\[
\omega_{AR} = \sqrt{\frac{k}{J_i}}
\]  

(53)

The Bode plot for the plant with \(T_m\) being the input and \(\theta_m\) being the output, was experimentally obtained for the system shown in Fig. 4-1. This experimentally obtained plant Bode plot for the two-inertia system is shown in Fig. 4-3.

---

**Fig. 4-3** – Experimental plant Bode plot for the two-inertia system for \(\theta_m/T_m\)
In Fig. 4-3, the anti-resonance frequency was identified as 113.5Hz, which applied to (53) yields:

\[
k = J \omega_{AR}^2 = 0.00252 \text{kgm}^2 (2\pi \times 113.5\text{Hz})^2
\]

\[
k = 1281\text{Nm/rad}
\]  

This demonstrated a high variation between the calculated torsional stiffness of 420Nm/rad and the experimentally obtained value of 1281Nm/rad. Thus, the torsional stiffness obtained experimentally was chosen.

4.2 Implementation of an H-Bot

The implementation of an H-Bot in respect to the design and programming of the processors, sensors, control system, power structure, and electrical circuit is described in this section.

4.2.1 Processor

For the H-Bot, Arduino Mega boards were used [98, 99]. An Arduino Mega board contains 54 digital I/O’s from which 15 can be PWM outputs, and 16 analog inputs. The digital I/O’s operate in 5V. The resolution of the analog inputs is 10-bit and they operate also in 5V. The carrier frequency of the PWM outputs is by default 490Hz, but the configuration of the Arduino board was modified to have some of the PWM outputs operating at 3.9 kHz to better control the dc motors shown later in this chapter.

Two Arduino Mega boards were used with the H-Bot. One board was used to decodify the encoders and another board was used for control. The board used for encoder decodification was programmed in C-code, while the board for control was programmed in Simulink and downloaded to the Arduino board via Embedded Code-
Generation [100, 101]. The encoder decodification was performed in a dedicated Arduino Mega board to avoid potential disruptions and interferences in the control of the H-Bot due to the interrupts needed in the encoder decodification process. The angular position was then sent from one Arduino board to the other via digital I/O’s.

4.2.2 Encoder

The encoders used with the self-balancing transporter are incremental encoders with 500 PPR (pulses per revolution). The program to convert the encoder signal in angular motor position was developed in C for Arduino Mega boards as shown in Appendix A.

The encoders are sensors extensively used in industry to measure angular position and velocity of motors. They are usually connected on the rear side of motors. A second encoder can also be strategically installed in other parts of a machine to better control position and/or velocity at that particular point. Linear encoders are also available and are used to measure linear position and linear velocity. A typical application for linear encoders is in the control of the multiple linear axes in CNC machines and gantries.

In the self-balancing transporter, each motor has an incremental encoder. Incremental encoders have two channels (A and B) out of phase by 90° as shown in Fig. 4-4 and Fig. 4-5 in order to allow the measurement of position and direction. The number of gaps in channels A and B of the rotating disk determines the resolution of the encoder. Each gap generates a pulse as shown in Fig. 4-5. Typical numbers of pulses per revolution (PPR) of incremental encoders are 500 PPR, 512 PPR, 1024 PPR, 2048 PPR, etc. Thus, a 1024 PPR encoder generates 1024 pulses in channel A and 1024 pulses in channel B in a single encoder revolution.
Fig. 4-4 - Mechanism of a rotating encoder with 40 pulses per revolution (PPR)

![Diagram of encoder mechanism]

The decodification of the signals in channels A and B into angular position values can use three different methods:

- **Single edge**: in this method, only the rising or falling edge of only one channel (A or B) is monitored to measure the angular position. The resolution of the measured angular position matches the PPR of the encoder. Thus, a 1024 PPR encoder using a
single edge decodification method yields a resolution of \(\frac{360 \text{ deg/rev}}{1024 \text{ pulses/rev}}\) = 0.351 deg/encoder pulse.

- **Dual edge**: in this method, both rising and falling edge of only one channel (A or B) is monitored to measure the angular position as shown in Fig. 4-6. Thus, the resolution of the measured angular position is twice the PPR of the encoder. For example, a 1024 PPR encoder using a dual edge decodification method yields a resolution of \(\frac{360 \text{ deg/rev}}{2 \times 1024 \text{ pulses/rev}}\) = 0.175 deg/pulse. Every time that an edge is detected on the channel being monitored, the state of the other channel is also measured. This allows to identify the direction of motion as shown in Fig. 4-6. Once the direction of motion is identified, the angular position can be either incremented or decremented according to the direction of motion. With the Arduino boards, the edges are detected via interrupts. Since the edges can occurs at fast rates, the interrupts allow to execute the encoder decodification code every time that an edge is detected and accurately measure the motor angular position.

Fig. 4-6 - Dual edge encoder decodification
• **Quadrature:** this method is also known as AQB (or AQuadB), and it monitors both rising and falling edges of both channels A and B to measure the motor angular position as shown in Fig. 4-7. The resolution of the angular position is four times the PPR of the encoder. Thus, a 1024 PPR encoder using an AQB decodification method yields a resolution of $(360 \text{ deg/rev}) / (4 \times 1024 \text{ pulses/rev}) = 0.088 \text{ deg/pulse}$.

The single edge was the method used to measure the motor angular position of the H-Bot which would provide enough resolution for a gearmotor with a 19.7:1 gearbox.

The C-code implementation of the single edge method to read the angular position of two encoders is shown in Appendix A. The basic algorithm to measure angular position can be derived from Fig. 4-6 as follows, assuming that the positive edges of channel A are being monitored:
When a positive edge of channel A is detected

If $A \neq B$, then
Increment Position (Forward direction detected)

If $A = B$, then
Decrement Position (Reverse direction detected)

The measured motor angular position from each encoder is then sent to digital outputs as a binary number at a predefined rate of 1ms. This rate was set to 1ms to match the fastest execution time that in general the control board would be able achieve since it was programmed in Simulink. These digital outputs of board decodifying the encoders are connected to digital inputs on the control board to transfer the measured angular position from one board to the other. The angular position values on the digital inputs of the control board is then converted from binary to decimal, and the respective angular velocity is calculated as detailed in Appendix B.

4.2.3 Power Structure Design

The torque, velocity and direction control of the motors is obtained with H-Bridges as shown in Fig. 4-8. An H-bridge consists of four power devices where only two are turned on at the same time. The power devices are in general transistors or IGBTs (Insulated Gate Bipolar Transistor). If IGBT 1 and 4 are turned on, the motor turns in one direction. If IGBT 2 and 3 are turned on, the motor turns in the opposite direction. If all IGBT’s are off, or if IGBT 2 and 4 are on, the motor stops. The gates of the IGBT’s are controlled by a PWM (Pulse Width Modulation) signal [80-82, 102]. By controlling the duty cycle of the PWM signal, the motor torque and speed can be controlled.
The selected H-Bridge for this project was the 5A H-Bridge TLE5206 [103]. This H-Bridge has two pins (IN1 and IN2) to control the four power devices. When IN1 = 1 and IN2 = 0, the motor turns in one direction, when IN1 = 0 and IN2 = 1, the motor turns in the opposite direction, and when IN1 = IN2, the H-Bridge goes in brake mode.

The Simulink code to control this H-Bridge for motors M1 and M2 is shown in Fig. 4-9. For motor M1, if the voltage command (“V_Cmd M1”) is positive, Pin 8 is set while Pin 10 is reset to command a forward move. The voltage command is then applied to Pin 11 after being converted to an 8-bit signal for the PWM output in Pin 11. If the voltage command is negative, Pin 8 is reset while Pin 10 is set to command reverse move. Similarly, the voltage command is applied to Pin 11 as a PWM signal with duty cycle proportional to the voltage command.
The logic to control the H-Bridge is then performed by two AND gates connected to Pins 8, 10 and 11 for motor M1 as shown in Fig. 4-10. Another AND gate is used for motor M2. For a positive voltage command in “V_Cmd M1” in this figure, Pin 8 will be 1, Pin 10 will be 0, and Pin 11 will have a PWM signal with duty cycle proportional to the voltage command. Thus, the AND gate connected to Pin 8 will let the PWM signal from Pin 11 pass to the input IN1 of the H-Bridge while IN2 will stay at zero. This will command the motor in one direction. For a negative command voltage in “V_Cmd M1”, Pin 8 will be “0”, Pin 10 will be 1, and Pin 11 will have a PWM signal with duty cycle proportional to the voltage command. Thus, the AND gate connected to Pin 10 of the...
Arduino board will let the PWM signal in Pin 11 be applied to input IN2 of the H-Bridge, while IN1 will stay at zero. This will command the motor in the opposite direction.

Fig. 4-10 – H-Bridge control

4.2.4 Control System and tuning

The design of the control system for the H-Bot was described in Section 3.2.10 while the tuning method was described in Section 3.2.11.

The implementation of the control system for the H-Bot is shown in Fig. 4-11 and detailed in Fig. 4-12. The “Position Reference” contains the implementation of the motion profile for the angular positions $\theta_1$ and $\theta_2$ to perform the desired trajectory in the Cartesian space. This subsystem contains the implementation of (17), (18) and (19). More details about this subsystem will be provided later in Section 5.2.4.

The subsystems “Theta 1 – Feedback Position” and “Theta 2 – Feedback Position” were implemented as shown in Appendix B. The only difference resides in Fig. B-2 where the gain “r_w” does not exist for the H-Bot.

The subsystem “H-Bridge Power Control” contains the code shown in Fig. 4-9.
Fig. 4-11 – Implementation of the control system of the H-Bot in Simulink to load into the Arduino

Fig. 4-12 - Detailed control system for the Arduino to control the H-Bot
4.2.5 Electrical Circuit Design

The electrical diagram of the H-Bot is shown in Fig. 4-13, and it includes:

- An Arduino Mega to decodify two encoders
- An Arduino Mega to control the H-Bot
- A 12V voltage regulator LM7812 to power both Arduino boards
- A 5V voltage regulator LM7805 to power the encoders and the AND gate (74LS08)
- Two 5A full H-Bridges TLE5206 [103] to power each motors
- Quadruple 2-input AND gate 74LS08 [104] to work as the logics between the digital outputs of the Arduino board and the inputs of the H-Bridges.
- Two 24V brushed gearmotor GM9236S021 from Pittman [105]

Fig. 4-13 – Electrical diagram of the H-Bot
The electrical diagram in Fig. 4-13 was mounted in a universal circuit board as shown in Fig. 4-14.

![Circuit board for control of the H-Bot](image)

**Fig. 4-14** - Circuit board for control of the H-Bot

### 4.2.6 Final System

The H-Bot with the motors, control boards, and Arduino boards is shown in Fig. 4-15.
4.2.7 Parameter Identification

The parameters of the motors used with the H-Bot are shown in Table 4-6 [105]. The parameters of the H-Bot are shown in Table 4-2 and the procedures to obtain each parameter are given next.
Table 4-6 – Parameter of the motors for the H-Bot

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part Number</td>
<td>-</td>
<td>GM9236S021</td>
<td>-</td>
</tr>
<tr>
<td>Manufacturer</td>
<td>-</td>
<td>Pittman</td>
<td>-</td>
</tr>
<tr>
<td>Torque Constant</td>
<td>(K_t)</td>
<td>0.0379</td>
<td>Nm/A</td>
</tr>
<tr>
<td>Motor Resistance</td>
<td>(R_a)</td>
<td>2.49</td>
<td>(\Omega)</td>
</tr>
<tr>
<td>Motor Inductance</td>
<td>(L_a)</td>
<td>2.63</td>
<td>mH</td>
</tr>
<tr>
<td>Rotor Inertia</td>
<td>(J_m)</td>
<td>7.1x10(^{-6})</td>
<td>kgm(^2)</td>
</tr>
<tr>
<td>Back EMF constant</td>
<td>(K_e)</td>
<td>0.0458</td>
<td>V/rad/s</td>
</tr>
<tr>
<td>Friction torque</td>
<td>(T_f)</td>
<td>0.055</td>
<td>Nm</td>
</tr>
<tr>
<td>Gear ratio</td>
<td>(GR)</td>
<td>19.7</td>
<td>-</td>
</tr>
<tr>
<td>Gearbox efficiency</td>
<td>(\eta)</td>
<td>0.84</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4-7 – Parameters of the H-Bot

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radius of pulleys 5, 6, 7, and 8</td>
<td>(r_p)</td>
<td>0.0222</td>
<td>m</td>
</tr>
<tr>
<td>Radius of pulleys 1, 2, 3, and 4</td>
<td>(r)</td>
<td>0.02136</td>
<td>m</td>
</tr>
<tr>
<td>Inertia of pulleys 5, 6, 7, and 8</td>
<td>(J_5, J_6, J_7, J_8)</td>
<td>8.9x10(^{-6})</td>
<td>kgm(^2)</td>
</tr>
<tr>
<td>Inertia of pulleys 1, 2, 3, and 4</td>
<td>(J_1, J_2, J_3, J_4)</td>
<td>2.908x10(^{-5})</td>
<td>kgm(^2)</td>
</tr>
<tr>
<td>Cart mass plus load mass</td>
<td>(m_{cart})</td>
<td>3.077</td>
<td>kg</td>
</tr>
<tr>
<td>Bridge mass</td>
<td>(m_{bridge})</td>
<td>1.86</td>
<td>kg</td>
</tr>
<tr>
<td>Viscous coefficient of friction at pulley 1 and 2</td>
<td>(B_{m1}, B_{m2})</td>
<td>0.05</td>
<td>Nms</td>
</tr>
<tr>
<td>Viscous friction coefficient in x-direction</td>
<td>(B_x)</td>
<td>3.8</td>
<td>Ns/m</td>
</tr>
<tr>
<td>Viscous friction coefficient in y-direction</td>
<td>(B_y)</td>
<td>13.8</td>
<td>Ns/m</td>
</tr>
</tbody>
</table>

4.2.7.1 Radius of Pulleys \((r_p, r)\)

The radius of all pulleys were obtained from the 3D model design of the H-Bot.

4.2.7.2 Inertia of Pulleys \((J_1 \text{ to } J_8)\)

The inertia of all pulleys were obtained from the 3D model design of the H-Bot.
4.2.7.3 Cart Mass plus Load Mass ($m_{cart}$) and Bridge Mass ($m_{bridge}$)

The cart mass plus the load mass ($m_{cart}$) and the bridge mass ($m_{bridge}$) were obtained from the 3D model design of the H-Bot.

4.2.7.4 Viscous coefficient of friction at pulleys 1 ($B_{m1}$) and 2 ($B_{m2}$)

The viscous coefficient of friction at pulleys 1 ($B_{m1}$) and 2 ($B_{m2}$) were obtained from the simulation of the system described in the next chapter until a match with experimental results was obtained.

4.2.7.5 Viscous Friction Coefficient in x-Direction ($B_x$)

The viscous friction coefficient in the x-direction ($B_x$) was obtained experimentally. The H-Bot was commanded to move in the x-direction as shown in Fig. 4-15 at various constant linear velocities $v_x$ while the motor current was measured. The force in x-direction $F_x$ was computed as follows:

$$F_x = (i_m - i_C) \frac{K_t G R \eta}{r}$$  \hspace{1cm} (55)

Where, $i_m$ is the sum of the motor current to move the H-Bot in x-direction at constant speed, and $i_C$ is the sum of current of both motors to overcome the Coulomb friction of the system. The viscous friction coefficient in x-direction $B_x$ was then calculated as follows:

$$B_x = \frac{F_x}{v_x}$$  \hspace{1cm} (56)

This experiment was repeated for various velocities $v_x$ and the average value of $B_x$ entered in Table 4-7.
4.2.7.6 Viscous Friction Coefficient in y-Direction ($B_y$)

The same experimental procedure to obtain $B_x$ was used to obtain $B_y$, but commanding the H-Bot to move in y-direction.

4.3 Implementation of a Self-Balancing Transporter

The implementation of a self-balancing transporter in respect to the design and programming of the processors, sensors, control system, tuning, power structure, and electrical circuit is described in this section.

4.3.1 Processor

The same approach used with the H-Bot and described in Section 4.2.1 was also used with the self-balancing transporter in regards to the processor.

4.3.2 Sensors

There are three types of sensors in the self-balancing transporter. A gyroscope and an accelerometer are used in a sensor fusion configuration to measure the tilt angle of the pendulum, while two encoders, one in each motor, are used to measure the speed of each wheel. The implementation of these sensors in the self-balancing transporter is described next.

4.3.2.1 Gyroscope

The gyroscope is a sensor to measure angular velocity in one, two or three axes. The gyroscope used in this experiment was the LPY503AL [106] mounted in a breakout board as shown in Fig. 4-16 [39]. This dual axis gyroscope measures angular velocity
along the X and Z axes with a full scale of ±30°/s. The X and Z angular velocity are provided in two sets of analog outputs: not amplified (x1) and four-times amplified (x4) as shown below [106]. The direction of the detectable angular rates on the breakout board is indicated in Fig. 4-16.

![Gyro Breakout Board LPY503AL](image)

Fig. 4-16 – Gyro Breakout Board LPY503AL

Some of the electrical characteristics of the gyroscope LPY503AL are as follows [106]:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply voltage</td>
<td>Min: 2.7; Typ: 3.3; Max: 3.6</td>
<td>V</td>
</tr>
<tr>
<td>Supply current</td>
<td>6.8</td>
<td>mA</td>
</tr>
<tr>
<td>Measurement range</td>
<td>± 30 (4x amplified output)</td>
<td>degree/sec</td>
</tr>
<tr>
<td></td>
<td>± 120 (not amplified output)</td>
<td>degree/sec</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>± 33.3 (4x amplified output)</td>
<td>mV/deg/sec</td>
</tr>
<tr>
<td></td>
<td>± 8.3 (not amplified output)</td>
<td>mV/deg/sec</td>
</tr>
<tr>
<td>Zero-rate output voltage</td>
<td>1.23</td>
<td>V</td>
</tr>
<tr>
<td>level</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bandwidth</td>
<td>140</td>
<td>Hz</td>
</tr>
</tbody>
</table>
The analog outputs of this sensor behave as follows:

- At zero angular velocity, the output X and output Z provide 1.23V
- The non-amplified analog outputs X and Z provide a voltage range of:
  \[ 1.23V + \text{Sensitivity} \times (\pm \text{Measurement range}) = \]
  \[ 1.23V + 8.3 \frac{\text{mV}}{\text{deg/s}} \times (\pm 120\text{deg/s}) = 0.234V \rightarrow 2.23V \]
- The four-time amplified analog outputs X and Z provide a voltage range of:
  \[ 1.23V + \text{Sensitivity} \times (\pm \text{Measurement range}) = \]
  \[ 1.23V + 33.3 \frac{\text{mV}}{\text{deg/s}} \times (\pm 30\text{deg/s}) = 0.231V \rightarrow 2.23V \]

The non-amplified analog output Z of the gyroscope with voltage that can vary from 0.23V to 2.23V was selected to be used in the tilt angle measurement process for the self-balancing transporter. The analog output of the gyroscope is connected to a 10-bit 0-5V analog input of an Arduino Mega board [98, 99]. At zero velocity, the gyroscope voltage is 1.23V. In order to measure positive and negative angular rate, the offset of 1.23V at zero velocity can be compensated in Simulink code as shown in Fig. 4-17. This offset of 1.23V is 252 in digital values (\(2^{10} \times 1.23V/5V = 252\)) as shown in this figure. Meanwhile, the gyroscope voltage connected to the analog input of the Arduino board can be converted to deg/sec with the factor 120/204 as shown in Fig. 4-17. This factor is derived as follows. When the gyroscope is connected to the 10-bit analog input of the Arduino board, the voltage from the gyroscope is converted to a digital value with 10-bit resolution (a number between 0 and 1023). At zero velocity, the gyroscope voltage is...
1.23V which is 252 in digital value. At the highest angular rate of 120 deg/sec, the gyroscope voltage is 2.23V which is 456 in digital value for a 10-bit analog input. Thus, 

\[
\frac{(120 \text{ deg/sec} - 0 \text{ deg/sec})}{(456-252)} = \frac{120}{204}
\]

which yields the factor to convert the gyroscope signal to deg/sec.

![Conversion of the gyroscope signal](image1)

**Fig. 4-17 – Conversion of the gyroscope signal**

4.3.2.2 Triple Axis Accelerometer

Accelerometers are used to measure gravitational acceleration. The accelerometer selected for the self-balancing transporter was the ±1.5g, ±6g three-axis low-g micro-machined accelerometer MMA7361L [107] mounted on a breakout board as shown in Fig. 4-18. The sensitivity of this accelerometer can be set to either ±1.5g or ±6g. The directions of acceleration detected by the accelerometer are on the breakout board MMA7361 [39] as shown in Fig. 4-18, where the arrows point on the positive direction:

![Accelerometer Breakout Board MMA7361L](image2)

**Fig. 4-18 - Accelerometer Breakout Board MMA7361L**
This accelerometer MMA7361L contains an input called g-Select to select the sensitivity between ±1.5g and ±6g. The self-balancing transporter was implemented with the sensitivity of the accelerometer set to ±1.5g.

For static acceleration, the analog output value for the three axes of the accelerometer behave as shown in Fig. 4-19 [107].

![Static Acceleration Diagram](image)

Fig. 4-19 – Behavior of the accelerometer for static acceleration [107].

Some of the electrical characteristics of the accelerometer MMA7361L are shown in Table 4-9 [107].
Table 4-9 – Electrical characteristics of the accelerometer MMA7361L

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply voltage</td>
<td>Min: 2.2; Typ: 3.3; Max: 3.6</td>
<td>V</td>
</tr>
<tr>
<td>Supply current</td>
<td>400 µA</td>
<td></td>
</tr>
<tr>
<td>Measure of range</td>
<td>± 1.5 (with g-Select input set to zero)</td>
<td>g</td>
</tr>
<tr>
<td></td>
<td>± 6.0 (with g-Select input set to one)</td>
<td>g</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>± 800 (with g-Select input set to zero)</td>
<td>mV/g</td>
</tr>
<tr>
<td></td>
<td>± 206 (with g-Select input set to one)</td>
<td>mV/g</td>
</tr>
<tr>
<td>Zero-rate output voltage level</td>
<td>1.65</td>
<td>V</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>400 (for X and Y axes)</td>
<td>Hz</td>
</tr>
<tr>
<td></td>
<td>300 (for Z-axis)</td>
<td></td>
</tr>
</tbody>
</table>

The analog outputs of this accelerometer shown in Table 4-9 behave as follows:

- At zero acceleration of gravity, the outputs X, Y and Z provide 1.65V.

- With g-select set to zero, the outputs X, Y, and Z provide a voltage range of:

\[
1.65V + \text{Sensitivity} \times (\pm \text{Measurement range}) =
\]

\[
1.65V + 800 \frac{\text{mV}}{g} \times (\pm 1.5g) = 0.450V \rightarrow 2.850V
\]

- With g-select set to one, the outputs X, Y, and Z provide a voltage range of:

\[
1.65V + \text{Sensitivity} \times (\pm \text{Measurement range}) =
\]

\[
1.65V + 206 \frac{\text{mV}}{g} \times (\pm 6g) = 0.414V \rightarrow 2.886V
\]

The Y-axis analog output was selected to be used in the tilt angle measurement process for the self-balancing transporter. As shown above, this analog output that can vary from 0.45V to 2.85V while the input “g-select” is set to zero. The analog output of the gyroscope is connected to a 10-bit 0-5V analog input of an Arduino Mega board. At zero acceleration, the voltage on Y-axis analog output is 1.65V. In order to measure
positive and negative acceleration, the offset of 1.65V at zero acceleration can be
compensated in Simulink code as shown in Fig. 4-20. The offset of 1.65V is 338 in
digital values \(2^{10} \times 1.65V/5V = 338\) as shown in the Simulink code in Fig. 4-20. This
zero DC offset needs to be measured in the physical sensor for proper compensation.
Meanwhile, accelerometer voltage connected to the analog input of the Arduino board
can be converted to “g” with the factor 1/164 as shown in Fig. 4-20. This factor is derived
as follows. When the accelerometer is connected to the 10-bit analog input of the
Arduino board, the voltage from the accelerometer is converted to a digital value with 10-
bit resolution. At zero acceleration, the accelerometer voltage is 1.65V which is 338 in
digital value. Since the sensitivity of the accelerometer is 800mV/g, at an acceleration of
1g, the accelerometer voltage increases by 800mV to 2.45V which is 502 in digital value
for a 10-bit analog input. Thus, \((1g - 0 g) / (502-338) = 1 / 167\) which yields the factor to
convert the accelerometer signal to “g”.

