To the Graduate Council:

I am submitting herewith a dissertation written by Renbin Zhou entitled “High Performance Control of a Transmission Based Servo Actuator System.” I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Mechanical Engineering.

William R. Hamel
Major Professor

We have read this dissertation
and recommend its acceptance:

Vijay S. Chellaboina

John Chiasson

Seddik M. Djouadi

Lynne E. Parker

Gary V. Smith

Acceptance for the Council:

Anne Mayhew
Vice Chancellor and
Dean of Graduate Studies

(Original signatures are on file with official student records.)

High Performance Control of a Transmission Based Servo Actuator System

A Dissertation

Presented for the Doctor of Philosophy Degree

The University of Tennessee, Knoxville

Renbin Zhou

May, 2006

Copyright © 2006, by Renbin Zhou

All rights reserved

Acknowledgement

I would like to thank Dr. William R. Hamel for his support and guidance throughout my graduate study in the Robotics and Electromechanical Systems Laboratory of the Department of Aerospace, Mechanical, and Biomedical Engineering at the University of Tennessee, Knoxville. I would like to thank him for his help in rewrite most of the dissertation draft, it is a rewarding experience which I will benefit from for the years ahead.

I would like to thank our TBA team, of which I was a member. It is this team's work that built the TBA prototype, on which my dissertation research is developed. The TBA team includes Dr. William Hamel, Dr. Arnold Lumsdaine, Dr. Spivey Douglass, Dr. Sewoong Kim, Kelley P. Brown, Sriram Sridharan, Kalyana Ganti, and many others.

I would like to thank Heather Humphreys for her kind help to edit the first draft of the dissertation. I would like to thank Ge Zhang for the technical discussions during my initial transition to real time Linux world. I would like to thank Dr. Reid Kress for his help during my initial transition to the university. I would also like to thank my committee members for their time and guidance in making this dissertation possible.

My greatest thanks go to my family: my parents, my wife and my daughter. It is their love that leads me through all the difficulties and makes all my effort meaningful, and here I humbly dedicate this work to them.

Abstract

High performance actuation is a key factor in the industrial robot area. The transmission based servo actuator system (TBA) is a new type of robot actuator with a brushless DC servo motor and a three speed discrete variable transmission (DVT). The proposed TBA design can match the performance of a typical hydraulic actuator with compact size and weight.

The TBA is a typical hybrid dynamic system consisting of three continuous dynamic systems and a discrete state controller. This dissertation addresses the fundamental problems associated with the TBA system control from a hybrid system point of view.

A detailed dynamic model of the TBA is developed. Due to the complexity of the TBA system, an exact model is unwieldy for control design and analysis purposes. In this research, the TBA system is simplified into a hybrid system with three second order linear time invariant systems, on which all the controls are developed.Dynamic stability of the TBA is critical for its function as a servoactuator. For a hybrid system, the stability problem has much broader range of issues than a purely continuous system.

In general, the plant stability and the subsystem stability are independent. For example, a hybrid system with stable subsystems can be unstable for certain switch sequences; on the other hand, a hybrid system with unstable subsystems can be stabilized by proper switch signals. In this dissertation, a sufficient condition is established for stability of the TBA system. It is proven that the hybrid system is stable under asynchronous switching if there exists a common

Lyapunov function for all subsystems. It is proven that the TBA subsystems can have a common Lypunov function by designing appropriate feedback controller. The feedback controller to stabilize the TBA can be transformed into a PID equivalent controller because the subsystems are second order linear time invariant systems (LTI). The PID controller was then implemented and high performance in terms of position error and transient suppression has been achieved. The discrete state controller should be stable, which means that its output should be consistent if the hybrid system is subjected to disturbances. A common phenomenon is that the state changes back and forth very frequently near the switch boundary, which is referred to as transition instability. This research proposes a switch strategy consisting of two boundaries to achieve the transition stability, and it is proved that the proposed switch strategy is transition stable.

An optimal controller is designed and difficulties associated with implementation are generated.

Based on the proposed control methods, a multithread real time control software has been developed to achieve a deterministic control loop sampling. The control software is developed in C/C++ under Real Time Application Interface (RTAI), which provides a real time programming environment in a normal Linux operating system.

With the proposed controller and a prototype TBA test system, TBA stability and control performance was demonstrated and evaluated. The following results were observed:

1. Steady state error of 0.005 degrees at the emulated robust manipulator shoulder pitch joint
2. Control loop sampling period of 1 millisecond with negligible delay
3. Transient disturbances associated with the gear shifting of ~20% in most cases.
4. The methods and applications used in this dissertation can be extended to a large range of hybrid dynamic systems in terms of control system design, analysis and implementation.

This research contributes to the literature and research knowledge base in the following ways:

1. Exploration and solution of the control problems of TBA's in the hybrid system control context.
2. Expansion of the fundamental understanding of the practical control issues of TBA's.
3. Analysis, design, and implementation of a real time TBA control system, and identification of the most suitable control strategy for the TBA.
4. The development of analysis and control methods that can be extended to a much broader range of hybrid dynamic systems.