![Fig. 4-20 – Signal conversion of the accelerometer MMA7361L](diagram)

Since the self-balancing transporter operates about the vertical axis, the Y-axis of
the accelerometer in Fig. 4-18 is more sensitive to tilt angle \(\theta\) (see Fig. 3-29) than the X-
axis assuming the X and Y orientations shown in Fig. 4-18, which justifies the use of the
Y-axis in the tilt angle measurement. The higher sensitivity of the Y-axis about a vertical orientation of the self-balancing transporter is demonstrated in Fig. 4-21. When the accelerometer is in the vertical position as shown in Fig. 4-21a, X-axis reads 1g while Y-axis reads 0g. If the angle is changed by 5 degrees for example as shown in Fig. 4-21b, the Y-axis will read $G_y = \sin(5) \times 1g = 0.087g$ while the X-axis will read $G_x = \cos(5) \times 1g = 0.996g$. Thus, the change in gravitational component in Y-axis is much larger than the change in gravity in X-axis about the vertical axis. This makes the Y-axis the most sensitive axis to measure the tilt angle about the vertical axis. In Fig. 4-20, the Y-axis is connected to the analog input of the Arduino board, and then the high frequency noise is filtered, the dc offset is removed, the tilt angle is calculated in radians with the $\sin^{-1}$ function and converted to degrees with the factor $180/\pi$.

![Fig. 4-21 – Behavior of the accelerometer for tilt angle set to zero (a) and then to five degrees (b).](image)

4.3.2.3 Accelerometer and Gyroscope Location and Orientation

The accelerometer and the gyroscope were placed on the self-balancing transporter as shown in Fig. 4-22.
Fig. 4-22 – Location of the gyroscope and accelerometer on the self-balancing transporter.

The orientation of the sensors in Fig. 4-22 was chosen to use the Y-axis of the accelerometer and the non-amplified Z-axis output of the gyroscope.

4.3.2.4 Sensor Fusion

The sensor fusion combines the signals from the accelerometer and gyroscope to obtain the tilt angle as shown in Fig. 4-23.

The gravitational acceleration measured by the Y-axis of the accelerometer and converted to angular position $\theta_A$ can be used to measure an angle between 0 and ±90 degrees. Accelerometers have high frequency noise components that need to be attenuated. Thus, a low-pass filter (LPF) is used to reduce the high frequency noise from the accelerometer. The filtered angular position measured by the accelerometer is called $\theta_{AF}$ in Fig. 4-23.

The angular rate about the Z axis, $\omega_z$, of the gyroscope was integrated to obtain the angular position $\theta_G$. However, the gyroscope signal drifts at low frequencies which can accumulate errors over time if not compensated. Thus, a high-pass filter (HPF) is
used to mitigate the low frequency drift from the gyroscope as shown in Fig. 4-23. The filtered angular position measured by the gyroscope is called $\theta_{GF}$ in this figure.

![Diagram of sensor fusion with accelerometer and gyroscope to compute the tilt angle.](image)

Fig. 4-23 – Sensor fusion with accelerometer and gyroscope to compute the tilt angle.

The sum of the filtered tilt angle from the accelerometer ($\theta_{AF}$) and the gyroscope ($\theta_{GF}$) yield the tilt angle as shown in Fig. 4-23. These filters were implemented in discrete form. The derivation of the sensor fusion approach shown in Fig. 4-23 with discrete filters is given in Appendix C. The discrete tilt angle $\theta_i$ obtained from this derivation is given as follows:

$$\theta_i = (\theta_{Gi} - \theta_{Gi-1})\beta + \theta_{i-1} + \theta_{Ai}(1 - \beta)$$  \hspace{1cm} (57)

Where, $\theta_i$ is the actual tilt angle, $\theta_{i-1}$ is the tilt angle in the previous time scan, $\theta_{Gi}$ is the actual tilt angle from the gyroscope, $\theta_{Gi-1}$ is tilt angle from the gyroscope in the previous time scan, $\theta_{Ai}$ is the actual tilt angle from the accelerometer, and $\beta$ is defined as follows:

$$\beta = \frac{\tau}{\tau + dt}$$  \hspace{1cm} (58)
Where $dt$ is the sampling time of accelerometer and gyroscope signals, and $\tau$ is time constant of the filters. In general, $\beta$ is chosen between 0.95 and 0.99.

The implementation of this sensor fusion is shown in Fig. 4-24.

![Fig. 4-24 – Implementation of the sensor fusion in Simulink.](image)

### 4.3.2.5 Encoder

In this self-balancing transporter, there is a 500 PPR incremental encoder in each motor. An Arduino Mega was used exclusively for the decodification of the two encoders as described for the H-Bot in Section 4.2.2

### 4.3.3 Power Structure Design

The power control of the self-balancing transporter is the same used with the H-Bot as shown in Section 4.2.3. However, a subsystem for “Tilt Angle Protection” was included as shown in Fig. 4-25 to turn the motors off if the tilt angle is greater than 20 degrees since any angle greater than 20 degrees was considered an unsafe condition.
4.3.4 Control System and Tuning

The balancing control of the self-balancing transporter is a state-space control as described in Section 3.3.10.

The tuning of the control gain $K$ given in (45) can be either obtained by resolving the Riccati equation in (46) with the LQR function in Matlab or defined experimentally. As experienced by many researchers, the tuning of $K$ via LQR function for unstable systems such as self-balancing transporters, requires exhaustive testing to identify the weighting matrices $Q$ and $R$ that will yield a $K$ control gain capable of providing the required performance with the physical system [93, 108-110]. This exhaustive testing is in general due to the differences between the mathematical model of the system and the actual system. The configuration of the sensor fusion for example has high impact in the
balancing performance of the system. If the modeling of the sensor fusion is not included in calculation of the $K$, the simulated performance may not be obtained with the physical system.

Since the self-balancing transporter is an unstable system, the ability to obtain the control loop gains via LQR function become sensitive to how well the model represents the actual system. Differences between the models and the physical system are due to simplifications, assumptions, and errors in parameter identification. In order to avoid numerous trial-and-error tests, a manual tuning procedure was developed to quickly determine the control gain $K$ to properly balance the system. This also allows to gain further understanding of the behavior of the system with respect to the control system.

The control gain $K$ for the physical self-balancing transporter was defined as a function of the tilt angle $\theta$, tilt angular velocity $\dot{\theta}$, linear displacement $x$, and linear velocity $\dot{x}$, i.e., $K(\theta, \dot{\theta}, x, \dot{x})$. The proposed manual tuning procedure to set the control loop gain $K$ is described below. In this procedure, each element of the vector $K(\theta, \dot{\theta}, x, \dot{x})$ is referred to as $K(\theta)$, $K(\dot{\theta})$, $K(x)$, and $K(\dot{x})$.

- **STEP 1** - this step is to set $K(\theta)$ in order to balance about $\theta = 0$:
  - The gain $K$ is set to zero ($K(\theta, \dot{\theta}, x, \dot{x}) = [0, 0, 0, 0]$)
  - The wheels are locked on the floor to avoid any motion in $x$-direction while the pendulum is placed in an inclined position; for example, 10 degrees.
  - The gain $K(\theta)$ is increased until the pendulum moves to $\theta = 0$ degree by itself due to the $K(\theta)$ effect and stay balancing about $\theta = 0$. This is the minimum value for $K(\theta)$.
  - The wheels are then released. The transporter should balance by itself at this point, although it may walk and present some instability. If the system is
unstable with $K(\theta)$ only while performing this test, then a small initial value for $K(\dot{\theta})$ can be set before performing Step 1.

- The gain $K(\theta)$ is then continuously increased until it presents an unstable behavior to a moderate disturbance applied to the pendulum. This is the maximum value for $K(\theta)$.

- The gain $K(\theta)$ is then reduced from this maximum value to the point that the bouncing (oscillations) of the pendulum to an external disturbance applied to the pendulum causes a damped behavior instead of an unstable behavior, i.e., the bouncing needs to reduce to zero in a few seconds. When an external disturbance is applied to the pendulum, the bouncing of the pendulum should damp to zero although the transporter still may walk.

NOTE 1: The function of $K(\theta)$ is to make the pendulum balance about $\theta = 0$.

NOTE 2: The sensor fusion should be adjusted before performing Step 1 in order to measure $\theta = 0$ when the pendulum is physically at zero degree.

• STEP 2 - this step is to set $K(\dot{\theta})$ to improve control robustness and reduce walking:

- The gain $K(\dot{\theta})$ is progressively increased starting from zero. As $K(\dot{\theta})$ increases, the balancing control becomes stiffer, i.e., the control presents a quicker response to disturbance applied to the pendulum and the swinging of the pendulum to an external disturbance reduces. This can be easily observed while holding the top of the pendulum and observing its reaction to any disturbance. As $K(\dot{\theta})$ increases, the response of the control system to the disturbances applied to the pendulum improves.

- As $K(\dot{\theta})$ is increased even further, the self-balancing transporter will present an unstable behavior to an external disturbance applied to the pendulum. This identifies the maximum value for $K(\dot{\theta})$.

- The gain $K(\dot{\theta})$ is then reduced to the point that the unstable behavior disappears.

NOTE: The function of $K(\dot{\theta})$ is to obtain a stiff control and possibly reduce the walking effect. With $K(\theta)$ and $K(\dot{\theta})$ tuned, the self-balancing transporter should
balance about $\theta = 0$, and have good disturbance rejection, although it may still walk.

- **STEP 3** – This step is to set $K(\dot{x})$ to reduce the walking:
  
  NOTE: Once $K(\theta)$ and $K(\dot{\theta})$ are set, then $K(\dot{x})$ and $K(x)$ will be set to reduce the walking. The gain $K(\dot{x})$, which is associated with $\dot{x}$, is tuned first. When the self-balancing transporter is balancing due to $K(\theta)$ and $K(\dot{\theta})$, it can walk back and forth or in a single direction. If $K(x)$ is tuned before $K(\dot{x})$, the system can simply start walking away.
  
  o The gain $K(\dot{x})$ is increased from zero until the system become unstable to an external disturbance applied to the pendulum. This is the maximum value for $K(\dot{x})$.
  
  o The gain $K(\dot{x})$ is then reduced from this maximum value until it becomes stable to an external disturbance applied to the pendulum. Even aggressive disturbances should not make the system unstable or fall.

- **STEP 4** – This step is to set $K(x)$ to control how far the walking is allowed:
  
  NOTE: Only with $K(\dot{x})$, the walking is minimized, but is can still continue walking away from $x = 0$, but significantly less than with just $K(\theta)$ and $K(\dot{\theta})$. The gain $K(x)$ is then set to control how far the walking will be allowed.
  
  o The gain $K(x)$ is increased until the self-balancing transporter starts moving back and forth further and further after a moderate external disturbance applied to the pendulum. This is the maximum value for $K(x)$.
  
  o The gain $K(x)$ is then reduced from this maximum value until the back and forth move in $x$ damps out even with an aggressive external disturbance applied to the pendulum.

The implementation of the balancing control in Simulink is shown in Fig. 4-26.

The subsystem “Sensor Fusion” in this figure consists of the code in Fig. 4-24. The subsystems “Motor 1 – Actual Position and Velocity” and “Motor 2 – Actual Position and Velocity” is described in Appendix B. The subsystem “Psn Cmd” consists of the model in Fig. 4-27 that uses four digital inputs to select the different types of motion
profiles to be used as the position reference signal to move the self-balancing transporter in the $x$-direction. These motion profiles were implemented according to the equations in Section 2.6 and are also defined later in Section 5.3. The subsystem “Forward/Reverse and Left/Right Ctrl” was implemented as shown in Fig. 4-28. The subsystem “Power Control” is shown in Fig. 4-25.

Fig. 4-26 – Arduino Code for control of the self-balancing transporter.

Fig. 4-27 – Position reference generator for the self-balancing transporter. This model is “Pos Cmd” in Fig. 4-26
4.3.5 Electrical Circuit Design

The electrical circuit design for the transporter is shown in Fig. 4-29 and it includes:

- An Arduino Mega to decodify two encoders
- An Arduino Mega for control of the Transporter
- A 12V voltage regulator LM7812 to power both Arduino boards
- A 5V voltage regulator LM7805 to power the encoders and the AND gate (74LS08)
- A 24V, 3300mA/h rechargeable battery to power the system
- Two 5A full H-Bridges TLE5206 [103] to power each motor
- Quadruple 2-input AND gate in the 74LS08 [104] as the logics between the Arduino digital outputs and the inputs of the H-Bridges.
- Two 24V brushed gearmotor GM9236S021 from Pittman
- A gyroscope LPY503AL
- An accelerometer MMA7361L

Fig. 4-29 – Electrical diagram of the self-balancing transporter

The electrical diagram in Fig. 4-29 was mounted in a universal circuit board as shown in Fig. 4-30.
4.3.6 Final System

The assembled self-balancing transporter with the motors, control boards, Arduino boards, switches to select the motion profiles for the energy optimization methods in Chapter 5, potentiometers for tuning, and battery, is shown in Fig. 4-31.
4.3.7 Parameter Identification

The motor used with the self-balancing transporter was the same used with the H-Bot and it is shown in Table 4-6. The parameters of the self-balancing transporter are shown in Table 3-5 and copied below in Table 4-10 for convenience. The procedure to obtain each one of these parameters is given next.

Table 4-10 – Motor and system parameters for the self-balancing transporter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radius of wheels</td>
<td>$r$</td>
<td>0.15m</td>
</tr>
<tr>
<td>Mass of wheels</td>
<td>$m_w$</td>
<td>1.166kg</td>
</tr>
<tr>
<td>Inertia of wheels</td>
<td>$J_w$</td>
<td>0.0216 kgm²</td>
</tr>
<tr>
<td>Mass of pendulum</td>
<td>$m_p$</td>
<td>6.314 kg</td>
</tr>
<tr>
<td>Center of mass</td>
<td>$\bar{r}$</td>
<td>0.115m</td>
</tr>
<tr>
<td>Pendulum inertia</td>
<td>$J_p$</td>
<td>0.35kgm²</td>
</tr>
<tr>
<td>Damping</td>
<td>$B$</td>
<td>0.024Nm/rad/s</td>
</tr>
<tr>
<td>Friction torque</td>
<td>$T_f$</td>
<td>0.11Nm</td>
</tr>
</tbody>
</table>
4.3.7.1 Radius of the Wheel ($r$)

The radius of each wheel $r$ was measured from the physical system as 0.15 m.

4.3.7.2 Mass ($m_w$) and Inertia of the Wheels ($J_w$)

A single wheel was modeled in SolidWorks in order to compute the mass $m_w$ and inertia $J_w$ of the wheel. The mass of a single wheel was estimated in 0.583 kg. Thus, the mass of both wheels is $m_w = 2 \times 0.583 \text{ kg} = 1.166 \text{ kg}$. Meanwhile, the moment of inertia of a single wheel at the center of mass about the axis of rotation was estimated in 0.00803 kgm$^2$. The inertia of both gearmotors also need take into account to compute the total inertia associated with the wheel. As shown in Table 3-6, the inertia defined in the datasheet for the selected motor (Pittman GM9236S021) used with the self-balancing transporter is $7.1 \times 10^{-6}$ kgm$^2$, but the inertia of the 19.7:1 gearbox is unknown. Thus, the total inertia associated with the wheels was estimated as follows: $J_w = 2(J_mG R^2 + J_{wheel}) = 2(7.1 \times 10^{-6} \times 19.7^2 + 0.00803) = 0.0216 \text{ kgm}^2$.

4.3.7.3 Mass of the Pendulum ($m_p$)

The mass of the entire system was measured with a scale as 16.5 lb which corresponds to 7.48 kg. The mass of the wheels is 1.166 kg as defined above. Thus, the mass of the pendulum was defined as: $m_p = 7.48 \text{ kg} - 1.166 \text{ kg} = 6.314 \text{ kg}$.

4.3.7.4 Center of Mass of the Pendulum ($\bar{r}$)

The mass of the pendulum includes the mechanism that holds the motors, the motors, the mounting plates, the battery, and electronics. Since the pendulum is a complex mechanism to manually compute the center of mass, experimentally measures
were performed to determine the center of mass of the pendulum. By using a lever approach, the self-balancing transporter was placed horizontally on a thin support and balanced to find the center of mass as shown in Fig. 4-32. The distance from the center of rotation of the wheels to the center of mass of the pendulum $\bar{r}$ was at 0.115m.

![Fig. 4-32 – Approach to measure the center of mass of the pendulum](image)

4.3.7.5 Inertia of the Pendulum $(J_p)$

The moment of inertia of the pendulum about the center of mass $G$ $(J_p)$ was experimentally determined. The test consisted in measuring the natural frequency $\omega_n$ of the pendulum while swinging it upside down as shown in Fig. 4-33 to determine the moment of inertia of the pendulum about the center of mass $J_p$ [111].

![Fig. 4-33 - Pendulum moment of inertia test](image)
The derivation of the equations to compute the moment of the inertia of the pendulum was obtained from the free-body diagram shown in Fig. 4-33 by the sum of momentums as follows:

\[ \sum M_o = J_o \ddot{\theta} \]

\[-B \dot{\theta} - m_p g \bar{r} \sin \theta = J_o \ddot{\theta} \quad (59)\]

\[ J_o \ddot{\theta} + B \dot{\theta} + m_p g \bar{r} \theta = 0 \]

\[ \ddot{\theta} + \frac{B}{J_o} \dot{\theta} + \frac{m_p g \bar{r}}{J_o} \theta = 0 \]

From the characteristic equation:

\[ s^2 + 2 \xi \omega_n s + \omega_n^2 = 0 \quad (60)\]

The natural frequency \( \omega_n \) can be estimated as follows:

\[ \omega_n^2 = \frac{m_p g \bar{r}}{J_o} \quad (61)\]

Or:

\[ \omega_n = \sqrt{\frac{m_p g \bar{r}}{J_o}} \quad (62)\]

Assuming that \( \omega_n \) can be obtained, \( J_o \), the inertia of the pendulum about the origin \( O \), can be obtained from (61) as follows:

\[ J_o = \frac{m_p g \bar{r}}{\omega_n^2} \quad (63) \]
Additionally, using parallel axes theorem [77], $J_p$, the inertia of the pendulum at the center of mass $G$ can be calculated from $J_o$, as follows:

$$J_p = J_o - m_p \bar{r}^2$$  \hspace{1cm} (64)

Thus, the unknown variable to calculate $J_p$ is the natural frequency $\omega_n$ which can be experimentally determined as follows. The natural frequency $\omega_n$ can be calculated from the damped natural frequency $\omega_d$ that can be determined experimentally by holding the self-balancing transporter upside down by the wheels, raise the pendulum at a certain angle (about 30 degrees was used in this test), release the pendulum, and measure the oscillations (swinging). Several methods can be used to measure these oscillations. In this work, an accelerometer was chosen to measure these oscillations and the results are shown in Fig. 4-34.

Fig. 4-34 - Measurements to calculate moment of inertia of the pendulum of the self-balancing transporter.
The period, $T$, was obtained from the first two oscillations and it was measured as 1.55 seconds (3.15sec-1.60sec) as shown in Fig. 4-34. Thus, the damped natural frequency $\omega_d$ is as follows:

$$\omega_d = \frac{2\pi}{T}$$  \hspace{1cm} (65)

And, the natural frequency $\omega_n$ is:

$$\omega_n = \frac{\omega_d}{\sqrt{1 - \xi^2}}$$  \hspace{1cm} (66)

Where, $\xi$ is the damping ratio calculated as:

$$\xi = \frac{1}{\sqrt{1 + \left(\frac{2\pi}{\delta}\right)^2}}$$  \hspace{1cm} (67)

Where, $\delta$ is the logarithm decrement for the test shown in Fig. 4-34, and it is calculated as follows:

$$\delta = ln\frac{x_1(t)}{x_2(t + T)}$$  \hspace{1cm} (68)

Where, $x_1(t)$ is the peak value of the first oscillation while $x_2(t + T)$ is the peak value of the second oscillation as shown in Fig. 4-24.

This yielded:
• Logarithm Decrement $\delta = 0.043$

• Damping Ratio $\xi = 0.00684$

• Damped Natural Frequency $\omega_d = 4.0536 \text{ rad/s}$

• Natural Frequency $\omega_n = 4.0537 \text{ rad/s}$

Thus, assuming the following values:

• $m_p = 6.314 \text{ kg}$ as defined in Section 4.3.7.3

• $g = 9.81 \text{ m/s}^2$

• $\bar{r} = 0.115 \text{ m}$ as defined in Section 4.3.7.4

• $\omega_n = 4.05 \text{ rad/sec}$ as defined above

The inertia $J_o$ can be calculated as follows:

$$J_o = \frac{m_p g \bar{r}}{\omega_n^2} = \frac{(6.314 \text{ kg})(9.81 \text{ m/s}^2)(0.115 \text{ m})}{(4.05 \text{ rad/sec})^2} = 0.433 \text{ kgm}^2$$ (69)

Which yields $J_p$ as:

$$J_p = J_o - m_p \bar{r}^2 = 0.433 \text{ kgm}^2 - (6.314 \text{ kg})(0.115 \text{ m})^2 = 0.35 \text{ kgm}^2$$ (70)

4.3.7.6 Damping ($B$)

From (59) and (60), the damping can be derived as follows:

$$2\xi \omega_n = \frac{B}{J_o}$$ (71)

Which can be rewritten as:

$$B = 2\xi \omega_n J_o$$ (72)

This results in $B = 0.024 \text{ Nm/rad/s}$. 
4.3.7.7 Friction Torque ($T_f$)

The friction torque is the friction between the wheels and the pendulum. The element connecting the wheels and the pendulum is the motor. Thus, the friction torque was assumed to be the sum of the friction torque of each motor. As shown in Table 3-6, the friction torque of the selected motor is 0.055Nm. Thus, the friction torque $T_f = 0.11$Nm.

4.3.8 RF Control for Maneuvering

In cases when the self-balancing transporter requires left/right and forward/reverse control, a remote control system can be used. A low cost solution was develop and implemented for the self-balancing transporter as described next.

The left and right command signal is connected to input 4 in Fig. 4-28, while the forward and reverse command signal from the remote control is connected to where the subsystem “Psn Cmd” is connected in Fig. 4-26. Alternatively, the forward and reverse command can be applied to a sum block that can be added to the connecting line named Theta in Fig. 4-26. Thus, a two-channel remote control system is required: one channel for left/right control and one channel for forward/reverse control. An approach was then developed to emulate a two-channel remote control from a single channel receiver/transmitter.

A low cost RF transmitter and receiver were used for the remote control of forward/reverse and left/right commands. The two-axis joystick shown in Fig. 4-35 was used as the reference signal for forward/reverse and left/right commands sent via RF signal from the transmitter to the receiver. The transmitter is mobile and it was powered by a 9V battery.
Fig. 4-35 - Two-axis analog joystick Parallax 27800 with independent 10kΩ potentiometers with common ground and springs to return automatically to the center position

4.3.8.1 Receiver

The receiver used with the self-balancing transporter was the low cost Wireless Hi Sensitivity Receiver Module RWS-371-6 from Wenshing shown in Fig. 4-36 [112].

Fig. 4-36 – Wireless Hi Sensitivity Receiver Module RWS-371-6 from Wenshing [112]

The main features of this receiver are as shown in Table 4-11.

Table 4-11 – Parameters of the Wireless Hi Sensitivity Receiver Module RWS-371-6

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency Range</td>
<td>433.92</td>
<td>MHz</td>
</tr>
<tr>
<td>Modulate Mode</td>
<td>Amplitude-shift keying (ASK)</td>
<td>-</td>
</tr>
<tr>
<td>Date Rate</td>
<td>4800</td>
<td>bps</td>
</tr>
<tr>
<td>Selectivity</td>
<td>-108</td>
<td>dBm</td>
</tr>
<tr>
<td>Channel Spacing</td>
<td>±500</td>
<td>kHz</td>
</tr>
<tr>
<td>Supply Voltage</td>
<td>5</td>
<td>V</td>
</tr>
</tbody>
</table>
The pinout is shown in Fig. 4-37.

![Pinout of the Wireless Hi Sensitivity Receiver Module RWS-371-6 from Wenshing](image)

**Fig. 4-37 – Pinout of the Wireless Hi Sensitivity Receiver Module RWS-371-6 from Wenshing [112]**

### 4.3.8.2 Transmitter

The transmitter used with the self-balancing transporter was the low cost Wireless Hi Power Transmitter Module TWS-BS-3 from Wenshing shown in Fig. 4-38 [4] which is the pair for the receiver in Section 4.3.8.1.

![Wireless Hi Power Transmitter Module TWS-BS-3](image)

**Fig. 4-38 - Wireless Hi Power Transmitter Module TWS-BS-3 from Wenshing [4]**

The main features of this module are shown in Table 4-12.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency Range</td>
<td>433.92 MHz</td>
<td></td>
</tr>
<tr>
<td>Modulate Mode</td>
<td>Amplitude-shift keying (ASK)</td>
<td></td>
</tr>
<tr>
<td>Date Rate</td>
<td>8000 bps</td>
<td></td>
</tr>
<tr>
<td>Supply Voltage</td>
<td>1.5~12 V</td>
<td></td>
</tr>
<tr>
<td>Output Power</td>
<td>14 dBm</td>
<td></td>
</tr>
</tbody>
</table>

**Table 4-12 – Parameters of the Wireless Hi Power Transmitter Module TWS-BS-3**
The pinout of this RF receiver is shown in Fig. 4-39.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GND</td>
</tr>
<tr>
<td>2</td>
<td>Data in</td>
</tr>
<tr>
<td>3</td>
<td>Vcc</td>
</tr>
<tr>
<td>4</td>
<td>ANT</td>
</tr>
</tbody>
</table>

Fig. 4-39 – Pinout of Wireless Hi Power Transmitter Module TWS-BS-3 from Wenshing [4]

Thus, it can be observed from the main features of the transmitter and receiver in Table 4-11 and Table 4-12, respectively, that they operate on the same frequency of 433.93MHz. The supply voltage of this transmitter can be from 1.5V to 12V, but the strength of the RF signal increases with the supply voltage, and better reception by the receiver is obtained when the transmitter is powered at higher voltages within the voltage range.

4.3.8.3 Wiring

The joystick shown in Fig. 4-35 was used for the forward/reverse and left/right commands. The two potentiometers of this joystick are connected to two analog inputs of an Arduino Uno board [98] as shown in Fig. 4-40a. The signals from the analog inputs were scaled and sent to the transmitter that is connected to the TX channel (pin 1) of the Arduino board. The transmitter is powered by the on-board 5V output from the Arduino board. The receiver is connected to the RX channel (pin 0) of the Arduino Mega board that controls the self-balancing transporter. The receiver is powered by the Arduino Mega board. The wiring for the receiver is shown in Fig. 4-40b.
At the center position, the joystick is at the center of each potentiometer resistance. These potentiometers are connected as a voltage divider. Thus, the voltage of the potentiometers at the center position is about 2.5V. For backward move, the voltage in pin 1 can go from 2.5V to zero. Meanwhile, the voltage in this pin can go from 2.5V to 5V for forward move. The same applies for left/right move command.

4.3.8.4 Arduino code

The Arduino code developed in Simulink for the RF transmitter is shown in Fig. 4-41. The receiver and transmitter have a single RF channel of data. But, two channels would be required to transmit the left/right and forward/reverse signals. In order to transmit two signals simultaneously through a single RF channel, the Simulink code shown in Fig. 4-41 was developed.
The RF channel can transmit one byte of data each time. The most significant bit was used to define which signal (forward/reverse or right/left signal) is being transmitted. When the most significant bit is one, the forward/reverse signal is transmitted. When the most significant bit is zero, the left/right signal is transmitted. Therefore, this most significant bit was used as a selection bit. Thus, the actual forward/reverse and right/left signal is transmitted with the remaining seven bits. The factor 255/1024 shown in Fig. 4-41 is to convert the 10-bit signal from the analog input of the Arduino board into an 8-bit signal. Meanwhile, the factor 0.5 on “Gain2” and “Gain3” is to convert these 8-bit signals into 7-bit signal for the RF transmission along with the selection bit.

The serial port for the transmitter in Fig. 4-41 was set to 600 baud. The data rate of the transmitter is up to 8000 bps (bits per second) while the data rate of the receiver is up to 4800 bps. Thus, the limiting factor is the 4800 bps from the receiver. However, it was observed experimentally that consistent and reliable transmission and reception would be achieved when the baud rate was set to no more than 300 baud. This imply that 8 bits take 8 bits/300bps = 26.7ms to be transmitted. It was observed that if the dwell time between transmissions is less than the transmission time, then the transmission was very reliable. Thus, the Sample Time of the “Serial Write” block was set to 40ms, which leaves about 13.3 ms for dwell.

The “Index Vector” shown in Fig. 4-41 selects the signal to be transmitted at a rate given by the “Pulse Generator” which is set to a period that is twice Sample Time of the Serial Write block. Thus, every 40ms either the forward/reverse or right/left signal is transmitted to the RF Receiver.
Fig. 4-41 – Arduino code for the transmitter.

The Arduino code for the RF receiver is shown in Fig. 4-42. The “Serial Read” block in this figure reads the 8-bit serial signal from the RF Receiver. The Sample Time in the blocks “Write Serial” in Fig. 4-41 and “Serial Read” in Fig. 4-42 were set to 40ms. Thus, data is transferred from the transmitter to the receiver every 40ms. This 8-bit signal consists of a selection bit (most significant bit) plus the 7-bit signal. If the selection bit is 1, the signal received via serial channel is from a forward/reverse command and this signal pass through the block “Product” in Fig. 4-42. If the selection bit is 0, the signal received via serial channel is from a left/right command and this signal pass through the block “Product1” in Fig. 4-42, while the output of the block “Product” remains in zero.
during this 40ms that the left/right command signal is being received. Thus, the signal at the output of the product blocks will be varying between the actual signal and zero as shown below.

The selection bit is detected by the comparison blocks in Fig. 4-42. Since an 8-bit signal can vary from 0 to 255, if the most significant bit is one, the serial signal will be equal or greater than 127, which is detected by the comparator block “>=127” in the 

Fig. 4-42 – Arduino code for the receiver
Simulink code shown in this figure. Similar analysis applies when the selection bit is zero.

The left/right command signal and the forward/reverse signal are rebuilt with the “Trigger Subsystem” block which is triggered by a comparison block that detects the transitions of the retrieved signal and updates its output only on the transitions from zero. An example of a rebuilt signal is shown in Fig. 4-42. The final assembly of the transmitter is shown in Fig. 4-43.

Fig. 4-43 – Remote control (transmitter) for the self-balancing transporter.

4.4 Summary

In this chapter, the physical implementation of the two-inertia system, H-Bot, and the self-balancing transporter was detailed. The two-inertia system was implemented with industrial equipment including a high-performance servo drive, a permanent-magnet synchronous motor, and an industrial controller. The hardware and programs to
implement H-Bot and the self-balancing transporter were based on fundamental concepts, including methods to decodify encoders, control of power delivered to motors via H-Bridges and PWM control, design of control systems, and conditioning of the sensor signals via digital signal processing.

These three systems will be used in the experimental validation of the energy optimization methods described in the next chapter.
CHAPTER 5

Energy Optimization Methods

Methodologies to optimize the energy consumption of industrial machines is fundamentally important to promote a sustainable industrial sector by reducing electrical energy consumption and consequently reducing production costs. This motivated the development of energy optimization methods in this dissertation that can be applied to industrial machines. These methods are developed for any type of industrial machine and are demonstrated for three systems: a two-inertia system, a Cartesian two-axis planar robot – H-Bot, and a self-balancing transporter. The method demonstrated with the two-inertia system is for point-to-point type of applications where only the initial position, final position, and move time of the desired move are provided. The method demonstrated with the H-Bot and the self-balancing transporter is for systems that need to follow a certain path within a pre-defined boundary region from the initial position to the target position.

The developed energy optimization methods optimize the motor electrical energy usage by designing a motion profile to be used as the reference signal in a closed-loop system to drive the motor and load from initial position to final position or along a path. As the motor and load follow this commanded motion profile to perform the desired move, the motor energy is minimized. If the optimized motion profile is replaced by other motion profiles as the reference signal to the servo drive controlling the motor to perform the same desired move, the motor energy consumption will be higher. The reference
signal is the only modification to the system to implement this energy optimization method. No mechanical or electrical changes are necessary. For a single axis system, the motion profile for an index move is optimized. For a multi-axis coordinated system, as a robot for example, the trajectory on the Cartesian space is optimized.

For a linear time-invariant system defined as \( \dot{x} = Ax + Bu \), this energy optimization method calculates the control effort \( u(t) \) to drive the system from the initial position to the target position by minimizing a cost function defined in terms of energy. This cost function is defined as the motor energy, which is the integral of motor power during a machine cycle. The motor power is the product of motor voltage and motor current. Thus, the cost function accounts for the energy cost to drive the system from the initial position to the final position during a machine cycle. Thus, this energy optimization method minimizes the energy cost during a machine cycle by minimizing a cost function defined in term of motor energy. However, this energy optimization method does not use the optimized control effort \( u(t) \) to optimize energy usage in machines for the following reason. This method is targeted for industrial machines, and the motors in industrial machines are powered by servo drives or variable-frequency drives (VFD’s) with embedded control loops for current, velocity, and position. Some low cost VFD’s may not have a position loop. Thus, the reference signal that controls these drives is not a control-effort signal, but either a position or velocity reference signal. The control effort is generated by the embedded control loops in the drives. Thus, instead of using the optimized control effort \( u(t) \) from this energy optimization method to control the system, the optimized position \( \theta(t) \), one of the states \( x \) of the system, is used as the position reference signal to be applied to the drive. This allows the system to follow an optimized
position-reference signal to drive the system from the initial position $x_0$ to the final position $x_f$ in a pre-defined amount of time $T$ and minimize the motor energy consumption. This approach is depicted in Fig. 5-1 where $C$ is the controller, $P$ is the plant, and $H$ is the feedback sensor.

This principle was applied to the energy optimization methods described and demonstrated next with a two-inertia system, a Cartesian two-axis planar robot – H-Bot, and a self-balancing transporter.

![Fig. 5-1 – Overview of the energy optimization method](image)

5.1 Two Inertia System

In this section, the proposed energy optimization method for single-axis systems performing point-to-point moves is described. This method is demonstrated with the two-inertia system described in Sections 3.1 and 4.1.
This energy optimization method optimizes the electrical energy that a motor consumes during motion. The motor energy is optimized by designing a motion profile that optimizes the motor energy to move the load from the initial position $x_0$ to the target position $x_f$ in a pre-defined amount of time $T$. If the load changes, the optimized motion profile needs to be recalculated. Thus, this method relies solely on the motion profile to optimize the energy of the system. No mechanical changes are required. Additionally, this method optimizes the shape of the motion profile without altering the move time defined in the machine specification. Since the implementation of this energy optimization method consists in updating the reference signal of the servo drive controlling the motor with the optimized motion profile, the implementation of this method in industrial machines is reduced to machine-code changes. This method takes into account the mechanical system in the design of the motion profile that optimizes energy. The mechanical system is entered into this method in state-space form. Therefore, the inputs to this energy optimization method are the target position, the move time, and state-space equation describing the mechanical system.

In this optimization method, a cost function defined in terms of the motor electrical energy is minimized to obtain the optimal control effort $u(t)$ that drives the system from the initial position to the target position with minimum energy cost. The cost function is defined as the electrical energy of the motor during a machine cycle. The motor electrical energy is the integral of the motor power, which is the instantaneous product of motor voltage and motor current. Since the cost function to be minimized is defined in terms of energy, then energy is minimized. Along with the control effort $u(t)$, the states of the system are also optimized. One of the optimized states is the motor
angular position. This optimized motor angular position is then used as the position reference in industrial machines to control the motion of the motor and optimize energy usage.

The procedures to obtain the state-space equation of the two-inertia system, define the cost function in terms of motor electrical energy, and calculate the optimized motion profile through the optimization method are described below.

5.1.1 State-Space Equation

For the two-inertia system, this energy optimization method optimizes the control effort \( u(t) \) for a system constituted by the following states: motor angular position \( \theta_m(t) \), motor angular velocity \( \dot{\theta}_m(t) \), load angular position \( \theta_l(t) \), motor angular velocity \( \dot{\theta}_l(t) \), and motor current \( i_a(t) \). This method minimizes a cost function defined in terms of energy (integral of motor power for a machine cycle) to move the load from the initial position \( x(t_0) = [0\ 0\ 0\ 0\ 0]^T \) to the final position \( x(T) = [\theta_m(T)\ \theta_l(T)\ 0\ 0]^T \), where the states are: \( x(t) = [\theta_m(t)\ \dot{\theta}_m(t)\ \theta_l(t)\ \dot{\theta}_l(t)\ i_a(t)]^T \). The notation “\( ^T \)” is for transpose. The derivation of the state-space equation is given next.

As shown in Section 3.1.5, the equations of motion of a two-inertia system are defined as follows:

\[
\dot{\theta}_m = \frac{1}{J_m} \left[ T_m - k(\theta_m - \theta_l) - b(\dot{\theta}_m - \dot{\theta}_l) \right] \tag{73}
\]

\[
\dot{\theta}_l = \frac{1}{J_l} \left[ k(\theta_m - \theta_l) + b(\dot{\theta}_m - \dot{\theta}_l) \right] \tag{74}
\]
In this case-study, the two-inertia system is powered by a permanent magnet synchronous motor (PMSM) that was modeled as given in (16). This model can be used to write the derivative of the motor current as follows:

\[
\dot{i}_a = \frac{v_a}{L_a} - \frac{R_a}{L_a} i_a - \frac{K_e}{L_a} \omega_m
\]  

(75)

From (73), (74) and (75) and assuming that \(T_m = K_t i_a\), where \(K_t\) is the torque constant of the motor, the state-space matrix of the two-inertia system can be derived with the states \(x(t) = [\theta_m(t) \omega_m(t) \dot{\theta}_l(t) \dot{\omega}_m(t) i_a(t)]^T\) as follows:

\[
\begin{bmatrix}
\dot{\theta}_m \\
\dot{\omega}_m \\
\dot{\theta}_l \\
\dot{\omega}_m \\
\dot{i}_a
\end{bmatrix} =
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
-k/J_m & -b/J_m & k/J_m & b/J_m & K_t/J_m \\
0 & 0 & 1 & 0 & 0 \\
k_l/J_l & b/J_l & -k_l/J_l & -b/J_l & 0 \\
0 & -K_e/L_a & 0 & 0 & -R_a/L_a
\end{bmatrix}
\begin{bmatrix}
\theta_m \\
\omega_m \\
\dot{\theta}_l \\
\dot{\omega}_m \\
i_a
\end{bmatrix} +
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
1/L_a
\end{bmatrix} v_a
\]  

(76)

This can be written in short-hand as:

\[
\dot{x} = Ax + Bu
\]  

(77)

5.1.2 Cost Function

The cost function is the function defined in terms of the system variable to be minimized during the optimization process. Thus, the cost function can be defined to minimize for example time, control effort, or the error between the actual trajectory and the desired trajectory. In this energy optimization method, the variable of interest to be
minimized is energy. Hence, the cost function is defined to optimize the motor electrical energy. The cost function for the energy optimization method is derived as follows.

From the steady condition \((\frac{di_\alpha}{dt} = 0)\) of (74), the motor current can be written as follows:

\[
i_\alpha = \frac{v_a - K_e \omega_m}{R_a}
\]

(78)

From (78), the motor power can be derived as follows:

\[
P = v_a i_\alpha
\]

(79)

\[
P = v_a \left(\frac{v_a - K_e \omega_m}{R_a}\right)
\]

(80)

It should be pointed out that the proposed energy optimization methods presented in this chapter are designed to optimize electrical energy which is a function of electric power \((P = v_a i_\alpha)\) and not mechanical power \((P = T_m \omega_m)\). Electrical power is a measure of the rate of input electrical energy, while mechanical power is a measure of the rate at which work is done.

The motor energy, which is used as the cost function \(J\) for the system to be optimized, is defined as follows:

\[
J = \int_0^T P dt
\]

(81)

\[
J = \int_0^T \left(\frac{v_a^2}{R_a} - \frac{v_a K_e \omega_m}{R_a}\right) dt
\]

(82)

\(T\) is the final time of a machine cycle.
5.1.3 Optimization Method

The proposed optimization method computes the motion profile of the motor angular position to minimize the motor energy in a point-to-point move. In industrial systems, the motor angular position $\theta_m(t)$ calculated from the energy optimization method is the reference signal applied to the servo drive controlling the motor. As the motor performs the point-to-point move following this optimized reference signal, energy is minimized. In this optimization method, the control effort $u(t)$ and the states $x$ of the state-space equation that model the mechanical system are optimized. When applying this energy optimization method with industrial machines, the optimized control input $u(t)$ can be ignored, since the reference used to control the system and optimize the energy is the optimized motion profile of the angular motor position, one of the states $x$. The details of this energy optimization method to compute the optimized states $x$ are given below.

With the cost function defined in (82) and the system equation defined in (76) and (77), the Hamiltonian equation, which is used to solve optimization problems according to the Pontryagin Maximum Principle [113], can be defined as follows [93]:

$$ H = L + \lambda^T f $$

(83)

$L$ is the cost function, also called performance index, $\lambda$ is the Lagrange multiplier, and $f$ is the system equation ($Ax + Bu$) [93]. The Lagrange multiplier $\lambda$ values are not necessarily of interest, but $\lambda$ is used in intermediate steps to find the variables of interest ($x$ and $u$).

The Hamiltonian equations for optimal control were developed by Lev Pontryagin while introducing the Maximum Principle in 1962 [113]. These equations were based on the Hamiltonian equations for classical mechanics that were developed and published by
William Rowan Hamilton in 1835 [114]. Meanwhile, the Hamiltonian mechanics equations were based on Lagrange equations developed earlier by Joseph Louis Lagrange in 1788, who also invented the calculus of variation in the 1750s [115].

The Pontryagin’s Maximum Principle can be used in maximization or minimization problems [113]. Examples of maximization problems include the maximization of speed of a rocket, and maximization of earnings in financial investments. Minimization problems include the minimization of time to warm up a system, and minimization of the error between the desired trajectory and the actual trajectory. The Pontryagin’s Maximum (or Minimum) Principle is used to compute a control effort that takes a system from the initial condition to the final condition with minimum cost. The optimal control effort is found by minimizing the Hamiltonian equation in (83) at each instant \( t \) as defined by the Minimum Principle [116]. The Hamiltonian equation is minimized by satisfying the necessary conditions defined by the Pontryagin’s Minimum Principle [116, 117]. These necessary conditions are defined below in (85), (86) and (87).

Substituting (77) and (82) into (83), the Hamiltonian equation \( H \) can be written as follows:

\[
H = \frac{v_{a}^{2}}{R} - \frac{v_{a}K_{e} \omega_{m}}{R} + \lambda^{T} (Ax + Bu) \quad (84)
\]

The optimal solution is obtained by calculating a control input \( u \) to drive the system from the initial to the final position by minimizing the cost function [93]. The cost function is minimized by satisfying necessary conditions. Through the calculus of variations, the necessary conditions can be derived as the gradients of \( H \) with respect to \( x \).
(symbolized as $H_s$), $\lambda$ (symbolized as $H\lambda$), and $u$ (symbolized as $H_u$), are defined as in (85), (86), and (87), respectively [93, 118]. The gradient of $H$ with respect to $\lambda$, $H\lambda$, yields the system equation $Ax + Bu$. The gradient of $H$ with respect to the states $x$ yields the costate equation $H_x$. The gradient of $H$ with respect to the states the control effort $u(t)$ yields the stationary condition $H_u = 0$. The $H_x$, $H\lambda$, and $H_u$ constitute the necessary conditions for a constrained minimum [93]:

\begin{align*}
\text{Costate Equation:} \quad & \dot{\lambda} = -H_x = \frac{v_a K_e}{R} - A^T \lambda \quad \text{(85)} \\
\text{System Equation:} \quad & \dot{x} = H_x = Ax + Bu \quad \text{(86)} \\
\text{Stationary Condition:} \quad & 0 = H_u = \frac{2v_a}{R} - \frac{K_e \omega_m}{R} + B^T \lambda \\
& 0 = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \end{bmatrix} \quad \text{(87)} \\
& = \begin{bmatrix} 2v_a \\ -\frac{K_e \omega_m}{R} \\ \frac{\lambda_5}{L_a} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 1/L_a \\ 0 & 0 & 0 & \frac{\lambda_5}{L_a} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix}
\end{align*}

From (87), the motor voltage can be derived as follows:

\begin{equation}
v_a = \frac{K_e \omega_m}{2} - \frac{R \lambda_5}{2L_a} \quad \text{(88)}
\end{equation}

Substituting (88) into (85), the costate equation can be rewritten as follows:

\begin{equation}
\dot{\lambda} = \frac{K_e}{R} \left( \frac{K_e \omega_m}{2} - \frac{R \lambda_5}{2L_a} \right) - A^T \lambda \quad \text{(89)}
\end{equation}

This can be written in matrix form as follows:
From (88), the output of the Hamiltonian system can be defined as follows:

\[
\begin{bmatrix}
\dot{\lambda}_1 \\
\dot{\lambda}_2 \\
\dot{\lambda}_3 \\
\dot{\lambda}_4 \\
\dot{\lambda}_5 \\
\end{bmatrix} = \begin{bmatrix}
\frac{K_e}{R} - \frac{R \lambda_5}{2L_a} \\
\frac{k}{J_m} & 0 & 0 & 0 & 0 \\
0 & -k/J_m & 0 & 0 & 0 \\
0 & b/J_m & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} \begin{bmatrix}
\lambda_1 \\
\lambda_2 \\
\lambda_3 \\
\lambda_4 \\
\lambda_5 \\
\end{bmatrix}
\]  \quad (90)

Substituting (88) into (75), the derivative of the motor current can be defined as a function of only the states, as shown below.

\[
i_a' = \frac{1}{L_a} \left( \frac{K_e \omega_m}{2} - \frac{R \lambda_5}{2L_a} \right) - \frac{R_a}{L_a} i_a - \frac{K_e}{L_a} \omega_m
\]

(91)

That can be simplified to be:

\[
i_a' = -\frac{R}{2L_a^2} \lambda_5 - \frac{R_a}{L_a} i_a - \frac{K_e}{L_a} \omega_m
\]

(92)

From (76), (90) and (92), the Hamiltonian system can be defined as follows:

\[
\begin{bmatrix}
\dot{\theta}_m \\
\dot{\omega}_m \\
\dot{\theta}_t \\
\dot{\omega}_t \\
\dot{i}_a \\
\dot{\lambda}_1 \\
\dot{\lambda}_2 \\
\dot{\lambda}_3 \\
\dot{\lambda}_4 \\
\dot{\lambda}_5 \\
\end{bmatrix} = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-k/J_m & -b/J_m & k/J_m & b/J_m & k_t/J_m & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
k/J_t & b/J_t & -k/J_t & -b/J_t & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -K_e/L_a & 0 & 0 & -R_a/L_a & 0 & 0 & 0 & 0 & -R_a/(2L_a^2) \\
0 & 0 & 0 & 0 & 0 & k/J_m & 0 & -k/J_t & 0 & 0 \\
0 & K_e^2/(2L_a) & 0 & 0 & 0 & -1 & b/J_m & 0 & -b/J_t & K_e/(2L_a) \\
0 & 0 & 0 & 0 & 0 & -k/J_m & 0 & k/J_t & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} \begin{bmatrix}
\theta_m \\
\omega_m \\
\theta_t \\
\omega_t \\
i_a \\
\lambda_1 \\
\lambda_2 \\
\lambda_3 \\
\lambda_4 \\
\lambda_5 \\
\end{bmatrix}
\]

(93)

This can be written in short-hand as:

\[
\dot{x} = Hx
\]

(94)

From (88), the output of the Hamiltonian system can be defined as follows:
This can be written in short-hand as:

$$u = Cx$$  \hspace{1cm} (96)$$

The Hamiltonian system can then be solved in Matlab with the “lsim” command defined as \( u^*,X\) = lsim(A,B,C,D,u,t,[x_0;\lambda_0]), where A is H defined in (93) and (94), B is a \( m \times 1 \) vector where \( m \) is the number of states, C is defined in (95) and (96), D = 0, \( u \) is a \( n \times p \) matrix where \( n \) is the number of system inputs, which is \( v_a \) for the two-inertia system, while \( p \) is the number of time samples of \( t \), which yields \( u \) defined as \( u = 0^*t \), \( x_0 \) is the initial condition defined as \( [\theta_{m0} \omega_{m0} \theta_{l0} \omega_{l0} i_{a0}]^T = [0 0 0 0]^T \), \( \lambda_0 \) is the initial condition for \( \lambda_1, \lambda_2, \lambda_3, \lambda_4, \) and \( \lambda_5 \), \( X \) is the optimized solution for \( [\theta_{m} \omega_{m} \theta_{l} \omega_{l} i_{a} \lambda_{1} \lambda_{i} \lambda_{l} \lambda_{i}]^T \), and \( u^* \) is the optimized control effort.