TABLE OF CONTENTS

CHAPTER 1 BACKGROUND AND INTRODUCTION 1
HISTORY OF INDUSTRY ROBOT –ACTUATOR AND CONTROL 1
AN INTRODUCTION OF TBA – IDEAS, DESIGN, AND INTEGRATION 10
TBA Prototype and the Experimental System 19
SCOPE AND ORGANIZATION OF THE DISSERTATION 22
CHAPTER 2 MATHEMATICAL MODELING 24
BLDC MODEL 24
THREE SPEED DVT MODEL - A HYBRID DYNAMIC SYSTEM 29
A Single Planetary Gear Model 33
A Three Speed DVT Model 36
A ROBOT ARM MODEL 40
THE TBA PROTOTYPE MODEL 41
CHAPTER 3 TBA CONTROLS 45
BACKGROUND 45
SUPERVISORY CONTROL 54
Switch Strategy- Transition Stability 54
HYBRID SYSTEM STABILITY - TBA 65
CONTROLLER DESIGN AND IMPLEMENTATION 72
State Feedback Control 73
Optimal Control and Robust Control Discussion 85
SUMMARY 93
CHAPTER 4 CONTROL SYSTEM SOFTWARE 95
TBA DESIGN REQUIREMENT 96
OPERATING SYSTEM - INTRODUCTION 98
GPOS and RTOS 98
RTAI/Linux 105
TBA SOFTWARE DESIGN 109
A Windows/LabVIEW Implementation 110
Multithread Design with RTAI 111

A Graphical User Interface 113
CHAPTER 5 SIMULATIONS AND EXPERIMENTS 115
MATLAB/SIMULATION 115
BLDC Motor 115
DVT 117
Simulation Results 117
REAL TIME CONTROL EXPERIMENTAL RESULTS 122
Transition Unstable vs. Stable Switch Signals 123
Position Error 126
DVT Speed and Disturbances 126
Real Time Performance 130
Load Torques 130
Experimental Results Summary 131
CHAPTER 6 CONTRIBUTIONS AND FUTURE WORK 133
CONTRIBUTIONS 133
FUTURE WORK 135
LIST OF REFERENCES 137
APPENDIX 142
DVT MODEL 143
3rd gear derivation: (Brake #1, lowest ratio) 143
2nd gear derivation (brake #2, middle ratio) 147
1st gear derivation (brake #3, highest ratio) 151
A planetary gear with ring gear locked: 156
EXPERIMENTAL RESULTS 158
Zero load 158
25% load 160
50% load 162
75% load 164
TBA LOW LEVEL CONTROL SOFTWARE SOURCE CODE 166
tbacontrol.c 166
common.h 169

common.c 172
control.h 175
loadcell.c 182
motorcontrol.c 184
ni6023e.h 197
ni6023e.c 199
ni6711.h 205
ni6711.c 207
setmode.h 209
setmode.c 210
shiftcontrol.h 219
shiftcontrol.c 220
tbacontrol.c 223
TBAGUI 226
realtime.c 226
mainwidow.h 227
mainwindow.cpp 228
randomplot.h 231
mainwindow.cpp 233
incrementalplot.cpp 242
scrollbar.h 244
scrollbar.cpp 245
scrollzoomer.h 248
scrollzoomer.cpp 250
VITA 258

LIST OF TABLES

Table 1.1, Titan II performance specifications 11

Table 1.2, Load matching index for different TBA design 19

Table 5.1, Motor parameters 116

Table 5.2, DVT parameters and motor-DVT combinations 117

Table 5.3, Test combinations 118

Table 5.4, Simulation results (Full load) 118

Table 5.5, simulation result (half load) 120

Table 5.6, Position errors for different loads 127

Table 5.7, Maximum tracking errors for different loads 127

Table 5.8, Real time performance and jitter results 130

LIST OF FIGURES

Figure 1.1, DC motor equivalent model 2

Figure 1.2, DC motor closed loop position control 3

Figure 1.3, BLDC system setup 5

Figure 1.4, BLDC servo control diagram 5

Figure 1.5, AC servo control system diagram 7

Figure 1.6, AC, BLDC power-weight curve 8

Figure 1.7, AC, BLDC, DDR power-weight curve 9

Figure 1.8, Titan II equivalent model 12

Figure 1.9, Baseline torque speed curve 12

Figure 1.10, Typical BLDC torque speed curve 13

Figure 1.11, BLDC with two speed DVT torque speed curve 17

Figure 1.12, BLDC with three speed gear box torque speed curve 18

Figure 1.13, TBA system hardware layout 20

Figure 1.14, Three speed DVT 21

Figure 2.1, A functional BLDC construction diagram with four pole pairs 25

Figure 2.2, Trapezoidal back EMF 26

Figure 2.3, The gear shift system modeled as a finite state machine 32

Figure 2.4, A single planetary gear set 34

Figure 2.5, A prototype robot arm 41

Figure 3.1, A typical hybrid dynamic system control block diagram 47

Figure 3.2, Unstable hybrid system with stable dynamic subsystems 51

Figure 3.3, Two dimensional switch curve 59

Figure 3.4, Two dimensional switch curve with disturbance 61
Figure 3.5, A shift strategy with two shift boundaries 62
Figure 3.6, State feedback control diagram 72
Figure 3.7, Lyapunov function sets for zero load 75
Figure 3.8, Lyapunov function sets for maximum load 76
Figure 3.9, Equivalent control subsystem diagram 77
Figure 3.10, Top level control system diagram 78
Figure 3.11, Servo motor control diagram 79
Figure 3.12, Dynamometer torque control diagram 79
Figure 3.13, TBA Control System Diagram 80
Figure 3.14, TBA gear shift strategy (prototype) 81
Figure 3.15, TBA gear shift strategy (real system) 82
Figure 3.16, General TBA optimal control diagram 86
Figure 3.17, Second optimal control problem system diagram 87
Figure 3.18, TBA robust for disturbance attenuation 90
Figure 3.19, Open and closed loop magnitudes 91
Figure 3.20, Open and closed loop magnitudes 92
Figure 4.1, TBA subsystems interconnection relation 95
Figure 4.2, Standard Linux three layer model 99
Figure 4.3, GPOS task status 101
Figure 4.4, Process state diagram in an RTOS 102
Figure 4.5, RTAI/Linux system architecture 106
Figure 4.6, Windows/LabVIEW diagram 110