The initial condition \( \lambda_0 \) can be derived from a transition matrix \( \Phi(t,t_0) \) which is used to map the states at any given time \( t \) as \( x(t) = \Phi(t,t_0)x(t_0) \) where \( \Phi(t,t_0) = \mathcal{L}^{-1}([I s - H]^{-1}) \) in the s-domain or \( \Phi(t,t_0) = e^{H(t-t_0)} \) in the time-domain [116]. Thus, the Hamiltonian system in (93) can be associated with the transition matrix as follows:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{\lambda}(t) \end{bmatrix} = \Phi(t-t_0) \begin{bmatrix} x(t_0) \\ \lambda(t_0) \end{bmatrix}$$  \hspace{1cm} (97)$$

The solution of the transition matrix can be symbolically represented as:
\[ \phi(t - t_0) = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix} \]  

(98)

Where \( \phi_{11}, \phi_{12}, \phi_{21}, \text{ and } \phi_{11} \) are \( m/2 \times m/2 \) matrices from each quadrant of the transition matrix solution, where \( m \) is the number of states plus the number of co-states. For the two-inertia system, \( \phi_{11}, \phi_{12}, \phi_{21}, \text{ and } \phi_{11} \) are \( 5 \times 5 \) matrices. Thus, from the transition matrix defined as in (98), the Hamiltonian system associated to the transition matrix in (97) can be represented as follows for the final time \( T \):

\[
\begin{bmatrix}
\theta_m(T) \\
\omega_m(T) \\
\theta_1(T) \\
\omega_m(T) \\
\lambda_1(T) \\
\lambda_2(T) \\
\lambda_3(T) \\
\lambda_4(T) \\
\lambda_5(T)
\end{bmatrix} = \begin{bmatrix}
\phi_{11}(T - t_0) & \phi_{12}(T - t_0) \\
\phi_{21}(T - t_0) & \phi_{22}(T - t_0)
\end{bmatrix}
\begin{bmatrix}
\theta_m(t_0) \\
\omega_m(t_0) \\
\theta_1(t_0) \\
\omega_m(t_0) \\
\lambda_1(t_0) \\
\lambda_2(t_0) \\
\lambda_3(t_0) \\
\lambda_4(t_0) \\
\lambda_5(t_0)
\end{bmatrix} + \begin{bmatrix}
\lambda_1(t_0) \\
\lambda_2(t_0) \\
\lambda_3(t_0) \\
\lambda_4(t_0) \\
\lambda_5(t_0)
\end{bmatrix} 
\]  

(99)

The upper portion of (99) can be rewritten as follows:

\[
\begin{bmatrix}
\theta_m(T) \\
\omega_m(T) \\
\theta_1(T) \\
\omega_m(T) \\
\lambda_1(T)
\end{bmatrix} = \phi_{11}(T - t_0)
\begin{bmatrix}
\theta_m(t_0) \\
\omega_m(t_0) \\
\theta_1(t_0) \\
\omega_m(t_0) \\
\lambda_1(t_0)
\end{bmatrix} + \phi_{12}(T - t_0)
\begin{bmatrix}
\lambda_1(t_0) \\
\lambda_2(t_0) \\
\lambda_3(t_0) \\
\lambda_4(t_0) \\
\lambda_5(t_0)
\end{bmatrix} 
\]  

(100)

From (100), the initial condition \( \lambda_0 = [\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5]^T \) can be obtained as shown below, which is the last parameter to be derived to solve the system with “Isim” described above:

\[
\begin{bmatrix}
\lambda_1(t_0) \\
\lambda_2(t_0) \\
\lambda_3(t_0) \\
\lambda_4(t_0) \\
\lambda_5(t_0)
\end{bmatrix} = \phi_{12}^{-1}(T - t_0)
\begin{bmatrix}
\theta_m(T) \\
\omega_m(T) \\
\theta_1(T) \\
\omega_m(T) \\
\lambda_1(t_0)
\end{bmatrix} - \phi_{11}^{-1}(T - t_0)
\begin{bmatrix}
\theta_m(t_0) \\
\omega_m(t_0) \\
\theta_1(t_0) \\
\omega_m(t_0) \\
\lambda_1(t_0)
\end{bmatrix} 
\]  

(101)
5.1.4 Simulation Results

The energy optimization method described in the previous section was applied to the two-inertia system to calculate the motion profile of the position reference signal that optimizes the energy for an index (point-to-point) move. This optimized motion profile was then validated with a two-inertia system controlled by a cascade PI control loop and implemented in Simulink. The cascade PI control loop consists of a position loop and a velocity loop. The position reference for the position loop is the optimized motion profile $\theta_m$ computed from the energy optimization method described in the previous section. The motor energy consumption to follow the optimized motion profile was then calculated. In order to verify that the optimized motion profile requires minimum energy, the same two-inertia system with the same control loop gains, was commanded by other motion profiles to perform the same move in the same amount of time as the optimized motion profile. Eight motion profiles typically used in industrial applications and described in Chapter 2 were used as the position reference to the two-inertia system to perform the same move as the optimized motion profile. The energy for each one of these eight motion profiles was also calculated and then compared to the energy consumption for the optimized motion profile. This comparison in energy consumption was used to validate the energy optimization method for single-axis systems. The details to calculate the optimized motion profile, simulate the two-inertia system in closed-loop, implement the eight motion profiles, and calculate motor energy during motion is given below. The simulation results comparing the motor energy while commanded by the various motion profiles are also provided below.
The optimization method described in the previous section for the two-inertia system shown in Fig. 4-1 was implemented in Matlab and simulated for the permanent magnet motor in Table 3-2, while connected to a flywheel through a rod and coupling with the parameters shown in Table 4-2.

The trajectory to be optimized in simulation was chosen to be a 0.1 motor revolution (0.628 rad) in 100 milliseconds, since short and fast moves are typical in motion control applications. Thus, the final condition was defined as $[\theta_m(T) \omega_m(T) \theta_l(T) \omega_l(T) i_a(T)]^T = [0.628 0 0.628 0 0]^T$. The initial condition was defined as $[\theta_m(0) \omega_m(0) \theta_l(0) \omega_l(0) i_a(0)]^T = [0 0 0 0 0]^T$. The transition matrix was then calculated as $\Phi(t - t_0) = e^{A(t-t_0)}$ using the command `expm` (matrix exponential) in Matlab. This allows one to compute $\lambda(t_0)$ defined in (101). The Hamiltonian matrix can be calculated from (93) with the parameters in Table 3-2 and Table 4-2. Thus, the states $x(t) = [\theta_m(t) \omega_m(t) \theta_l(t) \omega_l(t) i_a(t)]$ and the input $u(t)$ can be calculated from $[u^*, X] = lsim(H, B, C, D, u, t, [x_0; \lambda_0])$.

The optimized angular motor position ($\theta_m$) and the optimized angular load position ($\theta_l$) are shown in Fig. 5-2, while the optimized angular motor velocity ($\omega_m$) and the optimized angular load velocity ($\omega_l$) are shown in Fig. 5-3. Although, this position and velocity profile are similar to typical motion profiles as those shown in Section 2.6, these are unique profiles. It can also be observed from these profiles that there are no discontinuities, which contributes to smooth motion of the motor and load.
Fig. 5-2 - Optimized angular motor position $\theta_m$ and optimized angular load position $\theta_l$.

Fig. 5-3 - Optimized angular motor velocity $\omega_m$ and angular load velocity $\omega_l$.

The optimized motor voltage $v_a$ is shown in Fig. 5-4. Since the motor used in this case-study is a 460V permanent magnet motor, the calculated motor voltage $v_a$ must be within this limit of ±460V. As shown in Fig. 5-4, the optimized motor voltage $v_a$ is within the motor voltage limit.
The smoothness of the optimized profile can be evaluated from the acceleration profile. A smooth profile for an index move is characterized by zero acceleration at the beginning and the end of the motion profile without any discontinuities or sharp transitions along the move. These characteristics can be observed in the acceleration profile of the optimized load motion profile shown in Fig. 5-5.

In order to validate the efficiency of the optimized motion profile in reducing the energy consumption to perform an index move, the energy consumption of the optimized profile to move 0.1 revolution in 100 milliseconds was compared to the energy
consumption of eight motion profiles to perform the same task. The criterion to select these eight motion profiles for this comparison was to select motion profiles typically used in industrial applications. The selected motion profiles are: 5\textsuperscript{th}-order polynomial; 7\textsuperscript{th}-order polynomial; 9\textsuperscript{th}-order polynomial; sine; trapezoidal; cycloidal; modsine; and cubic profiles. These profiles are defined in Section 2.6.

The two-inertia system, motor, control system, motion profiles, and an energy estimator were modeled in Simulink, as shown in Fig. 5-6, to compute the energy required by the motor to perform an index move of 0.1 revolution in 100 milliseconds while the control system was commanded by the optimized motion profile and then by each one of the eight motion profiles listed above. The subsystem called ‘Two-Inertia System + Motor” in Fig. 5-6 consists of the motor and the two-inertia system shown in Fig. 3-11. The same cascade PI controller described in Section 3.1.12 and tuned as described in Section 3.1.11 was used as shown in Fig. 5-6 to validate the energy optimization method described in the previous section.

The reference signal for this closed-loop control was either the optimized motor angular position $\theta_m$ shown in Fig. 5-2, or one of the eight motion profiles listed above. The optimized motion profile was imported into the Simulink model from the Matlab Workspace with the “Optimized Motion Profile Pos Cmd” block. These eight motion profiles were implemented in the subsystem called “Motion Profiles” in Fig. 5-6.

The required energy of the motor to perform the index move of 0.1 revolution in 100 milliseconds was calculated by the subsystem “Energy Calculator” in Fig. 5-6. This subsystem consists of the implementation of (79) and (81).
A comparison of the final value of energy consumption for the eight motion profiles and the optimized motion profile is shown in Table 5-1. Meanwhile, a comparison of the energy curve for the various motion profiles tested in this dissertation is shown in Fig. 5-7. As shown in this table, the lowest energy consumption for the 0.1 revolution in 100 milliseconds move was achieved by the optimized motion profile. All other types of motion profile required higher energy than the optimized profile to make the same. The modsine motion profile required 13.5% more energy than the optimized motion profile. The 5\textsuperscript{th}-order polynomial profile, which is a motion profile extensively used in industrial applications, required 26.2% more energy than the optimized motion profile to perform the same move. As expected, the 9\textsuperscript{th}-order motion profile, which has the highest acceleration rates among the motion profiles listed in Table 5-1, required the highest amount of energy to perform this move. As shown in this table, the maximum position-following error for each motion profile was similar, which validates that fair tuning for the cascade PI controller was used in this test.
Table 5-1 - Comparison of energy consumption from the simulation results for various types of motion profiles applied to the two-inertia system to move 0.1revs in 100ms with low position-following error.

<table>
<thead>
<tr>
<th>Motion Profile</th>
<th>Final Energy (J)</th>
<th>Variation (%)</th>
<th>Max. Position-Following Error (rad)</th>
<th>Max. Position-Following Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized</td>
<td>0.799</td>
<td>0</td>
<td>0.0004</td>
<td>0.065</td>
</tr>
<tr>
<td>Trapezoidal</td>
<td>1.858</td>
<td>132.5</td>
<td>0.0006</td>
<td>0.090</td>
</tr>
<tr>
<td>Cycloidal</td>
<td>1.149</td>
<td>43.8</td>
<td>0.0005</td>
<td>0.080</td>
</tr>
<tr>
<td>ModSine</td>
<td>0.907</td>
<td>13.5</td>
<td>0.0004</td>
<td>0.069</td>
</tr>
<tr>
<td>Cubic</td>
<td>1.313</td>
<td>64.3</td>
<td>0.0006</td>
<td>0.091</td>
</tr>
<tr>
<td>Sine-Harmonic</td>
<td>1.120</td>
<td>40.1</td>
<td>0.0005</td>
<td>0.085</td>
</tr>
<tr>
<td>5th Order Polynomial</td>
<td>1.009</td>
<td>26.2</td>
<td>0.0005</td>
<td>0.076</td>
</tr>
<tr>
<td>7th Order Polynomial</td>
<td>1.479</td>
<td>85.1</td>
<td>0.0006</td>
<td>0.091</td>
</tr>
<tr>
<td>9th Order Polynomial</td>
<td>2.059</td>
<td>157.7</td>
<td>0.0007</td>
<td>0.111</td>
</tr>
</tbody>
</table>

Fig. 5-7 - Comparison of energy consumption for various types of motion profiles applied to the two-inertia system to move 0.1revs in 100ms with low position-following error
However, the control system needs to be well tuned to obtain the benefits of the optimized motion profile. If the control loops are not well tuned, the position-following error can be high, and consequently the motion profile at the load will deviate from the commanded optimized profile and the required energy will be higher than the energy that would be necessary if the load was moving according to the optimized motion profile.

In order to validate this concept, the control loops of the two-inertia system were re-tuned to obtain position-following errors 10 times higher than the position-following error in Table 5-1 and the results are shown in Table 5-2 and in Fig. 5-8. A comparison of the energy curves for all motion profiles shown in Table 5-2 is shown in Fig. 5-8. As shown in this table, the energy required for the optimized motion profile was higher than in Table 5-1. Additionally, the energy savings in comparison to the other motion profiles reduced to the point that the sine-harmonic profile was found to use less energy than the optimized profile. However, the sine-harmonic, cubic, and trapezoidal profiles are undesirable profiles for industrial applications due to the high jerk content with these profiles at the beginning and end of the move, since the acceleration is not zero at these points of the move. These motion profiles are highlighted in Table 5-2. Meanwhile, the optimized motion profile provides smooth motion as described for Fig. 5-5. Thus, when the energy required by the motor while commanded by the optimized motion profile is compared to the required motor energy while commanded by the other motion profiles in Table 5-2 that are recommended for industrial applications, the optimized motion profile saved at least 13.4% of energy, even when the control loop was not well tuned, as shown in this table. Therefore, the optimized motion profile not only minimizes the energy usage, but also provides smooth motion.
Table 5-2 - Comparison of energy consumption from the simulation results for various types of motion profiles applied to the two-inertia system to move 0.1revs in 100ms with higher position-following error

<table>
<thead>
<tr>
<th>Motion Profile</th>
<th>Final Energy (J)</th>
<th>Variation (%)</th>
<th>Max. Position-Following Error (rad)</th>
<th>Max. Position-Following Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized</td>
<td>0.820</td>
<td>0.000</td>
<td>0.004</td>
<td>0.623</td>
</tr>
<tr>
<td>Trapezoidal</td>
<td>0.975</td>
<td>18.962</td>
<td>0.005</td>
<td>0.722</td>
</tr>
<tr>
<td>Cycloidal</td>
<td>1.176</td>
<td>43.438</td>
<td>0.005</td>
<td>0.828</td>
</tr>
<tr>
<td>ModSine</td>
<td>0.930</td>
<td>13.403</td>
<td>0.004</td>
<td>0.697</td>
</tr>
<tr>
<td>Cubic</td>
<td>0.832</td>
<td>1.457</td>
<td>0.004</td>
<td>0.585</td>
</tr>
<tr>
<td>Sine-Harmonic</td>
<td>0.802</td>
<td>-2.113</td>
<td>0.004</td>
<td>0.612</td>
</tr>
<tr>
<td>5th Order Polynomial</td>
<td>1.024</td>
<td>24.987</td>
<td>0.005</td>
<td>0.761</td>
</tr>
<tr>
<td>7th Order Polynomial</td>
<td>1.524</td>
<td>85.966</td>
<td>0.006</td>
<td>0.923</td>
</tr>
<tr>
<td>9th Order Polynomial</td>
<td>2.131</td>
<td>160.030</td>
<td>0.007</td>
<td>1.077</td>
</tr>
</tbody>
</table>

Fig. 5-8 - Comparison of energy consumption for various types of motion profiles applied to the two-inertia system to move 0.1revs in 100ms with higher position-following error
The Matlab code used to generate simulation results shown in Fig. 5-2, Fig. 5-3, Fig. 5-4, Fig. 5-5, Fig. 5-7, Fig. 5-8, Table 5-1, and Table 5-2 is given in Appendix D.

It was observed that the torsional stiffness of the shaft, rod, and coupling affected the amount of energy saved with the optimized motion profile in respect to the other motion profiles. From the results shown in Table 5-1, the optimized motion profile used the least amount of energy followed by the modsine. Thus, the percentage of energy savings between the optimized motion profile and modsine for various torsional stiffness values was investigated and the results are shown in Fig. 5-9. It is shown in this figure that as the torsional stiffness reduced, which means that the system becomes more compliant, the energy savings with the optimized motion profile in comparison to the energy used with the modsine profile increases. In these results shown in Fig. 5-9, the load inertia was kept constant at 0.0025 kgm², which represents an inertia ratio between load and motor of 57 times.

![Fig. 5-9 – Percentage of energy savings between the optimized motion profile and modsine profile for various torsional stiffness values](image-url)
The effect of inertia ratio between motor and load to the energy savings that can be achieved with this energy optimization method was also investigated. The effect of the inertia ratio between motor and load from 10:1 up to 80:1 ratio to the energy consumption is shown in Fig. 5-10. This range of inertia ratio was chosen because it is a typical range for industrial machines, although lower than 10:1 ratio is also very common, and above 80:1 is also found. As shown in Fig. 5-10, as the inertia ratio increases, the required energy to drive a higher load inertia also increases as expected. Additionally, independent of the inertia ratio, the optimized motion profile required the lowest amount of energy as show in Fig. 5-10. The optimized motion profile was recalculated for each inertia ratio shown in this figure.

If the percentage change in energy consumption with the inertia ratio is calculated, it can be observed that the recommended motion profiles (cycloidal, modsine, 5th order polynomial, 7th order polynomial, and 9th order polynomial) and the optimized motion profile keep the same percentage increase in energy consumption as the inertia ratio increases. However, the motion profiles to be avoided in industrial applications (trapezoidal, cubic, and sine-harmonic) present a higher percentage increase in energy consumption as the inertia ratio increases as shown in Fig. 5-11.
Fig. 5-10 - Effect of inertia ratio to energy consumption.

Fig. 5-11 - Percentage of change in energy consumption with the energy ratio.
5.1.5 Experimental Results

The experimental results with the two-inertia system shown in Fig. 4-1 and Fig. 4-2 while commanded by the optimized motion profile shown in Fig. 5-2 and by the other eight motion profiles shown in Table 5-1 for a 0.1 rev move in 100ms are shown in Table 5-3 and Fig. 5-12.

The experimental results shown in Table 5-3 line up with the simulation results shown in Table 5-2. The position-following error in these two tables are on the same order of magnitude. As demonstrated in simulation in Table 5-2 and validated in the experimental results shown in Table 5-3, some of the motion profiles to be avoided in industrial applications required a little less energy than the optimized motion profile. As demonstrated from the simulation results while comparing Table 5-2 and Table 5-3, a better tuned system to reduce the position-following error would be necessary to make the load follow the optimized motion profile closer to reduce the energy usage with the optimized motion profile.

When comparing the final energy of the optimized motion profile in Table 5-3 with the final energy of the recommended motion profiles for industrial applications (cycloidal, modsine, 5th order polynomial, 7th order polynomial, and 9th order polynomial), the optimized motion profile saved at least 15.5% of energy to execute the same task than the other motion profiles.
Table 5-3 - Comparison of energy consumption from the experimental results for various types of motion profiles applied to the two-inertia system to move 0.1revs in 100ms

<table>
<thead>
<tr>
<th>Motion Profile</th>
<th>Final Energy (J)</th>
<th>Variation (%)</th>
<th>Max. Position-Following Error (rad)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized</td>
<td>1.175</td>
<td>0</td>
<td>0.0026</td>
</tr>
<tr>
<td>Trapezoidal</td>
<td>1.229</td>
<td>4.6</td>
<td>0.0026</td>
</tr>
<tr>
<td>Cycloidal</td>
<td>1.701</td>
<td>44.8</td>
<td>0.0031</td>
</tr>
<tr>
<td>ModSine</td>
<td>1.357</td>
<td>15.5</td>
<td>0.0027</td>
</tr>
<tr>
<td>Cubic</td>
<td>1.154</td>
<td>-1.8</td>
<td>0.0026</td>
</tr>
<tr>
<td>Sine-Harmonic</td>
<td>1.154</td>
<td>-1.8</td>
<td>0.0024</td>
</tr>
<tr>
<td>5th Order Polynomial</td>
<td>1.572</td>
<td>33.8</td>
<td>0.0029</td>
</tr>
<tr>
<td>7th Order Polynomial</td>
<td>2.321</td>
<td>97.5</td>
<td>0.0037</td>
</tr>
<tr>
<td>9th Order Polynomial</td>
<td>3.110</td>
<td>164.7</td>
<td>0.0045</td>
</tr>
</tbody>
</table>

Fig. 5-12 - Simulation results with the same position-following error between 0.010 to 0.020 rad
5.1.6 Discussion of Results

In Section 5.1, an energy optimization method was investigated to minimize the electrical energy consumption of a two-inertia system and this method was validated by comparing the energy cost of the system while commanded by the optimized profile to the energy cost while the system was commanded by eight motion profiles typically used in industrial applications. Both simulations and experiments presented significant energy reduction in comparison to recommended motion profiles for industrial applications (cycloidal, modsine, 5th order polynomial, 7th order polynomial, and 9th order polynomial). The simulation results presented energy savings between 13.4% and 160%, while the experimental results presented energy savings between 15.5% and 164% in comparison to these recommended types of motion profiles.

The energy optimization motion profile yields lower energy consumption than other types of motion profile to perform the same task when the control systems are tuned well to produce low position-following error. Approaches such as velocity feedforward, observers, and high tuning gains can help to reduce the position- and velocity-following errors and consequently reduce the electrical energy consumption.

As the inertia ratio between motor and load increases, the rate at which the energy consumption of the undesirable motion profiles (such as trapezoidal, cubic, and sinharmonic) increases is higher than the rate at which the energy consumption with recommended motion profiles (such as cycloidal, modsine, 5th order polynomial, 7th order polynomial, and 9th order polynomial) increases.

As the torsional stiffness of the system reduces and the system becomes more compliant, the ability of the energy optimization method to compute optimized motion
profiles that save more energy increases, i.e., the optimized motion profiles have a higher impact in saving energy for compliant systems than for rigid systems. This is potentially attributed to the ability of the energy optimization method in computing an optimized motion profile that better utilize the energy stored in the compliant components.

The calculation of the transition matrix in (98) that is \( \Phi(t - t_0) = e^{H(t-t_0)} \) may yield numbers too large when the motor and/or load inertia are small and when the move time is large, and consequently, the solution of this energy optimization method could fail to find a solution. This occurs because some elements of the Hamiltonian matrix \( H \) are divided either by motor inertia \( J_m \) or load inertia \( J_l \), as shown in (93), and as these inertias reduce, \( H \) increases, and the solution of the transition matrix also increases. As the move increases, the solution of the transition matrix also increases. Since typical values of motor inertia for motors used in motion control application can vary from 0.00001 kgm\(^2\) to 0.05 kgm\(^2\) [119] with load inertia being typically between 1 to 100 times the motor inertia, the solution of the transition matrix would require lower values of move time in the order of milliseconds to yield a solution. For motion control applications, where the inertias in the system are in general low, the move times are also short. Thus, the shorter move times may compensate for the low inertias and a solution can still be obtained with this energy optimization method. For heavier loads, where inertias are significantly higher than those found in typical motion control applications, the move times are higher than those in motion control applications. Thus, the higher motor inertia and load inertia compensates for the increased move time, which may allow this energy optimization method to provide a solution.
5.2 H-Bot

H-Bots can be used in tracking applications where the end-effector needs to track a pre-defined trajectory in the Cartesian space. H-bots can also be used in point-to-point applications where the position-following error of the end-effector during motion is not, in general, of a concern as long as the target position is reached accurately. Typical applications for H-Bots include pick-and-place, sorting, printing, and dispensing.

An iterative energy optimization method was developed to compute an optimized trajectory in the Cartesian space for multi-axis coordinated systems. This method was validated with an H-Bot. The optimized trajectory is designed to reside within two boundaries. One boundary is defined about the desired trajectory in the Cartesian space. The other boundary is defined as the maximum position error at the target position. Thus, this energy optimization method computes a trajectory in the Cartesian space within these boundaries with the lowest energy cost. The inputs for this energy optimization method are the desired trajectory in the Cartesian space, the boundaries in the Cartesian space, and the mechanical data to compute the kinetic model of the system. Based on these inputs, the trajectory in the Cartesian space with the lowest energy cost is obtained by minimizing a cost function. The cost function is defined by weighting matrices. The weighting matrices that yield a trajectory in the Cartesian space within these boundaries with the lowest energy cost are unknown. An iterative method was developed to compute the weighting matrices that yield a trajectory inside the boundaries with the lowest energy cost. This proposed iterative method allows to choose arbitrary initial values for the weighting matrices because it was designed to converge to the optimal solution independently of the initial values assigned to the weighting matrices. Hence, the
weighting matrix are recalculated at each iteration to search for a trajectory inside the boundaries in the Cartesian space with the lowest energy cost. The iterations can stop once the energy converges to the lowest level. As reported by many researchers, the identification of the values of the weighting matrices that provide the desired system behavior requires exhaustive simulations and/or experimental trials [93, 108-110]. The developed iterative method addresses this issue.

The proposed energy optimization method was validated for an H-Bot with the configuration shown in Fig. 5-13 [120]. The derivation of the state space equation for the H-Bot, cost function, Hamiltonian equation, necessary conditions, boundary conditions, and the derivation of the equations to compute the motion profile that optimized electric energy are given next.

The H-Bot configuration shown in Fig. 5-13 contains eight pulleys, a belt, three linear guides, and two motors. This H-Bot can work in any orientation, but it was set to the horizontal position for the derivations of the equations of motion shown below.

Fig. 5-13 - Diagram of an H-Bot.
5.2.1 State Space Equation

The equations of motion for the H-Bot shown in Fig. 5-13 are as follows.

\[
\ddot{\theta}_1 = \frac{1}{J_{m1} + \frac{r^2}{4}(m_x + m_y)} \left[ \tau_{m1} - \dot{\theta}_2 \left( \frac{r^2}{4} y_m - m_y \right) - \dot{\theta}_2 \left( B_{m2} \frac{r^2}{2} (B_i + B_j) \right) \right]
\]

\[
\ddot{\theta}_2 = \frac{1}{J_{m2} + \frac{r^2}{4}(m_x + m_y)} \left[ \tau_{m2} - \dot{\theta}_1 \left( \frac{r^2}{4} y_m - m_y \right) - \dot{\theta}_1 \left( B_{m1} \frac{r^2}{2} (B_i - B_j) \right) \right]
\]

(102)

Where, \( \ddot{\theta}_1 \) and \( \ddot{\theta}_2 \) are the angular acceleration of motors M1 and M2, respectively (see Fig. 5-13), \( r \) is the radius of the pulleys 1, 2, 3, and 4, \( \tau_{m1} \) and \( \tau_{m1} \) are the torques developed by motors M1 and M2, respectively, \( J_{m1} \) and \( J_{m2} \) are the moment of inertia at driving pulley 1 and 2, respectively, \( m_y \) is the mass in y-direction, and \( m_x \) is the mass in x-direction. The mass in y-direction \( m_y \) is a function of the mass of the cart \( m_{cart} \), load mass \( m_{load} \), radius \( r_p \) of the idler pulleys 5, 6, 7, and 8, and inertia of the idler pulleys 5, 6, 7, and 8 as shown in Fig. 5-13 which are respectively \( J_5 \), \( J_6 \), \( J_7 \), and \( J_8 \). Thus, \( m_y \) is defined as follows:

\[
m_y = m_{cart} + m_{load} + \frac{4J_p}{r_p^2}
\]

(103)

With:

\[
J_p = J_5 = J_6 = J_7 = J_8
\]

(104)
Meanwhile, the mass in \( x \)-direction, \( m_x \), is a function of \( m_y \) and the mass of the bridge, \( m_{\text{bridge}} \), which consists of the mass of the slide on the bridge and the mass of the four attached idler pulleys (5, 6, 7, and 8). Thus:

\[
m_x = m_{\text{bridge}} + m_y
\]  

(105)

The moment of inertia at the driving pulley 1, \( J_{m1} \), is a function of the inertia of the idler pulley 3, \( J_3 \), the driving pulley 1, \( J_1 \), and the motor inertia, \( J_m \), reflected through the gearbox ratio \( GR \) to the pulley side. Thus:

\[
J_{m1} = J_1 + J_3 + J_m GR^2
\]  

(106)

Similarly, the moment of inertia at the driving pulley 2, \( J_{m2} \), is a function of the inertia of the idler pulley 4, \( J_4 \), the driving pulley 2, \( J_2 \), and the motor inertia, \( J_m \), reflected through the gearbox ratio \( GR \) to the pulley side. Thus:

\[
J_{m2} = J_2 + J_4 + J_m GR^2
\]  

(107)

The viscous friction coefficient at the driving pulley 1, \( B_{m1} \), is a function of the motor viscous friction coefficient, \( B_m \), and the viscous friction coefficient at pulley 3, \( B_3 \), as follows:

\[
B_{m1} = B_3 + B_m
\]  

(108)

Similarly, the viscous friction coefficient at the driving pulley 2, \( B_{m2} \), is a function of the motor viscous friction coefficient, \( B_m \), and the viscous friction coefficient at pulley 4, \( B_4 \), as follows:
\[ B_{m2} = B_4 + B_m \]  

(109)

The viscous friction coefficient in y-direction, \( B_y \), is a function of the viscous friction coefficient between the cart and the bridge, \( B_{cart} \), and the viscous friction coefficient of the pulleys 5, 6, 7, and 8, which are \( B_5, B_6, B_7, \) and \( B_8 \), respectively. Thus, \( B_y \) is as follows:

\[ B_y = B_{cart} + \frac{4B_p}{r_p^2} \]  

(110)

With:

\[ B_p = B_5 = B_6 = B_7 = B_8 \]  

(111)

Meanwhile, the viscous friction coefficient in x-direction, \( B_x \), is the viscous friction coefficient between the bridge and both fixed linear guides, \( B_{bridge} \):

\[ B_x = B_{bridge} \]  

(112)

The equations of motion defined in (102) need to be linearized to use with the energy optimization method. In order to simplify the mathematical manipulations in the linearization of the equations of motion of the H-Bot, the following coefficients can be defined:

\[ C_1 = \frac{1}{J_{m1} + \frac{r^2}{4}(m_x + m_y)} \]  

(113)

\[ C_2 = \frac{r^2}{4}(-m_x + m_y) \]
\[ C_3 = B_{m1} - \frac{r^2}{2} (B_x + B_y) \]

\[ C_4 = \frac{1}{J_{m2} + \frac{r^2}{4} (m_x + m_y)} \]

\[ C_5 = B_{m2} + \frac{r^2}{2} (B_x - B_y) \]

Thus, the equations of motion in (102) can then be rewritten as follows:

\[ \ddot{\theta}_1 = C_1 (\tau_{m1} - \dot{\theta}_2 C_2 - \dot{\theta}_4 C_3 - \dot{\theta}_4 C_4) \]

\[ \ddot{\theta}_2 = C_5 (\tau_{m2} - \dot{\theta}_2 C_2 - \dot{\theta}_2 C_6 - \dot{\theta}_4 C_7) \]

(114)

Where, \( \tau_{m1} \) and \( \tau_{m2} \) are defined as follows:

\[ \tau_{m1} = K_i G R \eta i_1 \]

\[ \tau_{m2} = K_i G R \eta i_2 \]

(115)

Where, \( \eta \) is the efficiency of the gearbox.

The motor currents derivatives \( \dot{i}_1 \) and \( \dot{i}_2 \) can be derived from (75) as follows:

\[ \dot{i}_1 = \frac{v_{a1}}{L_a} - \frac{R_a}{L_a} i_1 - \frac{K_e}{L_a} \omega_1 G R \]

\[ \dot{i}_2 = \frac{v_{a2}}{L_a} - \frac{R_a}{L_a} i_2 - \frac{K_e}{L_a} \omega_2 G R \]

(116)
And, the state-space equation of the H-Bot can defined from (114), (115) and (116) as follows:

\[
\begin{bmatrix}
\dot{\theta}_1 \\
\dot{\omega}_1 \\
\dot{\theta}_2 \\
\dot{\omega}_2 \\
i_1 \\
i_2
\end{bmatrix} =
\begin{bmatrix}
0 & \frac{1}{-C_1C_3} & 0 & \frac{1}{C_1C_2C_4} & 0 & \frac{1}{C_1K_0GR} & 0 & \frac{-C_1C_2K_0GR}{1-C_1C_2^2C_4} \\
0 & 1 & \frac{0}{-C_1C_2^2C_4} & 0 & \frac{1}{1-C_1C_2^2C_4} & 0 & \frac{0}{1-C_1C_2^2C_4} & \frac{-C_1C_2K_0GR}{1-C_1C_2^2C_4} \\
0 & \frac{C_1C_2C_4}{1-C_1C_2^2C_4} & 0 & \frac{0}{-C_1C_2^2C_4} & 0 & \frac{-C_1C_2K_0GR}{1-C_1C_2^2C_4} & 0 & \frac{0}{1-C_1C_2^2C_4} \\
0 & \frac{-K_0GR}{L_a} & 0 & 0 & \frac{-K_0GR}{L_a} & 0 & \frac{0}{L_a} & \frac{-R_a/L_a}{L_a} \\
0 & 0 & 0 & 0 & -K_0GR/L_a & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -K_0GR/L_a & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\theta_1 \\
\omega_1 \\
\theta_2 \\
\omega_2 \\
i_1 \\
i_2
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
r_{a1} \\
r_{a2}
\end{bmatrix} (117)
\]

Equation (117) can be written in short-hand as follows:

\[
\dot{x} = Ax + Bu
\]  

(118)

Meanwhile, the output equation for the state space representation of the H-Bot is as follows:

\[
\begin{bmatrix}
\theta_1 \\
\omega_1 \\
\theta_2 \\
\omega_2 \\
i_1 \\
i_2
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\theta_1 \\
\omega_1 \\
\theta_2 \\
\omega_2 \\
i_1 \\
i_2
\end{bmatrix} (119)
\]

This can be written in short-hand as follows:

\[
y = Cx
\]  

(120)

5.2.2 Cost Function

The cost function \( J \) to optimize energy for an H-Bot can be defined as follows [93]:

\[
J = \frac{1}{2} (Cx(T) - r_d(T))^TP(Cx(T) - r_d(T)) + \frac{1}{2} \int_{t_0}^{T} ((Cx - r_d)^TQ(Cx - r_d) + u^TRu)dt
\]  

(121)
Where, \( T \) is the final time of the trajectory, \( r_d \) is the desired reference trajectory within the interval of time \([t_0, T]\), \( P \) is a weighting matrix that determines the cost with the accuracy to reach the target position, \( Q \) is a real symmetric positive semi-definite matrix in which each element of the matrix weights the states \((\theta_1 \text{ and } \theta_2)\) to track the desired trajectory \( r_d \), and \( R \) is a real symmetric positive definite weighting matrix that determines the cost associated with energy.

5.2.3 Optimization Method

Substituting (118) and (121) into the Hamiltonian equation defined in (83), the following is obtained:

\[
H = \frac{1}{2} (Cx - r_d)^T Q (Cx - r_d) + \frac{1}{2} u^T Ru + \lambda^T (Ax + Bu)
\]  

(122)

Which can be rewritten as follows by expanding each term:

\[
H = \frac{1}{2} (Cx)^T Q Cx - \frac{1}{2} (Cx)^T Q r_d - \frac{1}{2} r_d^T Q Cx + \frac{1}{2} r_d^T Q r_d + \frac{1}{2} u^T Ru + \lambda (Ax + Bu)
\]  

(123)

The necessary condition [93] derived from (123) are as follows:

Costate Equation: \[ \dot{\lambda} = -H_x = -C^T Q Cx + C^T Q r_d - A^T \lambda \]  

(124)

System Equation: \[ \dot{x} = H_{\lambda} = Ax + Bu \]  

(125)

Stationary Condition: \[ 0 = H_u = Ru + B^T \lambda \] 
\[ u = -R^{-1} B^T \lambda \]  

(126)
And merging (126) into (125), the system equation can be written as follows:

$$\dot{x} = Ax - BR^{-1}B^T \lambda$$  \hspace{1cm} (127)$$

The boundary condition is defined as follows [93]:

$$\lambda(T) = \partial \phi / \partial x(T)$$  \hspace{1cm} (128)$$

Where $\phi$ is the cost at the final time given in the cost function (121). Thus, replacing the cost of the final time from (121) into (128), the following is obtained:

$$\lambda(T) = \frac{\partial}{\partial x} \left[ \frac{1}{2} (Cx(T) - r_d(T))^T P (Cx(T) - r_d(T)) \right]$$

$$\lambda(T) = \frac{\partial}{\partial x} \left[ \frac{1}{2} (Cx(T))^T PCx(T) - \frac{1}{2} (Cx(T))^T PCR_d(T) - \frac{1}{2} r_d(T)^T P Cx(T) + \frac{1}{2} r_d(T)^T P r_d(T) \right]$$  \hspace{1cm} (129)$$

$$\lambda(T) = C^T PCx(T) - \frac{1}{2} C^T Pr_d(T) - \frac{1}{2} C^T Pr_d(T)$$

$$\lambda(T) = C^T PCx(T) - C^T Pr_d(T)$$

Since the costate equation (124) develops backwards in time, the control equation (126) can not be solved. Thus, an auxiliary term $v(t)$ can be added to compute the control effort and the states as a function of time [93]. Thus, the Lagrange multiplier $\lambda$ can be defined as follows:

$$\dot{\lambda}(t) = S(t)x(t) - v(t)$$  \hspace{1cm} (130)$$
Where, the matrix sequence $S$ is defined as $S(T) = C^TPC$, while the auxiliary function $v$ is defined as $v(T) = C^TPr(T)$. Calculating the derivative of (130), the following is obtained:

$$
\dot{\lambda}(t) = \dot{S}x + S\dot{x} - \dot{v}
$$

(131)

Substituting the system equation (127) into (131), the following is obtained:

$$
\dot{\lambda}(t) = \dot{S}x + S(Ax - BR^{-1}B^T\lambda) - \dot{v}
$$

(132)

$$
\dot{\lambda}(t) = \dot{S}x + SAx - SBR^{-1}B^T\lambda - \dot{v}
$$

Substituting (130) back into (132), the first equation for $\dot{\lambda}(t)$ is obtained:

$$
\dot{\lambda}(t) = \dot{S}x + SAx - SBR^{-1}B^T(Sx - v) - \dot{v}
$$

(133)

$$
\dot{\lambda}(t) = \dot{S}x + SAx - SBR^{-1}B^TSx - SBR^{-1}B^Tv - \dot{v}
$$

Meanwhile, if (130) is substituted into the costate equation (124), the second equation for $\dot{\lambda}(t)$ is obtained:

$$
\dot{\lambda} = -C^TQCx + C^TQr_d - A^T(Sx - v)
$$

(134)

$$
\dot{\lambda} = -C^TQCx + C^TQr_d - A^TSx - A^TSv
$$

Combining (133) and (134):

$$
\dot{S}x + SAx - SBR^{-1}B^T(Sx - v) - \dot{v} = -C^TQCx + C^TQr_d - A^TSx - A^TSv
$$

(135)
And equating the x-terms, the following is obtained:

\[ \dot{S} + SA - SBR^{-1}B^T S + C^TQC + A^T S = 0 \]  

(136)

And equating the other terms, the following is obtained:

\[ \dot{v} - SBR^{-1}B^Tv + C^T Q r_d + A^Tv = 0 \]  

(137)

Thus, (136) and (137) can be used to compute \( S \) and \( v \) backwards in time. The initial conditions for (136) and (137) is \( S(T) \) and \( v(T) \), respectively, which can be obtained as follows by combining (129) and (130):

\[ S(T) = C^T P C \]  

(138)

\[ v(T) = C^T P r_d(T) \]

Meanwhile, \( r_d \) is the desired trajectory calculated backwards in time. For the H-Bot, the trajectory in the Cartesian space is a circular move with radius \( R_{path} \) calculated backward in time as follows:

\[ X(T - t_{cub}) = R_{path} \sin(2\pi t_{cub}) \]  

(139)

\[ Y(T - t_{cub}) = R_{path} \cos(2\pi t_{cub}) \]

Where, \( t_{cub} \) is defined as a cubic cam profile in order to yield a smooth motion for the optimized motion profile calculated from this energy optimization method. Thus, while the energy optimization method computes a motion profile to optimize energy, the
smoothness of this optimized motion is obtained by defining the desired trajectory with a time vector that follows a cubic cam defined as follows:

\[
    t_{cub} = -2 \left( \frac{T - t}{T} \right)^3 + 3 \left( \frac{T - t}{T} \right)^2
\]  

Thus, with the move time \( T \) given, the cubic time profile \( t_{cub} \) can be calculated from (140) and applied to (139) to obtain the coordinates \( X \) and \( Y \) for a circular move, which can be applied to the inverse kinematics of the system given in (19) to compute the angular position for motor M1 and M2, \( \theta_1 \) and \( \theta_2 \) respectively, and obtain the desired trajectory \( r_d \):

\[
    r_d = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}
\]  

The desired trajectory \( r_d \) is used to compute \( v \) in (138). With \( S \) and \( v \) computed backwards in time, \( \lambda \) can be computed from (130), the control effort \( u \) can be calculated from (126), and the states \( x \) can be calculated from (118), all forward in time.

The energy optimization method described in this section is summarized in Fig. 5-14 and detailed in Fig. 5-15. In this investigation, it is assumed that the optimized trajectory for the H-Bot must remain within pre-defined boundaries in the Cartesian space. These boundaries are application dependent. In order to calculate a trajectory for the H-Bot that optimally fits these boundaries, an iterative process was developed to compute the coefficients \( P \) and \( Q \) of cost function (121). The energy optimization process with this iterative process to compute \( P \) and \( Q \) from the cost function is shown in Fig. 5-15.
Fig. 5-14 - Energy optimization algorithm
Fig. 5-15 - Energy Optimization Method for an H-Bot.
The proposed method to recalculate $P$ through an iterative process is shown in Fig. 5-16. This iterative process to recalculate $P$ is as follows. For each iteration, the forward kinematics of the H-Bot given in (142) is used to calculate the optimized trajectory of the end-effector in the Cartesian space ($x_{optimal}$, $y_{optimal}$) as shown in (142). This calculation is based on the optimized angular positions ($\theta_1(t)$, $\theta_2(t)$) that are calculated from the optimization method described in Fig. 5-15. In (142), $r$ is the radius of the pulleys connected to the motors.

\[
x_{optimal}(t) = -\frac{r}{2} (\theta_1(t) + \theta_2(t))
\]

\[
y_{optimal}(t) = \frac{r}{2} (\theta_1(t) - \theta_2(t))
\]

(142)

The error ($Err$) between the optimized final state ($x_{optimal}(T)$, $y_{optimal}(T)$) and the desired (ideal) final state ($X(T)$, $Y(T)$) is then calculated as follows:

\[
Err = \sqrt{(x_{optimal}(T) - X(T))^2 + (y_{optimal}(T) - Y(T))^2}
\]

(143)

Based on this error $Err$, the weighting matrix $P$ is recalculate at each iteration. If the error $Err$ is greater than the maximum error ($Err_{max}$) allowed between the desired final state ($X(T)$, $Y(T)$) and the optimized final state ($x_{optimal}(T)$, $y_{optimal}(T)$), then $P$ is recalculated as shown in (144) at each new iteration. The maximum error $Err_{max}$ is defined based on the machine functional specifications.

\[
P = P + P[\abs(1 - Err/Err_{max})]
\]

(144)
Otherwise, $P$ is recalculated as follows:

$$P = P - P[\text{abs}(1 - \text{Err}/\text{Err}_{\text{max}})]$$  \hfill (145)

This iterative process to recalculate $P$ is shown in Fig. 5-16.

![Fig. 5-16 - Method to recalculate $P$ for the energy optimization method of an H-Bot.](image)

Meanwhile, the proposed method to recalculate the weighting matrix $Q$ through an iterative process is shown in Fig. 5-17. This method is as follows. The center of the desired trajectory in $x$-direction ($X_0$) and $y$-direction ($Y_0$) can be calculated for a circular move as follows:
\[ X_0 = \frac{\max(X(t)) + \min(X(t))}{2} \]

\[ Y_0 = \frac{\max(Y(t)) + \min(Y(t))}{2} \]  

Fig. 5-17 - Method to recalculate \( Q \) for the energy optimization of an H-Bot.
The distance, $Err_{path}(t)$, from the center of a desired circular move $(X_0, Y_0)$ in Cartesian space to the optimized trajectory $(x_{optimal}(t), y_{optimal}(t))$ defined in (142), is calculated as follows:

$$Err_{path}(t) = \sqrt{(x_{optimal}(t) - X_0)^2 + (y_{optimal}(t) - Y_0)^2}$$  \hspace{1cm} (147)$$

The $Err_{path}(t)$ is the better described in Fig. 5-18a. In this figure, the optimized trajectory $(x_{optimal}(t), y_{optimal}(t))$ during one of the iterations, and the center of the desired trajectory $(X_0, Y_0)$ used to compute $Err_{Path}(t)$ are shown. The boundaries for the optimized trajectory shown in Fig. 5-18b is defined by $R_{tolerance}$, which is a user-defined parameter.

Fig. 5-18 - Identification of $Err_{path}$, desired trajectory, boundaries, and optimal trajectory.
The closest point, \( \text{Err}_{\text{min}} \), of the optimized trajectory \((x_{\text{optimal}}(t), y_{\text{optimal}}(t))\) to the boundaries of the desired trajectory \((R_{\text{path}} - R_{\text{tolerance}})\), where \(R_{\text{path}}\) is the radius of the desired trajectory as shown in Fig. 5-18, is calculated as follows:

\[
\text{Err}_{\text{min}} = \min \left[ \text{abs} \left( R_{\text{path}}(t) \right) \right] \tag{148}
\]

The weighting matrix \( Q \) is then recalculated at each iteration of the energy optimization process based on the closest point between the optimized trajectory \( \text{Err}_{\text{min}} \) and the boundaries of the desired trajectory \((R_{\text{path}} - R_{\text{tolerance}})\) as follows:

\[
\text{If } \text{Err}_{\text{min}} > R_{\text{path}} - R_{\text{tolerance}} \quad Q = Q - Q \left[ \text{abs} \left( 1 - \frac{\text{Err}_{\text{min}}}{R_{\text{path}} - R_{\text{tolerance}}} \right) \right] \tag{149}
\]

\[
\text{Else} \quad Q = Q + Q \left[ \text{abs} \left( 1 - \frac{\text{Err}_{\text{min}}}{R_{\text{path}} - R_{\text{tolerance}}} \right) \right]
\]

5.2.4 Simulation Results

A circular trajectory in the Cartesian space with radius of 0.04 meters was chosen to demonstrate the proposed energy optimization method. The maximum error \( R_{\text{tolerance}} \) allowed between the desired trajectory \((x(t), y(t))\) and the optimized trajectory \((x_{\text{optimal}}(t), y_{\text{optimal}}(t))\) was chosen to be 0.005 meters, while the maximum error \( \text{Err}_{\text{max}} \) at the final position between the desired trajectory and the optimized trajectory was chosen to be 0.0001 meter. The move time \( T \) was chosen to be 5 seconds.
After defining the desired trajectory and the boundaries for this trajectory, the initial condition for the weighting matrices $Q$ and $P$ can be chosen. Arbitrary initial values for $P$ and $Q$ can be selected since the proposed iterative process in the energy optimization method forces $P$ and $Q$ to converge to the same values independently of the initial values. The final values for $Q$ and $P$ will result in a motion profiles for $M_1$ and $M_2$ within the trajectory boundaries that optimizes the motor electrical energy.

The parameters of the motor and mechanical system used in the simulation results are given in Table 4-6 and Table 4-7.

The performance of this proposed iterative process to calculate an optimized motion profile using random initial conditions for $Q$ and $P$ are shown in Table 5-4 and Table 5-5. The progression of the values of $P$ and $Q$ converging from arbitrary initial values to specific values after 60 iterations is shown in Table 5-4. It is also shown in this table that $Q$ converges to 28 and $P$ to 110 after 60 iterations independently of the initial values. The results shown in Table 5-4 and Table 5-5 are for three test conditions: (1) initial condition of $Q$ less than its final value and initial condition of $P$ greater than its final value, (2) initial conditions of $P$ and $Q$ less than the respective final values, and (3) initial condition of $Q$ greater than its final value and initial condition of $P$ less than its final value.
Table 5-4 – Results for iterative process to compute Q and P

<table>
<thead>
<tr>
<th>Q, R, and P</th>
<th>Q values for 60 iterations</th>
<th>P values for 60 iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial values:</td>
<td><img src="image1" alt="Graph 1" /></td>
<td><img src="image2" alt="Graph 2" /></td>
</tr>
<tr>
<td>Q = [1 0]</td>
<td>30</td>
<td>500</td>
</tr>
<tr>
<td>R = [1 0]</td>
<td>25</td>
<td>200</td>
</tr>
<tr>
<td>P = [0 1]</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>Final values:</td>
<td><img src="image3" alt="Graph 3" /></td>
<td><img src="image4" alt="Graph 4" /></td>
</tr>
<tr>
<td>Q = [27.8 0]</td>
<td>15</td>
<td>50</td>
</tr>
<tr>
<td>R = [1 0]</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>P = [0 1]</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Initial values:

| Initial values: | ![Graph 5](image5) | ![Graph 6](image6) |
| Q = [1 0] | 30 | 500 |
| R = [1 0] | 25 | 200 |
| P = [0 1] | 20 | 100 |

Initial values:

| Initial values: | ![Graph 7](image7) | ![Graph 8](image8) |
| Q = [500 0] | 30 | 500 |
| R = [1 0] | 25 | 200 |
| P = [0 1] | 20 | 100 |

Initial values:

| Initial values: | ![Graph 9](image9) | ![Graph 10](image10) |
| Q = [27.9 0] | 30 | 500 |
| R = [1 0] | 25 | 200 |
| P = [0 1] | 20 | 100 |

Initial values:

| Initial values: | ![Graph 11](image11) | ![Graph 12](image12) |
| Q = [110.7 0] | 30 | 500 |
| R = [100 0] | 25 | 200 |
| P = [110 0] | 20 | 100 |

Initial values:

| Initial values: | ![Graph 13](image13) | ![Graph 14](image14) |
| Q = [110.7 0] | 30 | 500 |
| R = [100 0] | 25 | 200 |
| P = [110 0] | 20 | 100 |

Initial values:

| Initial values: | ![Graph 15](image15) | ![Graph 16](image16) |
| Q = [110.7 0] | 30 | 500 |
| R = [100 0] | 25 | 200 |
| P = [110 0] | 20 | 100 |

Initial values:

| Initial values: | ![Graph 17](image17) | ![Graph 18](image18) |
| Q = [110.7 0] | 30 | 500 |
| R = [100 0] | 25 | 200 |
| P = [110 0] | 20 | 100 |
The performance of this iterative energy optimization method in calculating an optimized trajectory that resides inside the pre-defined boundaries is shown in Table 5-5. Even if the initial trajectory resides outside the boundaries due to random initial values of $Q$ and $P$, this proposed method recalculate trajectories that progressively approach the boundaries at each iteration. This effect of moving the optimized trajectory towards a region defined by the boundaries is mostly due to the recalculation of $Q$.

Table 5-5 – Optimized trajectory for various initial $P$ and $Q$ values.