Figure 4.7, TBA real time control software implementation diagram 111

Figure 4.8, Screen shot of GUI 114

Figure 5.1, Top level Simulink model (Linear Model)..... 115

Figure 5.2, Three speed DVT with full load 119

Figure 5.3, Simulation results with new trajectory design 121

Figure 5.4, Single boundary, state based shift strategy (half load) 123

Figure 5.5, Single boundary, state based shift strategy (full load) 124

Figure 5.6, Two boundaries, hybrid switch signal (half load) 125

Figure 5.7, Two boundaries, hybrid switch signal (full load) 125

Figure 5.8, Position error with full load 127

Figure 5.9, BLDC speed with full load 128

Figure 5.10, DVT speed with full load 128

Figure 5.11, DVT speed disturbance near gear change from 2^{nd} gear to 1^{st} 129

Figure 5.12, BLDC servo motor and dynamometer load torque (full load)..... 132

LIST OF DEFINITIONS

Power index 14

Load region index 14

Load matching index 15

TBA gear numbers 33

Time-based switch signal [19] 55

State-based switch signal 55

Hybrid switch signal 56

Stabilities 66

RTOS 99

LIST OF SYMBOLS

$\omega_{\max 1}$	maximum velocity at the maximum load
$\omega_{\max 2}$	maximum velocity at the minimum load
J	motor moment of inertia
L, L_d, L_q	motor inductances
$i,\ i_d, i_q$	motor currents(average, d-axis, q-axis)
v, v_d, v_q	motor voltages(average, d-axis, q-axis)
k_e, k_t	motor back emf constant and torque constant
θ_R, ω_R	motor rotor angular position and velocity
b_i, b_m	coeffients of friction
f_s	static friction
i_{si}, u_{si}	motor phase currents and voltages
$T_{load}, T_{in}, T_{out}, T_b$	load, input, output, and brake torque
P_{titan}, P_r	baseline and required torques
$J_{si}, J_{ci}, J_{pi}, J_{ri}$	moments of inertia of sun, carrier, planetary, and ring gears
$\theta_{\text{si}}, \theta_{\text{ci}}, \theta_{\text{pi}}, \theta_{\text{ri}}$	angular positions of sun, carrier, planetary, and ring gears
δ_i	TBA states (i=1,2,3)
D	set of TBA states, D={δ_i (i=1,2,3)}
γ_i	inverse of TBA gear ratios (i=1,2,3)
T_{bi}	TBA brake torque (i=1,2,3)
J_{eqi}	TBA equivalent moment of inertia (i=1,2,3)
$r_{si}, r_{ci}, r_{pi}, r_{ri}$	radii of sun,carrier,planetory,ring gear (i=1,2,3)
λ_{k}	Lagrange multipliers (k=1,2,...,6)
K, V, F	kinetic, potential, and dissipation energy

CHAPTER 1

Background and Introduction

History of Industry Robot –Actuator and Control

Actuation of robot manipulators has been a driving factor in industrial robot research and applications; its impact can be found in both the expansion of the areas of application and in performance improvements. For example, if an electrically driven robot had not been invented, robot applications in the food industry and medical practice would have been unlikely. This advancement allowed both accuracy and repeatability to be achieved at levels as small as tens of micrometers.

The first industrial robot, a UNIMATE robot, was put into an assembly line by General Motors in 1961. It was purely hydraulic driven, and hydraulic servo-actuators were the most common actuator type in the decade that followed. The first UNIMATE had a successful life as a die-caster until it was retired after more than ten years of service, and it is now on display in the Smithsonian Institute in Washington.

Despite the success of hydraulic actuators in the early period, they have long been known to have problems like complexity, poor maintainability, low accuracy, and some environmental issues. Today, hydraulic actuator driven robots can only be found in application areas where high payloads or other

considerations must be addressed, such as undersea exploration, underground waste storage tanks, and deactivation and decommissioning (D&D) projects [1]. The Titan II and Titan III robot manipulators from Schilling Robotics are designed for these types of applications and are proven to be adequate and successful.

An electric motor driven robot does not have many of the problems of a hydraulic actuation. Around 1974, researchers started to build a pure electric motor driven robot, and fast growth occurred in this research area in the years that followed. Today, the electric motor driven robot prevails in almost all industrial robot applications.

Electric motors can be divided into two major categories: direct current (DC) and alternating current (AC) motors. The DC motor is characterized by an evenly distributed magnetic field in the air gap. It is known for its ease of use by simply supplying a DC voltage to the rotor windings. DC motors are widely used in robot design applications, especially small ones. Large DC motors are difficult to make; the challenges are management of the heat generated in the rotor windings and wearing of the mechanical brushes, etc. Like other electric motors, DC motor operation principles can be represented by an equivalent electrical circuit and a dynamic mechanical system as shown in Figure 1.1. The mathematical model of a DC motor is shown in Equation (1.1).