<table>
<thead>
<tr>
<th>$Q$, $R$, and $P$</th>
<th>Progression of the optimized trajectory for each iteration of the optimization method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initial values:</strong></td>
<td></td>
</tr>
<tr>
<td>$Q = \begin{bmatrix} 1 &amp; 0 \ 0 &amp; 1 \end{bmatrix}$</td>
<td></td>
</tr>
<tr>
<td>$R = \begin{bmatrix} 1 &amp; 0 \ 0 &amp; 1 \end{bmatrix}$</td>
<td></td>
</tr>
<tr>
<td>$P = \begin{bmatrix} 500 &amp; 0 \ 0 &amp; 500 \end{bmatrix}$</td>
<td></td>
</tr>
<tr>
<td><strong>Final values:</strong></td>
<td></td>
</tr>
<tr>
<td>$Q = \begin{bmatrix} 27.8 &amp; 0 \ 0 &amp; 27.8 \end{bmatrix}$</td>
<td></td>
</tr>
<tr>
<td>$R = \begin{bmatrix} 1 &amp; 0 \ 0 &amp; 1 \end{bmatrix}$</td>
<td></td>
</tr>
<tr>
<td>$P = \begin{bmatrix} 110.7 &amp; 0 \ 0 &amp; 110.7 \end{bmatrix}$</td>
<td></td>
</tr>
</tbody>
</table>
Initial values:
\[
Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \\
R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \\
P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\]

Final values:
\[
Q = \begin{bmatrix} 27.8 & 0 \\ 0 & 27.8 \end{bmatrix}, \\
R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \\
P = \begin{bmatrix} 110.6 & 0 \\ 0 & 110.6 \end{bmatrix}
\]
Meanwhile, the weighting matrix $P$ determines the accuracy in reaching the target position. In this developed iterative energy optimization process, $P$ is recalculated at each iteration to force the final state of the optimized trajectory into the boundaries defined by the maximum error at the target position $Err_{max}$. The efficiency of the proposed iterative energy optimization method to make the final position of the optimized trajectory reside into or at the boundaries defined by $Err_{max}$ by recalculating $P$, is shown in Table 5-6.

### Table 5-6 – Effect of $P$ to reach target position

<table>
<thead>
<tr>
<th>$Q$, $R$, and $P$</th>
<th>Progression of the optimized trajectory for each iteration of the optimization method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initial values:</strong></td>
<td><img src="image" alt="Optimized trajectory progression" /></td>
</tr>
<tr>
<td>$Q = \begin{bmatrix} 1 \ 0 \ 1 \end{bmatrix}$</td>
<td></td>
</tr>
<tr>
<td>$R = \begin{bmatrix} 1 \ 0 \ 1 \end{bmatrix}$</td>
<td></td>
</tr>
<tr>
<td>$P = \begin{bmatrix} 500 \ 0 \ 500 \end{bmatrix}$</td>
<td></td>
</tr>
<tr>
<td><strong>Final values:</strong></td>
<td><img src="image" alt="Optimized trajectory" /></td>
</tr>
<tr>
<td>$Q = \begin{bmatrix} 27.8 \ 0 \ 0 \end{bmatrix}$</td>
<td></td>
</tr>
<tr>
<td>$R = \begin{bmatrix} 1 \ 0 \ 1 \end{bmatrix}$</td>
<td></td>
</tr>
<tr>
<td>$P = \begin{bmatrix} 110.7 \ 0 \ 110.7 \end{bmatrix}$</td>
<td></td>
</tr>
</tbody>
</table>

| **Initial values:** | ![Optimized trajectory](image) |
| $Q = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$ |  |
| $R = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$ |  |
| $P = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$ |  |
| **Final values:** | ![Optimized trajectory](image) |
| $Q = \begin{bmatrix} 27.8 \\ 0 \\ 0 \end{bmatrix}$ |  |
| $R = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$ |  |
| $P = \begin{bmatrix} 110.6 \\ 0 \\ 110.6 \end{bmatrix}$ |  |
After completion of the iterative process to calculate $P$ and $Q$, the final optimized angular motion profile $\theta_1$ and $\theta_2$, which are the outputs of the optimization method in Fig. 5-15, are shown in Fig. 5-19.

During these simulations, the weighting matrix $R$ from the cost function in (121) was kept constant. This is because independently of the value of $R$ within certain limits, the optimized angular position $\theta_1$ and $\theta_2$ calculated by the iterative energy optimization
method results in the same profile shown in Fig. 5-19, although the final values for $P$ and $Q$ change from those shown in Table 5-4 as $R$ changes. An example is shown in Fig. 5-20 with the final angular position $\theta_1$ and $\theta_2$ calculated with $R$ set to $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and then $\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$.

As shown in this figure, the final values for $\theta_1$ and $\theta_2$ were the same independently of the values set for $R$. For reference, when $R$ was set to $1 \times I_{(2x2)}$, the final values of $P$ and $Q$ were $28 \times I_{(2x2)}$ and $111 \times I_{(2x2)}$, respectively, as shown in Table 5-4, and when $R$ was set to $10 \times I_{(2x2)}$, the final values for $P$ and $Q$ were $277 \times I_{(2x2)}$ and $1014 \times I_{(2x2)}$, respectively.

![Angular position $\theta_1$ and $\theta_2$ calculated with $R=1 \times I_{(2x2)}$ and $R=10 \times I_{(2x2)}$.]

Fig. 5-20 - Angular position $\theta_1$ and $\theta_2$ calculated with $R=1 \times I_{(2x2)}$ and $R=10 \times I_{(2x2)}$.

In order to validate the effectiveness of the proposed iterative energy optimization method in calculating a trajectory that saves energy, the energy required by the motors of
the H-Bot to perform this optimized trajectory was compared to the energy required by the motors to perform the same desired circular trajectory with radius of 0.04m in 5 sec when defined by eight motion profiles typically used in industrial applications. These motion profiles are the 5th-order polynomial, 7th-order polynomial, 9th-order polynomial, sine, trapezoidal, cycloidal, modsine, and cubic profile. The energy required by the motors of the H-Bot to perform this circular trajectory designed with these motion profiles was estimated with the model shown in Fig. 5-21. These eight motion profiles defined in (1), (2), (3), (4), (5), (6), (7), (8), (9), and (10) were implemented in the subsystem called “Motion Profiles” with unitary magnitude and move time of 5 seconds. In the subsystem called “Inverse Kinematics” in Fig. 5-21, the equations defined in (139) were used to calculate \(X\) and \(Y\) values for a circular move in the Cartesian space. In this implementation, \(t_{cub}\) in (139) was replaced by the signal from the selected motion profiles calculated in the subsystem “Motion Profiles”. The \(X\) and \(Y\) values were applied to inverse kinematics equations defined in (19) to convert the trajectory in the Cartesian space into reference signal \(\theta_1\) and \(\theta_2\) that control the two motors of the H-Bot through the control system implemented in the subsystem called “Control System + Motors” in Fig. 5-21. Meanwhile, the optimized motion profiles for \(\theta_1\) and \(\theta_2\) shown in Fig. 5-19 were directly imported from the Matlab Workspace into the Simulink model as shown in Fig. 5-21 to feed the controller as shown in Fig. 5-21. The control system consists of a PI controller with velocity feedforward control for each motor. Each controller generates motor voltage command \(v_a\) for each motor of the H-Bot. The model of a dc motor was also implemented in this subsystem for each motor according to (115) and (116). These PI controllers were manually tuned to yield stable control and low position-following
error. The resulting proportional gains was 20, the integral gain was 4, and feedforward control was set to unitary gain. Once tuned, the gains of both PI controllers remained the same while estimating the energy for each motion profile. The model of the H-Bot was implemented according to (102) in the subsystem “H-Bot Direct Kinetic Model” in Fig. 5-21. The torque signal from the model of each motor feeds the torque command inputs $\tau_{m1}$ and $\tau_{m2}$ of the H-Bot model. The sub-system called “H-Bot Direct Kinematics” in Fig. 5-21 is the implementation of (142). This subsystem converts the angular position ($\theta_1$ and $\theta_2$) from the H-Bot model to linear position in the Cartesian space ($X$ and $Y$) in order to monitor the resulting trajectory of the system. The parameter “Selector” in Fig. 5-21 was used to automatically select via Matlab script the motion profile to be simulated and capture the results.

![Fig. 5-21 - Model of an H-Bot, control systems, and motors.](image)

While the system was being simulated for each one of these motion profiles, the voltage ($v_{a1}$ and $v_{a2}$) and current ($i_1$ and $i_2$) of each motor were used to compute the energy on each motor ($E_1$ and $E_2$) and the total energy ($E_t$) as follows:
\[ E_1 = \int_0^T v_{a1} i_1 \, dt \]

\[ E_2 = \int_0^T v_{a2} i_2 \, dt \]  \hspace{1cm} (150)

\[ E_t = E_1 + E_2 \]

The total energy \( E_t \) required by the system while commanded by each one of these eight motion profiles was compared to the total energy required by the system while commanded by the optimized motion profile. The results are shown in Fig. 5-22. The H-Bot was carrying a 2.7kg load during these simulations.

Fig. 5-22 - Total energy required by several motion profiles and the optimized motion profile.
The final energy required by the system to perform a circular trajectory in the Cartesian space of 0.04m radius in 5 seconds with a pay-load of 2.7kg while commanded by each one of these eight motion profiles including the optimized motion profile is shown in Table 5-7. From the results shown in Table 5-7, the optimized motion profile required the lowest amount of energy among all motion profiles tested in these simulations. The cubic and sine-harmonic profiles required 23.2% and 23.8% more energy than the optimized profile, respectively. The other motion profiles in Table 5-7 required even more energy.

Table 5-7 - Comparison of energy consumption for various types of motion profiles applied to the H-Bot system to move through a circular trajectory of radius of 0.04m in 5 seconds with 2.7kg load.

<table>
<thead>
<tr>
<th>Motion Profile</th>
<th>Final Energy (J)</th>
<th>Variation (%)</th>
<th>Max. Position-Following Error (rad)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized</td>
<td>3.0581</td>
<td>0</td>
<td>0.0254</td>
</tr>
<tr>
<td>Trapezoidal</td>
<td>3.9179</td>
<td>28.1</td>
<td>0.0288</td>
</tr>
<tr>
<td>Cycloidal</td>
<td>4.6741</td>
<td>52.8</td>
<td>0.0315</td>
</tr>
<tr>
<td>ModSine</td>
<td>4.2602</td>
<td>39.3</td>
<td>0.0297</td>
</tr>
<tr>
<td>Cubic</td>
<td>3.7672</td>
<td>23.2</td>
<td>0.0275</td>
</tr>
<tr>
<td>Sine-Harmonic</td>
<td>3.8687</td>
<td>26.5</td>
<td>0.0278</td>
</tr>
<tr>
<td>5th Order Polynomial</td>
<td>4.4571</td>
<td>45.7</td>
<td>0.0306</td>
</tr>
<tr>
<td>7th Order Polynomial</td>
<td>5.0748</td>
<td>65.9</td>
<td>0.0337</td>
</tr>
<tr>
<td>9th Order Polynomial</td>
<td>5.6334</td>
<td>84.2</td>
<td>0.0389</td>
</tr>
</tbody>
</table>

5.2.5 Experimental Results

The H-Bot was built for the experimental results as described in Section 4.2. The subsystem called “Position Reference” in Fig. 4-11 was implemented as in Fig. 5-21 to generate the reference signals $\theta_1$ and $\theta_2$. 
The same test performed in simulation with the H-Bot was also performed experimentally. The H-Bot was commanded to perform the circular trajectory with radius of 0.04 meters in 5 seconds with command signal ($\theta_1$ and $\theta_2$) defined by each one of the eight motion profiles typically used in industrial applications and listed in Table 5-7, and also with the optimized motion profile shown in Fig. 5-19.

The voltage and current of each motor were acquired during tests, and the resulting total energy $E_t$ defined in (150) was calculated while each motion profile was experimentally tested. The results for $E_t$ are shown in Fig. 5-23 and Table 5-8. As shown in Fig. 5-23, the energy continues increasing after completing the 5 seconds move. This is due to the energy used by the motor to hold position. As in the simulations, the optimized motion profile in Fig. 5-19 yielded significant energy savings, in comparison to the other motion profiles typically used in industrial applications. In these experimental results, the optimized motion profile required 11.6% less energy than the cubic profile which was the profile with the lowest energy cost from the eight typical motion profile tested against the optimized motion profile in this experiment. Thus, the optimized motion profile saved at least 11.6% energy in comparison to the other motion profiles shown in Table 5-8. These results validate the efficiency to the proposed iterative energy optimization method to compute motion profiles that allow to save the energy consumption of a coordinated multi-axis system by simply modifying the motion profile without any mechanical changes and still perform the same original task.
Fig. 5-23 - Energy consumption by the optimized motion profile and eight others for the H-Bot with a 2.7kg load.

Table 5-8 - Comparison of energy consumption for various types of motion profiles applied to the physical H-Bot system to move through a circular trajectory of radius of 0.04m in 5 seconds with 2.7kg load.

<table>
<thead>
<tr>
<th>Motion Profile</th>
<th>Final Energy (J)</th>
<th>Variation (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized</td>
<td>3.8670</td>
<td>0</td>
</tr>
<tr>
<td>Trapezoidal</td>
<td>4.4560</td>
<td>15.2</td>
</tr>
<tr>
<td>Cycloidal</td>
<td>4.5440</td>
<td>17.5</td>
</tr>
<tr>
<td>ModSine</td>
<td>4.3890</td>
<td>13.5</td>
</tr>
<tr>
<td>Cubic</td>
<td>4.3170</td>
<td>11.6</td>
</tr>
<tr>
<td>Sine-Harmonic</td>
<td>4.3500</td>
<td>12.5</td>
</tr>
<tr>
<td>5th Order Polynomial</td>
<td>4.4280</td>
<td>14.5</td>
</tr>
<tr>
<td>7th Order Polynomial</td>
<td>4.5460</td>
<td>17.6</td>
</tr>
<tr>
<td>9th Order Polynomial</td>
<td>4.7450</td>
<td>22.7</td>
</tr>
</tbody>
</table>
5.2.6 Discussion of Results

In Section 5.2, an iterative energy optimization method was investigated to minimize the electrical energy consumption of a Cartesian two-axis parallel robot – H-Bot. This method was validated by comparing the energy cost of the system while commanded by the optimized profile, to the energy cost of the system while commanded by eight motion profiles typically used in industrial applications. Simulations and experiments demonstrated significant energy savings with the optimized motion profile in comparison to these eight motion profiles. In comparison to the motion profiles to be avoided (trapezoidal, cubic, and sine-harmonic), the energy savings was between 23.2% and 28.1% from the simulations, and 11.6% and 15.2% from the experimental results. In comparison to the recommended motion profiles (cycloidal, modsine, 5th-order polynomial, 7th-order polynomial, and 9th-order polynomial), the energy savings was between 39.3% and 84.2% from the simulations, and 13.5% and 22.7% from the experimental tests. As shown through these results, the simulations predicted higher energy savings than the experimental results. This is potentially attributed to the higher position-following error with the experimental tests than in simulations. Since the Arduino board used to control the H-Bot would not allow to monitor position-following error without affecting the control performance, it was not possible to measure the position-following error with the physical system to validate this hypothesis.

These simulations and experimental results validate the iterative energy optimization method as an important tool to be considered for industrial applications to help minimize the energy consumption in industrial machines.
5.3 Self-Balancing Transporter

The energy optimization method described in Section 5.2.3 and summarized in Fig. 5-14 for the H-Bot was also applied to optimize the energy usage of the self-balancing transporter described in Sections 3.3 and 4.3.

As described in Section 3.3.5, the self-balancing transporter has two independent variables: the linear displacement $x$ and the tilt angle $\theta$. The energy optimization method was applied with the self-balancing transporter to optimize the motion profile for the linear displacement $x$ to move from the current position to the target position and minimize the motor energy usage. As for the H-Bot, two boundaries were defined for the linear displacement $x$. One boundary was defined as bands about the desired trajectory in which motion is allowed. The other boundary was defined at the final position as the maximum error that the system can have between the desired final position and the actual final position. The energy consumption of the self-balancing transporter, while commanded by the optimized motion profile to move in $x$-direction, was compared to the energy of the self-balancing transporter while commanded to move in $x$-direction by each one of the eight motion profiles used in industrial applications.

The energy optimization method applied to the self-balancing transporter is described next.

5.3.1 State-Space Equation

The state-space of the self-balancing transporter is given in (38) and (40).
5.3.2 Cost Function

The cost function of the energy optimization for the self-balancing transporter is the same used for the H-bot as defined in (121).

5.3.3 Optimization Method

The energy optimization method summarized in Fig. 5-14 and described in Section 5.2.3 for the H-Bot was also used to optimize the energy consumption of the self-balancing transporter. The differences reside in the calculation of the desired final trajectory \( r_d \) and in the iterative optimization of \( P \) and \( Q \) from the cost function in (121). These differences are described next.

The desired final trajectory \( r_d \) was defined as a function of the desired linear displacement \( x_d \), the desired tilt angle \( \theta_d \), and the desired linear velocity \( \dot{x}_d \). The desired tilt angle \( \theta_d \) was defined to be at zero during motion to force the system to balance vertically. Meanwhile, the desired trajectory \( x_d \) was defined to follow a cubic time profile \( t_{cub} \) which was the same type of motion profile used with the H-Bot in (140). Thus, the desired trajectory \( x_d \) was defined as follows:

\[
x_d = x_{target} t_{cub}
\]  
(151)

Where \( x_{target} \) is the target final position for the liner displacement \( x \) of the self-balancing transporter, and \( t_{cub} \) is the time signal defined as a cubic time profile as given in (140). Meanwhile, the desired linear velocity \( \dot{x}_d \) was calculated as the derivative of \( x_d \). Thus, the desired trajectory \( r_d \) is given as shown in (152). The desired linear velocity \( \dot{x}_d \) is part of \( r_d \) for force the final velocity of the optimized profile to be zero:
\[ r_d = \begin{bmatrix} \theta_d \\ \dot{x}_d \\ \ddot{x}_d \end{bmatrix} \quad (152) \]

It is assumed that the optimized trajectory in \( x \) must reside inside pre-defined boundaries defined by the path tolerance \( x_{\text{tolerance}} \). An example of a unitary trajectory for the linear displacement \( x \) of a self-balancing transporter with the respective path tolerance \( x_{\text{tolerance}} \) is shown in Fig. 5-24. This boundary \( x_{\text{tolerance}} \) is application dependent and it is then an input to the energy optimization method. This boundary \( x_{\text{tolerance}} \) is equivalent to \( R_{\text{tolerance}} \) shown in Fig. 5-18 for the H-Bot. This maximum allowed error at the final position \( Err_{\text{max}} \) is also shown in Fig. 5-24.

![Fig. 5-24 - Unit trajectory for the self-balancing transporter.](image)

The detailed energy optimization method with the iterative process to compute \( P \) and \( Q \) for the self-balancing transporter is shown in Fig. 5-25. The computation of the
desired trajectory $r_d$ as shown above is also in this figure. The other steps in this iterative energy optimization method shown in Fig. 5-25 are as described for Fig. 5-15.

Fig. 5-25 - Energy Optimization Method for a Self-Balancing Transporter.

The proposed method to optimize $P$ for the self-balancing transporter through an iterative process is shown in Fig. 5-26. The weighting matrix $P$ was used to optimize the final position of the linear displacement $x$ in a two-fold process. First, $P$ is recalculated to
force the final value of of the linear displacement \( x \) into a boundary defined at the final position by the parameter \( Err_{\text{max}} \) shown in Fig. 5-24. This maximum allowed error at the final position \( Err_{\text{max}} \) is application dependent and it is, therefore, an input to this energy optimization method. Second, after the final position of \( x \) been moved inside the boundary \( Err_{\text{max}} \), \( P \) is then iteratively recalculated to determine the final value of the linear displacement \( x \) inside the boundary \( Err_{\text{max}} \) that requires the least amount of energy. The simplified process to recalculate \( P \) is described in Fig. 5-26 and detailed process is given in Fig. 5-27.

**Fig. 5-26 - Method to recalculate \( P \) for the energy optimization method of a self-balancing transporter.**
In the iterative process to recalculate $P$, the absolute difference $Err$ between the desired target linear position $x_{\text{target}}$ and the optimized final position $x_{\text{optimal}}(T)$ obtained from $x(t)$ is calculated as follows:

$$Err = \text{abs}(x_{\text{target}} - x_{\text{optimal}}(T))$$

(153)

This difference $Err$ is then assessed to iteratively recalculated $P$ as defined in Fig. 5-26 and detailed in Fig. 5-27 to move the optimized final position $x_{\text{optimal}}(T)$ into the boundaries defined by $Err_{\text{max}}$ as shown in Fig. 5-24.

Once inside these boundaries, $P$ is iteratively recalculated as shown in Fig. 5-27 to minimize the motor energy $E$ given as:

$$E = \int_{t_0}^{T} v_a i_a \, dt$$

(154)

Where the motor voltage $v_a$ is the control input $u$ in the state-space in (39) and it is calculated from (126) and (130), while the motor current $i_a$ is obtained from the solution of the state-space equation forward in time as shown in Fig. 5-25. In Fig. 5-27, the suffix “rec” is for the variables recorded from the previous iteration step.
Fig. 5-27 – Detailed algorithm to iteratively recalculate $P$ to optimize the final position of $x(T)$.

The proposed iterative method to recalculate the weighting matrix $Q$ is shown in Fig. 5-28. The weighting matrix $Q$ is iteratively recalculated for two purposes. First, $Q$ is recalculated to obtain a motion profile for the linear displacement $x$ that resides inside the boundaries defined by $x_{\text{tolerance}}$ as shown in Fig. 5-24. Once inside the boundaries, $Q$ is then iteratively recalculated to optimize the trajectory of $x$ in terms of energy consumption $E$ that is obtained as given in (154). These two conditions to recalculate $Q$ are described next.
Fig. 5-28 - Method to recalculate $Q$ for the energy optimization method of a self-balancing transporter.

The process to recalculate $Q$ in order to obtain an optimized motion profile for the linear displacement $x$ that resided inside the boundaries defined by $x_{tolerance}$ is as follows. First, the maximum distance $Err_{max}$ between the optimized trajectory $x(t)$ and the desired trajectory $x_d(t)$ is calculated as shown in the pseudo-code in Fig. 5-29 to identify if the optimized trajectory $x(t)$ resided inside the boundaries defined by $x_{tolerance}$. 

Recalculate $Q$ to first obtain an optimized trajectory $x(t)$ that resides inside the boundaries defined by $x_{tolerance}$ and then to optimize the energy consumption $E$:

If $Err_{max} > x_{tolerance}$
Recalculate $P$ to move optimized final position $x_{target}(T)$
inside boundaries defined by $x_{tolerance}$.
Else
Recalculate $P$ to optimize energy $E$ once inside the boundaries at the final position.

Calculate energy consumption:

$E = \int_{t_0}^{T} u(t)x_5(t)dt$
**Loop**

\[ t = 0 \ldots T \]

- **If** \( x(t) > x_d(t) \)
  - **If** \( x(t) - x_d(t) > \text{Err}_{\text{max}1} \), **Then** \( \text{Err}_{\text{max}1} = x(t) - x_d(t) \)
  - **Else**
  - **If** \( x(t) - x_d(t) < \text{Err}_{\text{max}1} \), **Then** \( \text{Err}_{\text{max}2} = x(t) - x_d(t) \)

**Err_{max} = \max[\text{Err}_{\text{max}1} \, \text{abs}(\text{Err}_{\text{max}2})]**

**Fig. 5-29** - Pseudo-code to calculate the distance between the optimized trajectory \( x(t) \) and the desired trajectory \( x_d(t) \).

Then, if any portion of the optimized trajectory \( x(t) \) falls outside the boundaries defined by \( x_{\text{tolerance}} \), the weighting matrix \( Q \) is iteratively recalculate as shown in **Fig. 5-30** to force the trajectory \( x(t) \) into the boundaries.

- **If** \( \text{Err}_{\text{max}1} > \text{abs}(\text{Err}_{\text{max}2}) \)
  \[ Q = Q - \text{abs}(Q \, \frac{\text{Err}_{\text{max}}}{x_{\text{tolerance}}}) \]
- **Else**
  \[ Q = Q + \text{abs}(Q \, \frac{\text{Err}_{\text{max}}}{x_{\text{tolerance}}}) \]

**Fig. 5-30** – Pseudo-code to recalculate \( Q \) to obtain an optimize \( x(t) \) that resides within the boundaries defined by \( x_{\text{tolerance}} \).

Once the optimized trajectory \( x(t) \) resides inside the pre-defined boundaries, the process of recalculating \( Q \) is switched from the process given in **Fig. 5-30** of forcing \( x(t) \) into the boundaries, to the process given in **Fig. 5-31** of optimizing the trajectory \( x(t) \) to
minimize the energy consumption $E$ defined in (154). This process of optimizing $x(t)$ is based on the comparison of the energy consumption $E$ obtained from (154) with the energy consumption from the previous iteration $E_{rec}$.

<table>
<thead>
<tr>
<th>If the motor energy $E$ increased in comparison to the previous iteration $E_{rec}$:</th>
</tr>
</thead>
<tbody>
<tr>
<td>If $Q &gt; Q_{rec}$</td>
</tr>
<tr>
<td>$Q = Q - Q \left[ \text{abs} \left( 1 - \frac{E_{rec}}{E} \right) \right]$</td>
</tr>
<tr>
<td>Else</td>
</tr>
<tr>
<td>$Q = Q + Q \left[ \text{abs} \left( 1 - \frac{E_{rec}}{E} \right) \right]$</td>
</tr>
<tr>
<td>If the motor energy $E$ reduced in comparison to the previous iteration $E_{rec}$:</td>
</tr>
<tr>
<td>If $Q &gt; Q_{rec}$</td>
</tr>
<tr>
<td>$Q = Q + Q \left[ \text{abs} \left( 1 - \frac{E_{rec}}{E} \right) \right]$</td>
</tr>
<tr>
<td>Else</td>
</tr>
<tr>
<td>$Q = Q - Q \left[ \text{abs} \left( 1 - \frac{E_{rec}}{E} \right) \right]$</td>
</tr>
</tbody>
</table>

Fig. 5-31 – Pseudo-code to recalculate $Q$ to optimize $x(t)$ in terms of energy when the trajectory of $x(t)$ resides within the boundaries defined by $x_{tolerance}$.

### 5.3.4 Simulation Results

After defining the desired trajectory and the boundaries for this trajectory, the initial condition for the weighting matrices $Q$ and $P$ need to be defined. Since, the proposed iterative process described in the previous section forces $Q$ and $P$ to converge to the same value independently of the initial condition, the initial values for $P$ and $Q$ can be arbitrarily defined. The final values for $Q$ and $P$ will result in a motion profile within the boundaries of the trajectory and with minimum energy cost.

A target position $x_{target}$ for the self-balancing transporter was chosen to be 2 meters with a path tolerance $x_{tolerance}$ of 0.2 meters, and boundary at the final position $Err_{max}$ of 0.04 meters. This trajectory $x(t)$ was chosen to be completed in 5 seconds,
which keeps the peak speed for this move below the limits of the self-balancing transporter defined in Section 3.3.1.

The performance of this proposed iterative process of calculating an optimized motion profile for $x(t)$ using arbitrary initial conditions for $Q$ and $P$ are shown in Table 5-9, and Table 5-10. As shown in Table 5-9, $Q$ and $P$ converged to the same final values independently of the initial values. The progression of this iterative process to converge the values of $Q$ and $P$ is shown in Table 5-9 for the following test conditions: (1) initial condition of $Q$ and $P$ greater than the final values, (2) initial conditions of $P$ and $Q$ lower than the final values, and (3) initial condition of $Q$ greater than its final value and $P$ lower than its final value.

Table 5-9 – Results for iterative process to compute $Q$ and $P$ with random initial conditions for a move time of 5 seconds, $x_{\text{target}} = 2\text{m}$, $x_{\text{tolerance}} = 0.2\text{m}$, and $Err_{\text{max}} = 0.04\text{m}$.  

<table>
<thead>
<tr>
<th>$Q, R, \text{ and } P$</th>
<th>$Q$ values</th>
<th>$P$ values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initial values:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Q = 10I_{3x3}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P = 1000I_{3x3}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R = 1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Final values:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Q = 3.9I_{3x3}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P = 419I_{3x3}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R = 1$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The performance of this iterative energy optimization method in calculating an optimized trajectory that resides inside the pre-defined boundaries is shown in Table 5-10. Even if the initial trajectory resides outside the boundaries due to arbitrary initial values of $Q$ and $P$, this proposed iterative method recalculates trajectories that progressively move towards the region inside the boundaries at each iteration. This effect of moving the optimized trajectory towards a region defined by boundaries is mostly due to the recalculated of $Q$.

Meanwhile, the weighting matrix $P$ determines the accuracy of reaching the target position. In this proposed iterative energy optimization process, $P$ is recalculated at each iteration to force the final state of the optimized trajectory into the boundaries defined by

<table>
<thead>
<tr>
<th>Initial values:</th>
<th>Final values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q = 1 I_{3x3}$</td>
<td>$Q = 3.3 I_{3x3}$</td>
</tr>
<tr>
<td>$P = 10 I_{3x3}$</td>
<td>$P = 428 I_{3x3}$</td>
</tr>
<tr>
<td>$R = 1$</td>
<td>$R = 1$</td>
</tr>
</tbody>
</table>
the maximum error at the target position $Err_{max}$. The efficiency of the proposed iterative energy optimization method in recalculate $P$ to make the final position of the optimized trajectory reside inside or at the boundaries defined by $Err_{max}$ is shown in Table 5-10. It can also be noticed in Table 5-10 that the tilt angle $\theta(t)$ remains about zero degrees during move, which means that the self-balancing transports stays vertically balanced.

Table 5-10 – Resulting energy and trajectory for iterative process of computing $Q$ and $P$ with arbitrary initial conditions for a move time of 5 seconds, $x_{\text{target}} = 2m$, $x_{\text{tolerance}} = 0.2m$, and $Err_{max} = 0.04m$.

<table>
<thead>
<tr>
<th>$Q$, $R$, and $P$</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy for 150 iterations:</td>
<td>Optimized trajectories for 150 iterations:</td>
</tr>
</tbody>
</table>

Initial values:
$Q = 10I_{3x3}$
$P = 1000I_{3x3}$
$R = 1$

Final values:
$Q = 3.9I_{3x3}$
$P = 419I_{3x3}$
$R = 1$
Initial values:
$Q = 1 I_{3x3}$
$P = 10 I_{3x3}$
$R = 1$

Final values:
$Q = 3.3 I_{3x3}$
$P = 42 I_{3x3}$
$R = 1$

Energy for 150 iterations:

Optimized trajectories for 150 iterations:

Zoomed final trajectories:

Final trajectory:
During these simulations of the self-balancing transporter, the weighting matrix $R$ from the cost function in (121) was kept constant while optimizing $P$ and $Q$. This was the same approach used with the H-Bot as described in Section 5.1.4. The weighting matrix $R$ can stay constant because independently of the value of $R$ within certain limits, the optimized linear trajectory $x(t)$ calculated by this iterative energy optimization method results in the same profile for the linear displacement $x$ as shown in Table 5-10, although
the final values of $P$ and $Q$ change from those shown in Table 5-10 as $R$ changes. An example is shown in Fig. 5-32 with the final linear displacement $x(t)$ calculated with $R = 1$ and then $R = 10$. As shown in this figure, the final profile for $x(t)$ was the same independently of the values set for $R$. For reference, when $R$ was set to 1, the final values of $P$ and $Q$ were $428 \times I_{(3 \times 3)}$ and $3.3 \times I_{(3 \times 3)}$, respectively, as show in Table 5-9, while the final values for $P$ and $Q$ were $4211 \times I_{(3 \times 3)}$ and $53.4 \times I_{(3 \times 3)}$, respectively, when $R$ was set to 10.

Fig. 5-32 – Linear displacement $x(t)$ calculated with $R = 1$ and $R = 10$.

In order to validate the effectiveness of the proposed iterative energy optimization method in calculating a trajectory that reduces the energy cost, the energy required from the motors of the self-balancing transporter to perform the optimized trajectory was compared to the energy required by the motors to perform the same desired linear move $x$ of 2 meters in 5 seconds when defined by eight motion profiles typically used in
industrial applications. The energy required by the motors of the self-balancing transporter to perform this linear move designed with these motion profiles was estimated with the model shown below in Fig. 5-33. This model in Fig. 5-33 is an extension of the model shown in Fig. 3-40. The difference resides in the command signal. In Fig. 3-40, the command signal is the implementation of the signal in Fig. 3-30, while in Fig. 5-33, the command signal is the implementation of these eight motion profiles and the optimized motion profile. These eight motion profiles defined in (1), (2), (3), (4), (5), (6), (7), (8), (9), and (10) were implemented in the subsystem called “Motion Profiles” for a 2 meter move in 5 seconds. Meanwhile, the optimized motion profile shown in Table 5-10 was directly imported from the Matlab Workspace into the Simulink model as shown in Fig. 5-33.

The self-balancing transporter was then commanded by each of these motion profiles. State-space control was used to control the system as described in Section 3.3.10 and tuned as described in Section 3.3.11 with \( Q = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \). The control system generates the motor voltage command \( v_a \) for the motors. The tuning gains remained the same while estimating the energy for each motion profile. The torque signal from the model of both motors feeds the torque command inputs of the self-balancing transporter. The parameter “Selector” in Fig. 5-33 was used to automatically select, via Matlab script, the motion profile to be simulated while the results are captured.
While the system was being simulated for each one of these motion profiles, the motor voltage $v_a$ and motor current $i_a$ of one of the motors were used to compute the energy of one motor using (79) and (81). The total energy $E_t$ was calculated as the twice the energy of a single motor. Two motors were modeled in the subsystem “DC Motors” in Fig. 5-33.

The total energy $E_t$ required by the system while commanded by each one of these eight motion profiles was compared to the total energy required by the system while commanded by the optimized motion profile. The results are shown in Fig. 5-34.
Fig. 5-34 - Total energy required by the self-balancing transporter while commanded by several motion profiles and the optimized motion profile

The final energy required by the motor of the self-balancing transporter to move 2 meters in 5 seconds in \( x \)-direction while commanded by each one of these eight motion profiles and the optimized motion profile is shown in Table 5-11. From the results shown in Table 5-11, the optimized motion profile required the lowest amount of energy when compared to these eight motion profile typically used in industrial applications. The optimized motion profile presented an energy cost at least 4.5% lower than any of these eight motion profiles. The cubic, sine-harmonic, and trapezoidal profiles, which are types of motion to profiles to be avoided in industrial applications, had energy cost 4.5%, 5.2%, and 5.4% higher than the optimized profile. These motion profiles are highlighted in Table 5-11. When, the energy cost with the optimized motion profile is compared to the
energy cost with the other profiles shown in Table 5-11, which are the types of motion profiles recommended for industrial applications, the energy savings with the optimized profiles was at least 7.4%.

Table 5-11 - Comparison of energy consumption for various types of motion profiles applied to the self-balancing transporter to move 2m in 5 seconds in simulation

<table>
<thead>
<tr>
<th>Motion Profile</th>
<th>Final Energy (J)</th>
<th>Variation (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized</td>
<td>4.4012</td>
<td>0</td>
</tr>
<tr>
<td>Trapezoidal</td>
<td>4.6396</td>
<td>5.4</td>
</tr>
<tr>
<td>Cycloidal</td>
<td>4.8067</td>
<td>9.2</td>
</tr>
<tr>
<td>ModSine</td>
<td>4.7263</td>
<td>7.4</td>
</tr>
<tr>
<td>Cubic</td>
<td>4.5985</td>
<td>4.5</td>
</tr>
<tr>
<td>Sine-Harmonic</td>
<td>4.6281</td>
<td>5.2</td>
</tr>
<tr>
<td>5th Order Polynomial</td>
<td>4.7667</td>
<td>8.3</td>
</tr>
<tr>
<td>7th Order Polynomial</td>
<td>4.8732</td>
<td>10.7</td>
</tr>
<tr>
<td>9th Order Polynomial</td>
<td>4.9476</td>
<td>12.4</td>
</tr>
</tbody>
</table>

5.3.5 Experimental Results

The self-balancing transporter was built as described in Section 4.3 for the experimental results.

As in simulation, the self-balancing transporter was commanded to perform a 2 meter in the x-direction in 5 seconds. The command signal $x(t)$ was designed by each one of the eight motion profiles typically used in industrial applications and defined in (1), (2), (3), (4), (5), (6), (7), (8), (9), and (10). The energy required to perform this move with each one of these motion profiles was experimentally measured and compared to the energy required for the same move while the self-balancing transporter was commanded by the optimized motion profile shown in Fig. 5-32. The energy required by the motors of the self-balancing transporter while commanded by each one of these motion profiles was
compared. This comparison was used to validate the efficiency of the proposed iterative energy optimization method in generating a motion profile that has lower energy cost than other typical motion profiles.

The position command $x$ generated by these eight motion profiles and the optimized motion profile was implemented in the subsystem called “Psn Cmd” in Fig. 5-24.

The voltage and current of each motor was acquired during tests, and the resulting total energy $E_t$ defined in (150) was calculated while each motion profile was experimentally tested. The results for $E_t$ are shown in Fig. 5-35. As in simulation, the use of optimized motion profile in Fig. 5-35 to command the motion of the self-balancing transporter in $x$-direction yielded significant energy saving in comparison to the use of the other motion profiles typically used in industrial applications. In these experimental results, the optimized motion profile required 5.6% less energy than the sine-harmonic profile which was the profile with the lowest energy cost from the eight typical motion profile tested. Thus, the optimized motion profile saved at least 5.6% energy in comparison to all the other motion profiles tested in this experiment as shown in Table 5-12. Meanwhile, the energy of the self-balancing transporter commanded by optimized motion profile was at least 12.2% lower than its energy consumption while commanded by the recommended motion profiles (cycloidal, modsine, $5^{th}$ order polynomial, $7^{th}$ order polynomial, $9^{th}$ order polynomial). This validates the ability of the proposed iterative energy optimization method in computing a motion profile that allow to minimize the energy consumption of the self-balancing transporter by simply modifying the motion profile without any mechanical changes.
Fig. 5-35 - Energy consumption by the optimized motion profile and eight others for the self-balancing transporter

Table 5-12 - Comparison of energy consumption for various types of motion profiles applied to the physical self-balancing transporter to move 2m in 5 seconds in x-direction.

<table>
<thead>
<tr>
<th>Motion Profile</th>
<th>Final Energy (J)</th>
<th>Variation (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized</td>
<td>3.6760</td>
<td>0</td>
</tr>
<tr>
<td>Trapezoidal</td>
<td>4.2800</td>
<td>16.4</td>
</tr>
<tr>
<td>Cycloidal</td>
<td>4.5100</td>
<td>22.7</td>
</tr>
<tr>
<td>ModSine</td>
<td>4.1230</td>
<td>12.2</td>
</tr>
<tr>
<td>Cubic</td>
<td>4.1620</td>
<td>13.2</td>
</tr>
<tr>
<td>Sine-Harmonic</td>
<td>3.8810</td>
<td>5.6</td>
</tr>
<tr>
<td>5(^{th}) Order Polynomial</td>
<td>4.6850</td>
<td>27.4</td>
</tr>
<tr>
<td>7(^{th}) Order Polynomial</td>
<td>4.6910</td>
<td>27.6</td>
</tr>
<tr>
<td>9(^{th}) Order Polynomial</td>
<td>4.8370</td>
<td>31.6</td>
</tr>
</tbody>
</table>
5.3.6 Discussion of Results

In Section 5.3, an iterative energy optimization method was investigated to minimize the electrical energy consumption of a self-balancing transporter. This method was validated by comparing the energy cost of the system commanded by the optimized profile to the energy cost of the system commanded by eight motion profiles typically used in industrial applications. Simulations and experiments presented significant energy reduction with the optimized motion profile in comparison to these eight motion profiles. When the energy consumption of the system commanded by the optimized motion profile was compared to the energy of the system commanded by the motion profiles to be avoided in industrial applications (trapezoidal, cubic, and sine-harmonic), the energy savings was between 4.5% and 5.4% from simulations, and 5.6% and 16.4% from the experimental results. When the energy consumption of the system commanded by the optimized motion profile was compared to the energy of the system commanded by the recommended motion profiles (cycloidal, modsine, 5th order polynomial, 7th order polynomial, and 9th order polynomial), the energy savings was between 7.4% and 12.4% from simulations, and 12.2% and 31.6% from the experimental tests.

Since the self-balancing transporter is a slow dynamic system, the energy optimization method may not yield the most energy savings for fast motion profiles due to the inability to follow the command motion profile with low position-following error. Thus, systems as the self-balancing transporter may not take full advantage of the energy savings that could be provided with optimized motion profiles. Consequently, the energy consumption of the system commanded by the optimized motion profiles may be higher.
that what the optimized profile would provide if the system could follow the command profile $x$ with low position-following error.
Chapter 6

Conclusions

The conclusions and suggested future work of this research are now presented.

6.1 Conclusions

The mechatronic design process and energy optimization methods presented in this dissertation can be of great benefit for the industrial sector. The mechatronic design process provides a methodology to properly select motors and systematically design the motion profile that controls each motor. The developed trajectory planning method avoids undesirable effects such as vibration, noise, and stress in mechanical and electronic components. A machine, that has the motor sized through this mechatronic design process and has the motion profiles designed with the developed trajectory planning method, has the potential to achieve a very high performance level, once tuned as described in Section 3.1.11 or through other tuning methods that lead to high performance levels. This mechatronic design process can be applied to single-axis machines, multi-axis machines, or multi-axis coordinated systems (e.g., robots).

Although, this method was demonstrated for three systems (two-inertia system, Cartesian two-axis parallel robot – H-Bot, and a self-balancing transporter), it is a generic method that can be used with any industrial machine.

In motion-control applications, there is a range of motor sizes that yield a proper solution for a given servo axis. Small-frame-size motors may not have enough torque to
power the axis, while large-frame-size motors may require too much acceleration torque to drive the rotor inertia, and the remaining torque is too low to power the servo axis. The proposed mechatronic design method provides a systematic approach to identify the range of motors to best power the system.

The energy optimization methods presented in this dissertation can be used to minimize the energy consumption of industrial machines. Three systems were used to validate these methods. The energy optimization method demonstrated with the two-inertia system yielded energy savings of 13.4% through simulations, and 15.5% through experiments when compared to the energy consumption of the system while commanded by recommended motion profiles typically used in industrial applications (cycloidal, modsine, 5th-order polynomial, 7th-order polynomial, and 9th-order polynomial). The energy optimization method developed for multi-axis coordinate systems and demonstrated with the H-Bot yielded energy savings of at least 39.3% from the simulations and 13.5% from the experimental results in comparison to the energy consumption of the system while commanded by position reference signals designed with the recommended motion profiles. The differences in energy savings between simulation and experimental results are attributed to the simplification in the model of the H-Bot.

Meanwhile, the method demonstrated with the self-balancing transporter saved energy by 8.3% from the simulations and 12.2% from the experimental results in comparison to the energy cost with the command signal designed with the recommended motion profiles. Therefore, the proposed energy optimization methods provided energy cost reduction of at least 12% from the experimental tests for any of the systems tested in this dissertation.
when compared to the energy cost with the systems commanded by position reference
signals designed with typical motion profiles recommended for industrial applications.

In comparison to any of the eight motion profiles tested in this dissertation, which
includes the motion profiles to be avoided in industrial applications (trapezoidal, cubic,
and sine-harmonic), the energy optimization method tested with the H-Bot saved at least
23.1% of energy via simulations and 11.6% via experimental tests. Meanwhile, the
energy optimization method tested with the self-balancing transporter saved at least 4.5%
of energy via the simulations and 5.6% via experimental results.

The efficiency of the optimized motion profile in minimizing energy cost depends
on tuning. As the position-following error is reduced, the ability to reduce the energy cost
is also reduced. This is due to the fact that with high position-following error, the actual
motion profile that the load executes does not match the optimized motion profile, and
the actual profile at the load may not retain the characteristics that enable energy
minimization. Thus, a well-tuned system is important to achieve higher energy savings.

It was also observed that as compliance in the system increases, the ability to
save more energy in comparison with the eight motion profiles tested in this dissertation
also increases. Although, compliance is not always desirable in industrial systems, it is
always present, and the amount of energy savings depends on the level of compliance.

The level of electrical energy savings that can be achieved with industrial
machines as demonstrated in this dissertation is of great importance to reduce production
costs with minimum impact to the machine design. The only change is in the design of
the motion profile commanding each servo axis. This method only modifies the profile of
the command signal without altering the timing for each segment of the original motion
profile. This facilitates the implementation of the optimized motion profile since it does not change the time for each step of the machine. Additionally, no changes are necessary in the mechanical or automation system. This also helps to reduce the cost of implementation of this method in industrial machines.

6.2 Future Work

For the mechatronics design process presented in this dissertation, the thermal model of the motor and the IGBT’s in the servo drive powering the motor can be included in the analysis for the search of the proper motor to power the system.

For the energy optimization method, the analysis of the dc-bus in the drive can be included in the analysis to optimize the energy also in terms of the regenerative energy recovered in the capacitors in the servo drive.

The feasibility of applying this energy optimization method for on-line optimization instead of off-line could also be investigated. This would allow one to re-compute motion profiles during the machine process for the case in which the command signal needs to be modified to account for changing processes, machine variability, different products, etc.

The energy optimization method was validated for three systems, but to make this system robust enough and generic enough to apply for any industrial machine, exhaustive tests are still required.


[27] "VDI 2221 Systematic approach to the development and design of technical systems and products," ed: The Association of German Engineers (VDI), 1993, p. 44.


[75] S. Parasuraman, H. Chiew Mun, and F. Sai Cheong, "Trajectory planning for redundant manipulator using evolutionary computation technique," in *IEEE*
273


[107] "±1.5g, ±6g Three Axis Low-g Micromachined Accelerometer," ed: Freescale Semiconductor, 2008.


Appendix A - Arduino code to decodify encoders

The code shown below measures the rising edge of either channel A or B of an incremental via Interrupts of the Arduino Mega board. Since only one edge or only on channel of each encoder triggers a count in position in this code, the resolution of the decodification matches the encoder pulses per revolution (PPR) specification. For example, for a 500 PPR encoder, this code will measure 500 pulses per motor revolution. This code can be modified to measure either the falling or rising edges of both channels A and B which doubles the resolution of position measurement, or it can be modified to measure the falling and rising edges of channels A and B which quadruples the resolution of position measurement. When both edges or both channels are measured, the decodification method is in general called “Quadrature” or “AQuadB”. If a geared-motor is used, the gear ratio can be entered in this code and the measured angular position will represent the angular position of the output of the gearbox. If the encoder resolution is too high, it can be reduced by the parameter “Ratio” in this code. The measured angular position is sent to the digital output at every loop in the code by directly writing to the digital outputs.