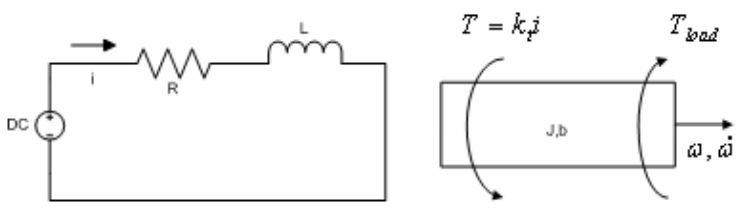


Figure 1.1, DC motor equivalent model

$$L\frac{di}{dt} = V - K_e\omega_R - iR$$
$$J\frac{d\omega_R}{dt} = K_t i - b_m\omega_R - f_s - T_{load} \qquad (1.1)$$

When DC motors are operated at steady state, the left sides of the equations are very small and can be treated as zero; thus, the relation between motor speed and supply voltage is linear if the load torque is constant, which makes it possible to use open loop speed control. Similarly, it is also possible to have open loop torque control if the motor speed is constant. But closed loop control is generally used in practice in order to achieve the desired servo performance.

Figure 1.2 shows a simple DC motor closed loop position control in voltage mode with a PID controller.

In order to achieve continuous rotation, some devices must be used to change the current direction in the rotor windings, which is called commutation. For example, for a two-pole DC motor with a permanent magnet stator and a wound rotor, the rotor current must commute every 180 degrees.

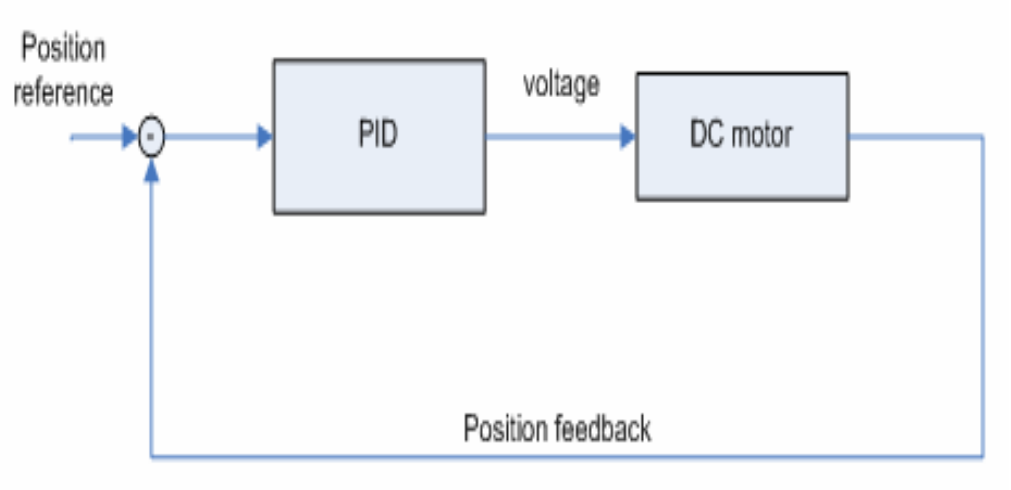


Figure 1.2, DC motor closed loop position control

Conventionally, the commutation is achieved using mechanical brushes, generally made from carbon material. The main problems associated with brushes are arcing and brush wear, which not only limits its application areas, but also increases maintenance costs. By turning the conventional DC motor inside out, and using a wound stator and a permanent rotor, researchers created another version of the electric motor. It is known as a Brushless DC Motor, or simply a BLDC. For this type of motor to be continuously rotating, the commutation on the stator current is achieved electronically with the aid of a motor drive, thus avoiding the use of mechanical brushes. By eliminating the mechanical brushes, the BLDC reliability is vastly improved. Figure 1.3 shows a schematic diagram of a BLDC system. A BLDC motor generally has the following characteristics:

1. It has multiphase stator windings, usually three.
2. It has a multi-pole permanent magnet rotor.
3. Correct operation requires correct commutation of the stator current, generally by a pulse width modulation (PWM) drive; thus, the current in a phase winding is alternating.
4. Combined with its drives and an appropriate controller, the BLDC can be controlled as if it were a regular DC motor [2]. Figure 1.4 shows a typical BLDC servo motor control diagram.

Unlike a DC motor, an AC motor has a sinusoidally wound stator, which generates a sinusoidal distributed and rotating magnetic field in the air gap as the current in the stator windings alternates. Based on the rotor type, AC motors can be divided into synchronous and asynchronous motors.