This code to decodify two encoders can be copied and pasted in the Arduino Software [98] and downloaded to an Arduino Mega board. The electrical diagram to wire the encoders to the Arduino board is shown in Fig. 4-13.

```cpp
// ====================================================================================== // Single Edge Decodification for Two Incremental Encoders // Description: // This code can measure angular position and angular velocity of two incremental encoders. Single Edge mode means that rising or falling edges of channel A of each encoder is watched via interrupts of Arduino. Thus, a 500 PPR encoder for example, yield 500 counts per revolution, which matches the PPR of the encoder. ```
Encoder 1 connected to Pins 2 and 4 on Arduino Mega
Encoder 2 connected to Pins 3 and 5 on Arduino Mega
The code shown below can be used for more than 40000 encoder pulses per second per motor.

// Inputs:
// MotorMaxSpdRPS = this is the maximum motor speed in Rev/sec
// EncoderPPR = this is the Pulses per Revolution of the encoder.
// Outputs:
// Pos1 = it contains the unwinded angular position of the encoder 1. The value of
// Pos1 varies from 0 to 250. Since the encoder used in this
// experiment has 500 PPR, Pos1 unwinds every half encoder revolution.
// Pos2 = it contains the unwinded angular position of the encoder 2. The value of
// Pos1 varies from 0 to 250. Since the encoder used in this
// experiment has 500 PPR, Pos2 unwinds every half encoder revolution.
// By: Aderiano da Silva
// Marquette University
// Department of Mechanical Engineering
// Created: Nov 2011
// Updates:
// Sep 28, 2012: Added code to ouptut angular position of both encoders.
// Aug 16, 2013: created Pos1 and Pos2 variables and write value directly to
guage outputs.
// Dec 19, 2014: Added "Ratio" to decodification. Thus, lower resolution data is
sent to second Arduino.

#include "WProgram.h"  // This header file includes all the definitions
// needed for the standard Arduino core. This enables us to use the pinMode, digitalWrite, delay etc.
#include "Arduino.h"  // This header file includes all the definitions
// needed for the standard Arduino core. This enables us to use the pinMode, digitalWrite, delay etc.
#include <digitalWriteFast.h>  // library for high performance digital reads and

// Input Data
#define MotorMaxSpdRPS 35L  // maximum motor speed in Revolutions Per Second (RPS)
#define EncoderPPR 500L  // PPR (Pulses per revolution) of encoder. Also
called CPR (counts per revolution)
const float GB_ratio = 1;  // Gear Ratio of the gear box connected to the motor.
Enter 1 for no gear box case.

// Parameters to measure angular shaft position:
// Encoder 1
#define InterruptNumber_A1 0  // number of one out of two interrupt: interrupt 0 is
for Digital Input 2
#define PinNumberChannel_A1 2  // Digital Input pin number for interrupt 0: Digital
Input 2, for Channel A of encoder
#define PinNumberChannel_B1 4  // Digital Input pin number for interrupt 0: Digital
Input 3, for Channel B of encoder
#define VelocityPinOutEnc1 9  // Number of Analog Output with angular shaft
velocity in RPS converted to digital values (0-255). Avoid pins 5 and 6 due to higher
duty cycles.
#define PositionPinOut_1 6  // Number of Analog Output with angular shaft
position at output of gear box in Rev converted to digital values (0-255). Avoid pins 5 and 6 due to higher duty cycles.
volatile bool Channel_A1_Status;  // status of Channel A of the encoder. The status is
// checked when an interrupt occurs due to a transition in either A or B channels.
volatile bool Channel_B1_Status;  // status of Channel B of the encoder. The status is
// checked when an interrupt occurs due to a transition in either A or B channels.
volatile long AngPos_in_EncCnts_1 = 0;  // angular shaft position in encoder counts.
Thus, 1 revolution = EncoderPPR x 4

// Encoder 2
#define InterruptNumber_A2 1  // number of one out of two interrupt: interrupt 0 is
for Digital Input 2
#define PinNumberChannel_A2 3  // Digital Input pin number for interrupt 0: Digital
Input 2, for Channel A of encoder
```c
#define PinNumberChannel_B2 5       // Digital Input pin number for interrup 0: Digital
// Input 3, for Channel B of encoder
//#define VelocityPinOutEnc2 10     // Number of Analog Output with angular shaft
velocity in RPS converted to digital values (0-255). Avoid pins 5 and 6 due to higher
duty cycles.
//#define PositionPinOut_2 11       // Number of Analog Output with angular shaft
position at output of gear box in Rev converted to digital values (0-255). Avoid pins 5
and 6 due to higher duty cycles.
volatile bool Channel_A2_Status;    // status of Channel A of the encoder. The status is
checked when an interrupt occurs due to a transition in either A or B channels.
volatile bool Channel_B2_Status;    // status of Channel B of the encoder. The status is
checked when an interrupt occurs due to a transition in either A or B channels.
volatile long AngPos_in_EncCnts_2 = 0;  // angular shaft position in encoder counts.
Thus, 1 revolution = EncoderPPR x 4

// Parameters to calculate angular position and velocity of Encoder 1 and 2
signed long NewTime;                // current clock time in microseconds to calculate
velocity
signed long OldTime;                // clock time in microseconds to calculate "dt" for
velocity (dP/dt)
signed long NewPos_1;               // current angular position in encoder counts to
calculate velocity
signed long OldPos_1;               // angular position in encoder counts to calculate
"dp" for velocity (dP/dt)
signed long NewPos_2;               // current angular position in encoder counts to
calculate velocity
signed long OldPos_2;               // angular position in encoder counts to calculate
"dp" for velocity (dP/dt)
signed long NewVelCntsPerSec_1;      // calculated angular shaft velocity (dP/dt) in
encoder counts per second
signed long NewVelCntsPerSec_2;      // calculated angular shaft velocity (dP/dt) in
encoder counts per second
signed long MotorMaxVelCntsPerSec;  // maximum motor shaft velocity in encoder counts per
second
signed long VelAnalogOut_1;         // calculated angular shaft velocity (dP/dt) in
digital value (0 to 255), where 0 = min neg velocity, 127 = zero velocity, 255 = max pos
velocity
signed long VelAnalogOut_2;         // calculated angular shaft velocity (dP/dt) in
digital value (0 to 255), where 0 = min neg velocity, 127 = zero velocity, 255 = max pos
velocity
signed long PosAnalogOut_1;         // calculated angular shaft position in digital value
(0 to 255), where 0 = zero position, 255 = 1 rev at output of gear box
signed long PosAnalogOut_2;         // calculated angular shaft position in digital value
(0 to 255), where 0 = zero position, 255 = 1 rev at output of gear box
signed long Pos1;                   // angular position of encoder 1 in binary from 0 to
2^PinCount
signed long Pos2;                   // angular position of encoder 2 in binary from 0 to
2^PinCount
int LowPinNum = 30;                 // pin number of the first pin to be sent the digital
angular position
to the digital outputs
int PinCount = 12;                   // nummer of bits on digital angular position sent
to the digital outputs
int Pos1Mult;                       // used to unwind the encoder 1 position
int Pos2Mult;                       // used to unwind the encoder 1 position
int Ratio = 4;                       // this is the ratio btw full resolution of
decodification and the
// value sent to the digital outputs, Examples:
// Ratio = 1 --> 1000/1 = 1000 encoder counts per
motor rev
// Ratio = 4 --> 1000/4 = 250 encoder counts per
motor rev
// Ratio = 8 --> 1000/8 = 125 encoder counts per
motor rev
int Ratio250 = Ratio*250;

// The setup() function is called when a sketch starts. Use it to initialize variables,
pin modes, start using libraries, etc.
// The setup function will only run once, after each powerup or reset of the Arduino
board.
void setup()
{

```
Serial.begin(115200);
OldTime=micros();                                         // initialize
variable OldTime
OldPos_1=0;                                              // initialize
variable OldPos_1 for Encoder 1
OldPos_2=0;                                              // initialize
variable OldPos_2 for Encoder 2
MotorMaxVelCntsPerSec = MotorMaxSpdRPS*EncoderPPR*4L;    // calculate
maximum motor angular velocity in encoder counts per second.

// The suffix "L"
force
 // Encoder 1
pinMode(PinNumberChannel_A1, INPUT);                     // sets as input
the pin for channel A signal from the encoder
digitalWrite(PinNumberChannel_A1, LOW);                  // turn on pullup
resistors for channel A signal from the encoder
pinMode(PinNumberChannel_B1, INPUT);                     // sets as input
digitalWrite(PinNumberChannel_B1, LOW);                  // turn on pullup
resistors for channel A signal from the encoder
attachInterrupt(InterruptNumber_A1, HandleMotorInterrupt_1, RISING); // sets interrupt
to watch for rising and falling edges in Pin 2 (Channel A). Call function
HandleMotorInterruptA when transition happens.

// Encoder 2
pinMode(PinNumberChannel_A2, INPUT);                     // sets as input
digitalWrite(PinNumberChannel_A2, LOW);                  // turn on pullup
resistors for channel A signal from the encoder
pinMode(PinNumberChannel_B2, INPUT);                     // sets as input
digitalWrite(PinNumberChannel_B2, LOW);                  // turn on pullup
resistors for channel A signal from the encoder
attachInterrupt(InterruptNumber_A2, HandleMotorInterrupt_2, RISING); // sets interrupt
to watch for rising and falling edges in Pin 3 (Channel B). Call function
HandleMotorInterruptB when transition happens.

// Set as OUTPUTS all Digital Outputs used for encoder position
for (int i = 0; i < PinCount * 2; i++) {
    pinMode(LowPinNum+i, OUTPUT);
}

// Calculation of angular position in encoder counts
// --------------------------------------------------------------------------------
// Interrupt service routines when a transition in Pin 2 - Channel A of the encoder 1 - occurs, and then calculate new angular position
void HandleMotorInterrupt_1()
{
    Channel_B1_Status = digitalReadFast(PinNumberChannel_B1); // fast read of the status
    Channel_A1_Status = digitalReadFast(PinNumberChannel_A1); // fast read of the status
    if (Channel_B1_Status != Channel_A1_Status) // if A!=B with transition
        AngPos_in_EncCnts_1++;
    else                                        // if A==B with transition
        AngPos_in_EncCnts_1--;
}

// Interrupt service routines when a transition in Pin 3 - Channel A of the encoder 2 - occurs, and then calculate new angular position
void HandleMotorInterrupt_2()
{
    Channel_B2_Status = digitalReadFast(PinNumberChannel_B2); // fast read of the status
}
Channel_A2_Status = digitalReadFast(PinNumberChannel_A2); // fast read of the status of the digital input pin 3

if (Channel_B2_Status == Channel_A2_Status) // if A==B with transition
    AngPos_in_EncCnts_2++; // A-- Positive Direction, then increment counts
else // if A!=B with transition
    AngPos_in_EncCnts_2--; // A-- Negative Direction, then decrement counts
}

void loop()
{
    // --------------------------------------------------------------------------------
    // Unwind Angular Position of Encoder 1 to binary
    // --------------------------------------------------------------------------------
    // Calculate Angular Shaft Position of Encoder 1:255
    PosAnalogOut_1 = AngPos_in_EncCnts_1*1L;

    // Unwinds the encoder position from 0 to 255 in any direction:
    if (PosAnalogOut_1 < 0) {
        Pos1Mult = -PosAnalogOut_1/Ratio250;
        Pos1 = Ratio250 + PosAnalogOut_1 + Pos1Mult*Ratio250;
        Pos1 = Pos1 / Ratio;
    }
    if (PosAnalogOut_1 >= 0) {
        Pos1Mult = PosAnalogOut_1/Ratio250;
        Pos1 = PosAnalogOut_1 - Pos1Mult*Ratio250;
        Pos1 = Pos1 / Ratio;
    }

    // --------------------------------------------------------------------------------
    // Send angular position of Encoder 1 to digital outputs:
    // --------------------------------------------------------------------------------
    PORTC = Pos1 & B11111111; // see: http://arduino.cc/en/Reference/BitwiseAnd

    // --------------------------------------------------------------------------------
    // Unwind Angular Position of Encoder 2 to binary
    // --------------------------------------------------------------------------------
    // Calculate Angular Shaft Position of Encoder 2
    PosAnalogOut_2 = AngPos_in_EncCnts_2*1L;

    // Unwinds the encoder position from 0 to 250 in any direction:
    if (PosAnalogOut_2 < 0) {
        Pos2Mult = -PosAnalogOut_2/Ratio250;
        Pos2 = Ratio250 + PosAnalogOut_2 + Pos2Mult*Ratio250;
        Pos2 = Pos2 / Ratio;
    }
    if (PosAnalogOut_2 >= 0) {
        Pos2Mult = PosAnalogOut_2/Ratio250;
        Pos2 = PosAnalogOut_2 - Pos2Mult*Ratio250;
        Pos2 = Pos2 / Ratio;
    }

    // --------------------------------------------------------------------------------
    // Send angular position of Encoder 2 to digital outputs:
    // --------------------------------------------------------------------------------
    PORTL = Pos2 & B11111111; // see: http://arduino.cc/en/Reference/BitwiseAnd

delay(1); // this is the loop time given in milliseconds.

    // WriteSerial(); // ADD ANY CODE HERE!
/ OR YOU CAN CREATE A NEW FUNCTION AND ADD YOUR CODE THERE. FOR EXAMPLE, CREATE THE
FUNCTION MyCode():
  // MyCode();
}

/*
void WriteSerial()
{
  Serial.print(Pos1 DEC);
  Serial.print(" ");
  delay(500);
} */
Appendix B – Arduino code to read angular position in the control board

The sub-assembly in Simulink to read the angular position of motor 1 from the digital inputs is shown in Fig. B-1.

Fig. B-1 - Sub-assembly in Simulink to read the angular position of motor 1 from the digital inputs

The content of the sub-assembly shown in Fig. B-1 is shown in Fig. B-2.

Fig. B-2 - Content of the sub-assembly used to read the angular position of motor 1 from the digital inputs
The content of the block “Convert Encoder Position from Cyclic to Linear” is shown in Fig. B-3.

The code of the “MATLAB Function - Convert Encoder Position from Cyclic to Linear” is shown below:

```matlab
function [ActualVel,UnlimitedActualPos] = EncUnwind(CyclicPosAtual,CyclicPosPrev,UnlimitedPosPrev,CntsPerCycle,Ts)
%#codegen
% Convert cyclic angular position of an encoder into a linear position
% Input variables:
%   CyclicPosAtual: actual reading from encoder (this is an integer number
%                  from zero to "CntsPerCycle")
%                  [Encoder Counts]
%   CyclicPosPrev:  encoder position from the previous scan (this is an
%                  integer number from zero to "CntsPerCycle")
%                  [Encoder Counts]
%   LinearPosPrev:  converted encoder position from cyclic to linear, but from
%                  the previous scan (this is a number from -inf to +inf)
%                  [Encoder Counts]
%   CntsPerCycle:   number of encoder counts per revolution. For a 500 PPR
%                  encoder with X2 decodification, the CntsPerCycle is:
%                  500 Pulses/Rev x 2 Counts/Pulse = 1000 Counts/Rev
%                  [Counts/Rev]
% Output variables
%   UnlimitedPosActual: converted encoder position from cyclic to unlimited
% (this is a number from -inf to +inf)
%   ActualVel: calculated angular velocity [Encoder Counts/sec]
%
% Convert cyclic angular position in unlimited angular position:
if abs(CyclicPosAtual - CyclicPosPrev) > 0.5 * CntsPerCycle % detects unwind condition
  % A unwind condition occurred, then calculate LinearPosActual based on CW or CCW rotation:
  if CyclicPosAtual < CyclicPosPrev % CW rotation (0 ...CntsPerCycle)
```
UnlimitedActualPos = UnlimitedPosPrev + (CntsPerCycle - CyclicPosPrev) + CyclicPosAtual;
else
    UnlimitedActualPos = UnlimitedPosPrev - (CntsPerCycle - CyclicPosAtual) - CyclicPosPrev;
end
else
    UnlimitedActualPos = UnlimitedPosPrev + (CyclicPosAtual - CyclicPosPrev);
end
% Calculate angular velocity:
ActualVel = (UnlimitedActualPos - UnlimitedPosPrev)/Ts;
Appendix C – Sensor Fusion Derivation

The sensor fusion is given in Fig. 4-23 and introduced in Section 4.3.2.4. The derivation of equation (57) is given next.

A low-pass filter is defined as follows:

$$LPF = \frac{y}{x} = \frac{1}{\tau s + 1}$$  \hspace{1cm} (155)

Where, \(\tau\) is the time constant of the filter. By manipulation of the fundamental discrete-time equation of a LPF, \(x_l - y_l = \tau(y_l - y_{l-1})/dt\), the LPF equation can be written as follows:

$$y_l = x_l(1 - \beta) + y_{l-1}\beta$$  \hspace{1cm} (156)

Where:

$$\beta = \frac{\tau}{\tau + dt}$$  \hspace{1cm} (157)

Where \(dt\) is the sampling time of the signals and \(\tau\) is time constant that defines the LPF response.

Meanwhile, the a high-pass filter is defined as:

$$HPF = \frac{y}{x} = \frac{\tau s}{\tau s + 1}$$  \hspace{1cm} (158)

Similarly, by manipulation of the fundamental discrete-time equation of a HPF, \(y_l = \tau[((x_l - x_{l-1})/dt) - ((y_l - y_{l-1})/dt)]\), the HPF equation can be written as follows:

$$y_l = (x_l - x_{l-1})\beta + y_{l-1}\beta$$  \hspace{1cm} (159)
The LPF equation (156) can then be defined in terms of the accelerometer signal as follows:

$$\theta_{AF} = \theta_{A}(1 - \beta) + \theta_{AF_{i-1}}$$  \hspace{1cm} (160)$$

Similarly, HPF equation can be defined in terms of the gyroscope signal as follows:

$$\theta_{GF} = (\theta_{G_{i}} - \theta_{G_{i-1}}) \beta + \theta_{GF_{i-1}}$$  \hspace{1cm} (161)$$

The tilt angle is the sum of the angle measured by the accelerometer and the gyroscope. Therefore, by combining both signal from the accelerometer, an accurate and clean measurement of the tilt angle is possible. The tilt angle is then calculated as follows:

$$\theta_{i} = \theta_{GF_{i}} + \theta_{AF_{i}}$$  \hspace{1cm} (162)$$

Substituting (160) and (161) into (162), the following is obtained:

$$\theta_{i} = (\theta_{G_{i}} - \theta_{G_{i-1}}) \beta + \theta_{GF_{i-1}} + \theta_{A}(1 - \beta) + \theta_{AF_{i-1}}$$  \hspace{1cm} (163)$$

Since:

$$\theta_{i-1} = \theta_{GF_{i-1}} + \theta_{AF_{i-1}}$$  \hspace{1cm} (164)$$

Then, substituting (164) into (163):

$$\theta_{i} = (\theta_{G_{i}} - \theta_{G_{i-1}}) \beta + \theta_{i-1} \beta + \theta_{A}(1 - \beta)$$  \hspace{1cm} (165)$$

Which yields, the discretized sensor fusion equation:

$$\theta_{i} = (\theta_{G_{i}} - \theta_{G_{i-1}}) \beta + \theta_{i-1} \beta + \theta_{A}(1 - \beta)$$  \hspace{1cm} (166)$$
Appendix D – Matlab code for two-inertia system

This code is to generate the results shown in Fig. 5-2, Fig. 5-3, Fig. 5-4, and Fig. 5-5.

clear all; close all; clc

% INPUT DATA
T = 0.1; % machine cycle time [sec]
ThetaM_T = 0.1*(2*pi); % desired final position for motor [rad]
ThetaL_T = 0.1*(2*pi); % desired final position for load [rad]
Ts = -1; % used for Simulink model

% MECHANICAL DATA
Kt = 1.58/1.7; % torque constant [Nm/A] calculated as stall torque / stall rms current
Imax = 2.4/sqrt(2); % maximum motor current (peak current) [A]
Vmax = 460; % maximum motor voltage [V]
Rm = 18.9; % Motor resistance [ohms]
GR = 1; % gear ratio of gear box
Vm = 460; % motor voltage [V]
Ke = 98 * 60/(1000*2*pi); % Back EMF constant [V/rad/s]
Kb = Ke; % Back EMF constant [V/rad/s]
L = 92e-3; % inductance [H]
Jm = 0.000044 * (GR^2); % motor inertia reflected to output of gearbox [kg-m2]. NOTE: Gearbox inertia is unknown.
Tf = 0.068; % Friction torque (note: the motor used has a seal shft, so Tf is for seal motor shaft case
b = (0.016/1000/2/pi); % for MPLA310, b=0.016Nm/krpm and it needs to be converted to Nm-s/rad

% Torsional stiffness of motor shaft and rod connecting (btw load and coupling)
k = 1281; % [Nm/rad]

% Fly-wheel (1 load)
Dens = 7900; % material density [kg/m^3]. Steel=7900,
OD_L = 0.127; % outer diameter of fly-wheel [m]
ID_L = OD; % inner diameter of fly-wheel [m]
L_L = 0.0125; % length of fly-wheel [m]
Vo = pi*(OD_L/2)^2*L_L; % volume [m^3]
Vi = pi*(ID_L/2)^2*L_L; % volume [m^3]
JL = 0.5*Dens*Vo*(OD_L/2)^2 - 0.5*Dens*Vi*(ID_L/2)^2; % Load inerital [kg/m2]

% ENERGY OPTIMIZATION METHOD

% Hamiltonian matrix using cost function as int((va/R)-(va*Kb*omega/R))
H = [0 1 0 0 0 0 0 0 0 0 ;
    -k/Jm -b/Jm k/Jm b/Jm k/Jm 0 0 0 0 0 ;
    0 0 1 0 0 0 0 0 0 0 ;
    k/JL b/JL -k/JL -b/JL 0 0 0 0 0 0 ;
    0 -v*2*L) 0 0 -Rm/L 0 0 0 0 -Rm/(2*L^2) ;
    0 0 0 0 0 k/Jm 0 -k/JL 0 -k/JL 0 ;
    0 v*2/L) 0 0 0 0 -b/Jm 0 b/JL 0 0 ;
    0 0 0 0 0 0 -k/Jm 0 0 -Kt/Jm 0 0 ];

% Boundary Conditions
x0 = [0 0 0 0 0]';
xT = [ThetaM_T 0 ThetaL_T 0 0]';

%========== Calculation of Transition Matrix and control effort =========

% Calculating lambda 0:
t=0:1/20000:T; % Note: t needs to be defined this way instead of linspace for the Simulink model match the Matlab results
A = H;
PhiT=expm(A*T);
PhiT11=PhiT(1:5,1:5);
PhiT12=PhiT(1:5,6:10);
PhiT22=PhiT(6:10,6:10);

% From equations (13) and (11):
lambda0 = PhiT12\(xT - PhiT11*x0);

% Calculating the dynamic response of the system:
B = zeros(10,1);
C = [0 0.5*Ke 0 0 0 0 0 0 0 -Rm/(2*L)];
D = 0;
u = 0*t;
[ustar,X]=lsim(A,B,C,D,u,t,[x0;lambda0]);

% PLOTS
figure; plot(t,X(:,1:5), t,ustar,'LineWidth',2)
xlabel('Time (sec)')
ylabel('Angular Position (rad)')
grid on
legend('Mtr Pos(t)', 'Mtr Vel(t)', 'Load Pos(t)', 'Load Vel(t)', 'Ia(t)', 'u(t)')

figure
plot(t,X(:,1),t,X(:,3), 'LineWidth',2)
xlabel('Time (sec)')
ylabel('Angular Position (rad)')
grid on
legend('$\theta_m(t)$', '$\theta_l(t)$')

figure
plot(t,X(:,2), 'LineWidth',2)
hold on
plot(t,X(:,4), 'LineWidth',2)
xlabel('Time (sec)')
ylabel('Angular Velocity (rad/s)')
grid on
legend('$\omega_m(t)$', '$\omega_l(t)$')

figure
plot(t,ustar, 'LineWidth',2)
xlabel('Time (sec)')
ylabel('Motor Voltage (V)')
grid on

The code below is to generate the results shown in Fig. 5-7 and Fig. 5-8 and also in Table 5-1 and Table 5-2. The Simulink model called below is shown in Fig. 5-6:

%% Plot energy curves for all motion profiles
Selector=1;
sim('TwoJ_SystemComparison_V3_for_Fig_DualLoop')
EnergyAllMP.Optm=Energy;
FinalEnergy.Optm=Energy(length(Energy),2);
MaxPosError.Optm=max(PosError(:,2));

Selector=2;
sim('TwoJ_SystemComparison_V3_for_Fig_DualLoop')
EnergyAllMP.Trap=Energy;
FinalEnergy.Trap=Energy(length(Energy),2);
MaxPosError.Trap=max(PosError(:,2));

Selector=3;
sim('TwoJ_SystemComparison_V3_for_Fig_DualLoop')
EnergyAllMP.Cycloidal=Energy;
FinalEnergy.Cycloidal=Energy(length(Energy),2);
MaxPosError.Cycloidal=max(PosError(:,2));

Selector=4;
sim('TwoJ_SystemComparison_V3_for_Fig_DualLoop')
EnergyAllMP.ModSine=Energy;
FinalEnergy.ModSine=Energy(length(Energy),2);
MaxPosError.ModSine=max(PosError(:,2));

Selector=5;
sim('TwoJ_SystemComparison_V3_for_Fig_DualLoop')
EnergyAllMP.Cubic=Energy;
FinalEnergy.Cubic=Energy(length(Energy),2);
MaxPosError.Cubic=max(PosError(:,2));

Selector=6;
sim('TwoJ_SystemComparison_V3_for_Fig_DualLoop')
EnergyAllMP.SHM=Energy;
FinalEnergy.SHM=Energy(length(Energy),2);
MaxPosError.SHM=max(PosError(:,2));

Selector=7;
sim('TwoJ_SystemComparison_V3_for_Fig_DualLoop')
EnergyAllMP.Fifth=Energy;
FinalEnergy.Fifth=Energy(length(Energy),2);
MaxPosError.Fifth=max(PosError(:,2));

Selector=8;
sim('TwoJ_SystemComparison_V3_for_Fig_DualLoop')
EnergyAllMP.Seventh=Energy;
FinalEnergy.Seventh=Energy(length(Energy),2);
MaxPosError.Seventh=max(PosError(:,2));

Selector=9;
sim('TwoJ_SystemComparison_V3_for_Fig_DualLoop')
EnergyAllMP.Ninth=Energy;
FinalEnergy.Ninth=Energy(length(Energy),2);
MaxPosError.Ninth=max(PosError(:,2));

% Plot results
figure('Position',[500 500 700 500])
plot(EnergyAllMP.Optm(:,1),EnergyAllMP.Optm(:,2),'r')
hold on
plot(EnergyAllMP.Trap(:,1),EnergyAllMP.Trap(:,2),'k')
plot(EnergyAllMP.Cycloidal(:,1),EnergyAllMP.Cycloidal(:,2),'Color',[0 0.4 0])
plot(EnergyAllMP.ModSine(:,1),EnergyAllMP.ModSine(:,2),'Color',[0.682 0.467 0])
plot(EnergyAllMP.Cubic(:,1),EnergyAllMP.Cubic(:,2),'Color',[0.6 0 0.6])
plot(EnergyAllMP.SHM(:,1),EnergyAllMP.SHM(:,2),'g')
plot(EnergyAllMP.Fifth(:,1),EnergyAllMP.Fifth(:,2),'m')
plot(EnergyAllMP.Seventh(:,1),EnergyAllMP.Seventh(:,2),'b')
plot(EnergyAllMP.Ninth(:,1),EnergyAllMP.Ninth(:,2),'c')
xlabel('Time (sec)')
ylabel('Energy (J)')
legend1=legend('Optimized','Trapezoidal','Cycloidal','ModSine','Cubic','SHM','5't'h Order','7't'h Order','9't'h Order');
set(legend1,'Position',[0.1505 0.52968 0.184249 0.373124]);
Marquette University

This is to certify that we have examined this copy of the dissertation by

Aderiano M. da Silva, B.S., M.S.

and have found that it is complete and satisfactory in all respects.

The dissertation has been approved by:

____________________________________________
Dr. Kevin Craig
Dissertation Director, Department of Mechanical Engineering

____________________________________________
Dr. Mark L. Nagurka

____________________________________________
Dr. Philip A. Voglewede

____________________________________________
Dr. Anthony Bowman

____________________________________________
Dr. Ronald H. Brown

Approved on

_____________________________