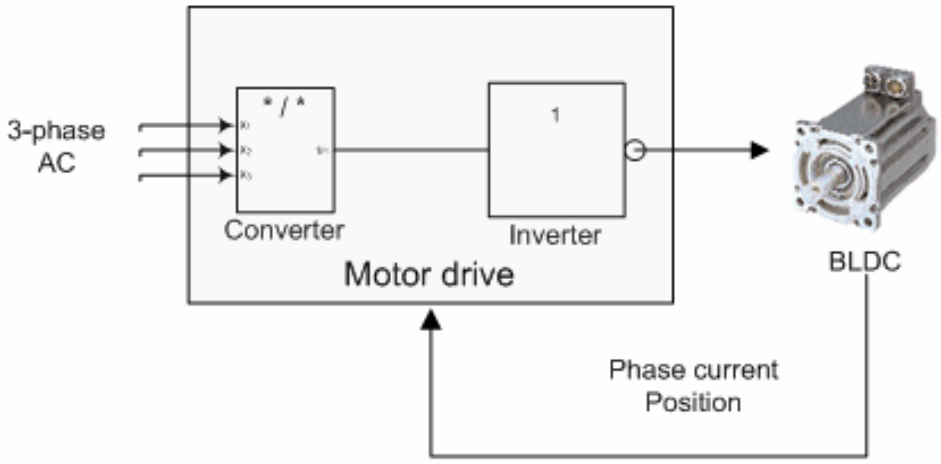


Figure 1.3, BLDC system setup

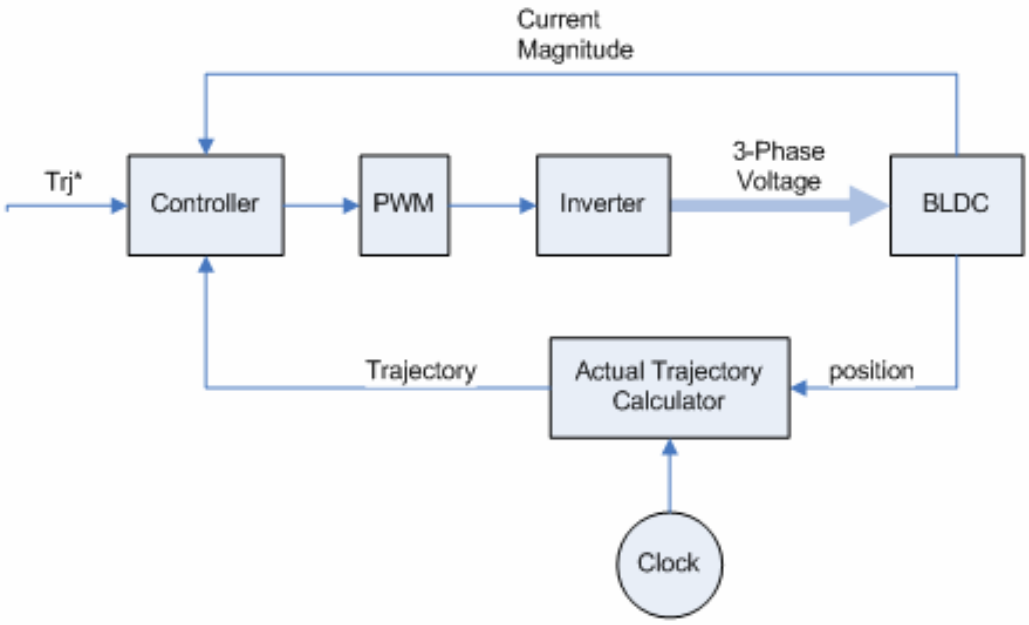


Figure 1.4, BLDC servo control diagram

A synchronous motor is characterized by the capability of being operated with the same speed for the rotor and the stator in steady state operation.

The torque on the rotor is generated by the direction difference between the stator and rotor magnetic fields. The rotor of a synchronous motor can be a permanent magnet or winding. For a wound rotor, the current in the rotor must be kept constant in order to be operated as a synchronous motor, which generally requires a constant current power source.

An asynchronous motor, on the other hand, needs a speed difference, which is called slip, between the rotor and stator magnetic fields in order to generate torque. The rotor does not have an external power source, so it is also called induction motor.

An induction motor only needs an AC power source to be operated, is virtually maintenance free and can accommodate a large range of loads. Because of its simplicity to use, it is the best choice for applications where no servo performance is required.

The reason an induction motor cannot be used as a servo motor is that the speed of the rotor is fixed by the AC power source and the motor construction.

With the help of a power electronic drive, an induction motor can be controlled as a servo motor. A typical control system configuration is shown in Figure 1.5.

Even though an induction motor can be used as servo motor, it generally has a large power-weight ratio when compared with the BLDC servo motor.

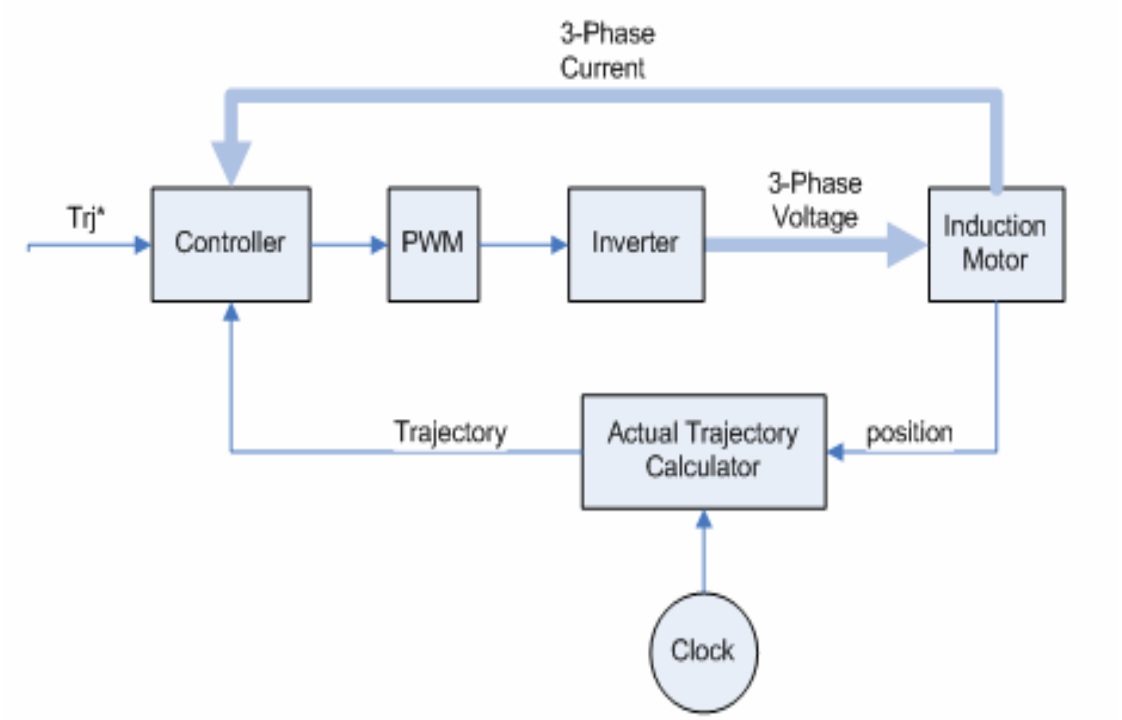


Figure 1.5, AC servo control system diagram

Because of advancements in permanent magnet material science, the BLDC motors have higher power output with more compact size, which is a critical constraint in applications with stringent size and weight limits, for example, in industrial robot actuation applications.

Figure 1.6 shows a survey of the power-weight ratios of midsize electric motors (2 hp to 5hp power) from some of the major manufacturers in industry: Allen-Bradley, Reliance Electronic, Danaher, Baldor and Bayside. Figure 1.6 also shows that BLDCs have about half of the weight of a normal AC motor with similar power output.

Even though a BLDC motor provides adequately large power with a compact size, it cannot be used in a robot actuator directly. The reason is that it generally has a high rated speed and a low rated torque.

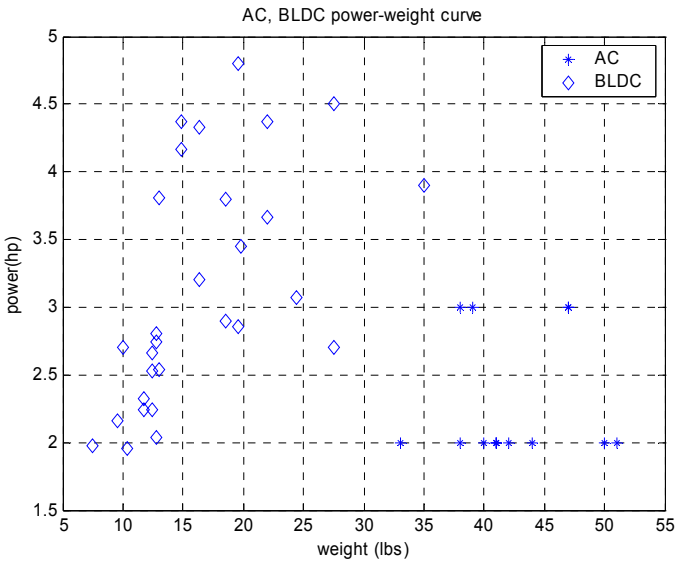


Figure 1.6, AC, BLDC power-weight curve

For example, the Allen Bradley MPL-420P-M BLDC motor has a rated speed of 5000 rpm and a rated torque of 4 N-m. However, a robot arm with a similar power rating has a relatively low speed and high torque. For example, the ABB IRB6400/3.0-100 industrial robot has a load capacity of 100 kg with speed of about 20 rpm, and the torque output is about 3500 N-m. The Titan II hydraulic manipulator is capable of handling about 100 kg load with a speed of about 3.5 rpm, and the corresponding torque requirement is about 3000 N-m.

The speed and torque incompatibility between high speed electric motor and low speed robot joints must be resolved in order to use these electric motors. The conventional solution is to attach a fixed ratio gear box to the motor output shaft, and thus expand the torque capability of the motor with the penalty of decreased speed. By using an appropriate motor and gear box, an electric motor driven robot can handle a relatively large load at a low speed; for example, the ABB IRB6400R can handle up to 500 kg with 2.3 m reach.

A technical limitation of this approach occurs when the robot is used to handle small weight. It cannot make full use of the motor power because the maximum speed of the manipulator is determined by the gear box and motor maximum speeds unless a motor magnetic field weakening technique is used. Other problems include maintenance of the gearbox, gear backlash, and gear tooth wear, which generate noise, vibration and performance degradation.

In order to eliminate a gear box in the robot design, researchers have been trying to design a high torque, low speed servo motor; such motors are called direct drive rotary (DDR) motors. DDR motors generally have large diameters due to the need for multiple stator phases, and the weight is generally much higher than BLDC and induction motors. Good examples of DDR motors are Danaher's Kollmorgen direct drive rotary D-series motors, which were selected as Products-of-the-Year by Electronics Products magazine. Figure 1.7 shows the power-weight relation for BLDC, AC, and DDR motors.

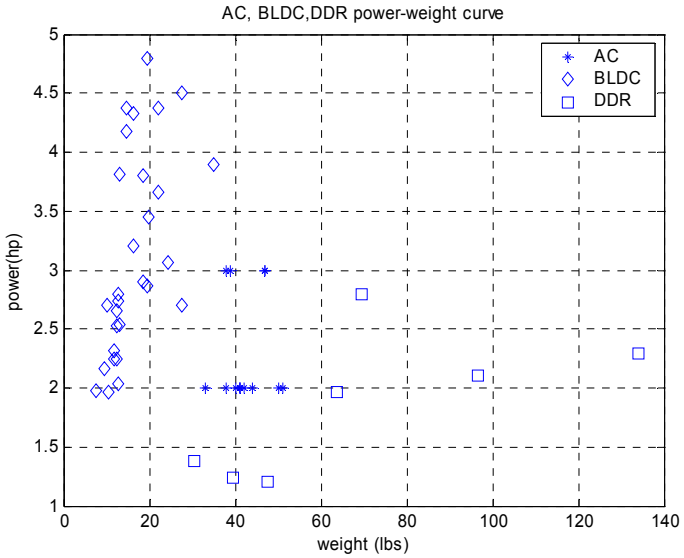


Figure 1.7, AC, BLDC, DDR power-weight curve

Even though the DDR motor has a higher weight than both the BLDC and AC motors; because of the high torque and low speed capability, the gear box can be small or even eliminated, which causes the system weight to decrease. For example, with a similar power rating as the 2 KW Allen Bradley motor mentioned earlier, the Kollmorgen DDR motor has a torque of 56 N-m and speed of 250 rpm. In order to achieve the 3000N-m torque of the Titan II, the required gear ratio is about 54, whereas for the BLDC motor, the required gear ratio is 750; this is more than 15 times the requirement for the DDR motor.

It is not easy to tell which system has the large total weight by only comparing the motor weight, but existing DDR motors generally have a larger overall size compared with BLDC motors with the same power rating. For example, the Allen Bradley BLDC motor, MPL-420P-M, has an envelope diameter of about 6 inches, whereas the Kollmorgen DDR of similar power rating has an envelope diameter of about 11 inches. In summary, hydraulic actuators are still used despite their obvious disadvantages. Electric motors are the most widely used actuators in the industrial robot applications. With same power rating, a BLDC actuator provides more compact design than an induction motor design because the BLDC has a higher power to weight ratio.

An Introduction of TBA – Ideas, Design, and Integration

As discussed in the previous section, combined with a fixed ratio gearbox, a BLDC can expand its torque range with the penalty of reduced speed. By using an appropriate gearbox, a BLDC can often match necessary payload

requirements, even these in the range of hydraulics. The feasibility of TBA has been proven by an early project funded by DOE under grant # DE-AC26-01NT41309. The TBA (Transmission based servo-actuator system) extends the idea by using a multi-ratio gearbox to replace the single ratio gearbox.

An obvious argument about this idea is that the TBA increases system complexity, which may decrease reliability. But further analysis shows that the TBA has advantages compared with a single fixed ratio gearbox in terms of operation efficiency, which can be illustrated by the design example introduced later in this section. Before design of a TBA system, a baseline has to be established. The Titan II hydraulic actuator at the shoulder pitch is taken as the baseline actuator. A major consideration to use the Titan II as a baseline actuator is that it provides enough load capability to cover most D&D tasks. Table 1.1 shows the performance specifications of a Titan II manipulator. When the Titan II is at full reach, it can be modeled as a simple beam as shown in Figure 1.8, where T is torque, θ is angular position, ω is angular velocity, α is angular acceleration, L is the manipulator length, M_a is the manipulator mass, M_p is the load mass and g is the gravity constant.

Table 1.1, Titan II performance specifications

Maximum reach	1915 mm	75.4 in
Maximum payload at full reach	113 Kg	250 lb
Manipulator mass	103 Kg	225 lb
Manipulator center of gravity	1000 mm	39.27 in
Maximum angular velocity (zero load)	0.73 rad/s (42º/s, 7rpm)	same
Maximum angular velocity (half load)	0.52 rad/s (30º/s, 5rpm)	same
Maximum angular velocity (full load)	0.35 rad/s (20º/s, 3.3rpm)	same

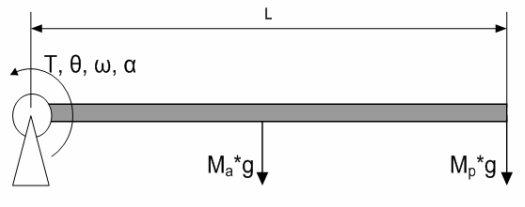


Figure 1.8, Titan II equivalent model

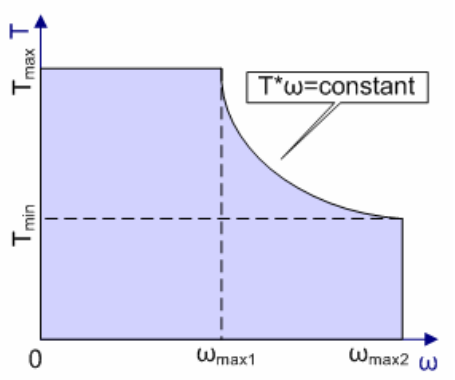


Figure 1.9, Baseline torque speed curve

When the manipulator is at a horizontal position as in Figure 1.8, if acceleration is zero, the relationship between the hydraulic actuator torque output and the manipulator angular velocity can be shown in Figure 1.9.

It is assumed that the hydraulic actuator is able to be operated at rated power between full load and zero load. Notice that at zero load, the required torque is not zero because of the manipulator weight. The power output of the baseline hydraulic actuators can be calculated as in Equation (1.2).

$$P_{\text{titan}} = T_{\max}\omega_{\max 1} = T_{\min}\omega_{\max 2}$$

where

$$\omega_{\max 1} - \text{maximum speed when load torque is maximum}$$

$$\omega_{\max 2} - \text{maximum speed when load torque is minimum} \qquad (1.2)$$

After the baseline is established, the design example can be formulated. Suppose an electric motor actuator is to be used to match the baseline hydraulic actuator's performance, the following design questions are apparent:

1. Between a single fixed ratio gear box and a multiple ratio gearbox, which is a better solution?
2. If a multiple ratio gear box is better, what is the best number of gear ratios?
3. Is it possible to find a single quantitative parameter which can be used to compare different designs?

First, let's look at a typical BLDC motor torque speed curve, which can be shown as the shaded area in Figure 1.10. For simplicity, a rectangular shape is used for analysis as in Figure 1.10.

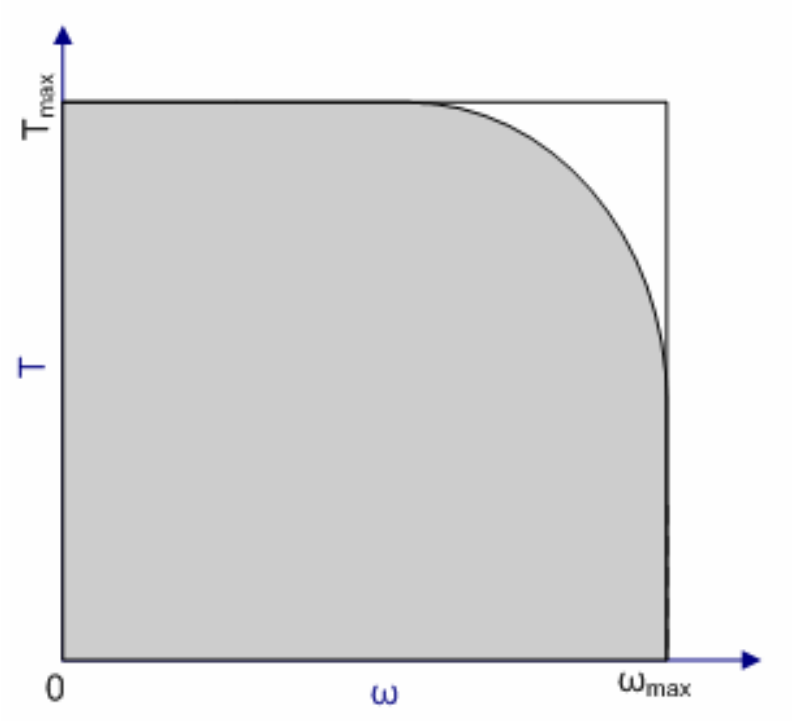


Figure 1.10, Typical BLDC torque speed curve

Before comparing the different design options, it is useful to define some dimensionless indices that characterize the sizing properties of the actuators.

Definition 1.1: Power index

Power index (ζ_p) ***- The ratio between required power rating and the power of the baseline hydraulic actuator.***

$$\zeta_p = \frac{P_r}{P_{\text{Titan}}}$$

P_r – Required power of a design (rated power) (1.3)

P_{titan} – Baseline power

An immediate observation is that a design must provide at least the power of the Titan II manipulator in order to match its torque and speed capability. Therefore, by obvious assumption, the optimal value of power index is 1.

A desired design should have a power index close to 1, and a design with a power index greater than 1 is called power over-designed.

Definition 1.2: Load region index (ζ_a) ***- The ratio between area covered by a new design in the torque speed curve and the area of the baseline hydraulic actuator torque speed curve.***

$$\zeta_a = \frac{A_r}{A_{\text{Titan}}}$$

A_r – Area covered by the torque speed curve for a design (1.4)

A_{titan} – Area covered by the torque speed curve for the baseline

A desirable design will have a large load region index. Again, a similar assumption as the previous one is used, that is, the optimal load region index is 1.

Therefore, a design should have a load region index close to 1, and a load region index greater than 1 is over designed.

Definition 1.3 : Load matching index (ζ) - The summation of equally weighted power index and load region index.

$$\zeta = 0.5\zeta_a + 0.5\zeta_p \qquad (1.5)$$

The load matching index is relevant because, conceivably, the design objective of the TBA system is to achieve a large load region index with a small power index, which is equivalent to achieve large area coverage of the baseline torque speed curve without going beyond the region. A design should have an ζ close to 1, and a design with an ζ greater than 1 is over designed.

In this research, four designs are evaluated based on the load matching index. The four designs are fixed gear reduction, two speed DVT, three speed DVT, and four speed DVT.

Now, let's first look at the design of a BLDC combined with a fixed ratio gear box. In order to meet the torque and speed requirements simultaneously, a BLDC motor must provide a rating power of at least $T_{\max} \times \omega_{\max 2}$ as in Figure 1.10, which is about two times of the power output of hydraulic actuator as described in Equation (1.2).

The increased power rating generally means increased motor weight and/or size. For this design, the load region index is ~2 and the power index is 2, so the load match index is ~2. For all other three designs, the power indexes are the same, since they use the same BLDC motor. In this analysis, the power index

is 1 because the motor is selected to match the Titan II power rating. When combining a BLDC with a two speed DVT, the gear ratios are chosen such that the high torque region and the load torque region on the baseline torque speed curve are covered.

Specifically, the lower ratio is chosen to match the maximum speed at lowest load, the area covered by the torque speed curve associated with the low ratio can be represented approximately by the region is 0-4-5-6-0 in Figure 1.11; the high ratio is chosen to match the maximum speed at the highest load, such that the area covered by torque speed curve associated with the high ratio can be represented approximately by the region is 0-1-2-3-0 in Figure 1.11. As a result, the area covered by the torque speed of this two speed DVT system has a torque speed curve as shown in Figure 1.11, which is overlaid with the baseline torque speed curve as in Figure 1.9. The torque speed curve for low ratio is the rectangular area enclosed by lines 0-1-2-3-0, and 0-4-5-6-0 for high ratio; thus, the total area for the two speed design is enclosed by 0-1-2-7-5-6-0.

Even though this design matches the performance specification for the maximum load and minimum load, and power requirement for the above design is similar to that of the baseline actuator as calculated in Equation (1.2).

One problem still exists: the area enclosed by 2-5-7-2 is not reachable by this two speed design, which means this two speed design cannot work in the region enclosed by 2-5-7-2, and the load matching index is 0.943. By adding one more gear ratio between the high and low ratio to the two speed DVT, a three speed DVT is formed.

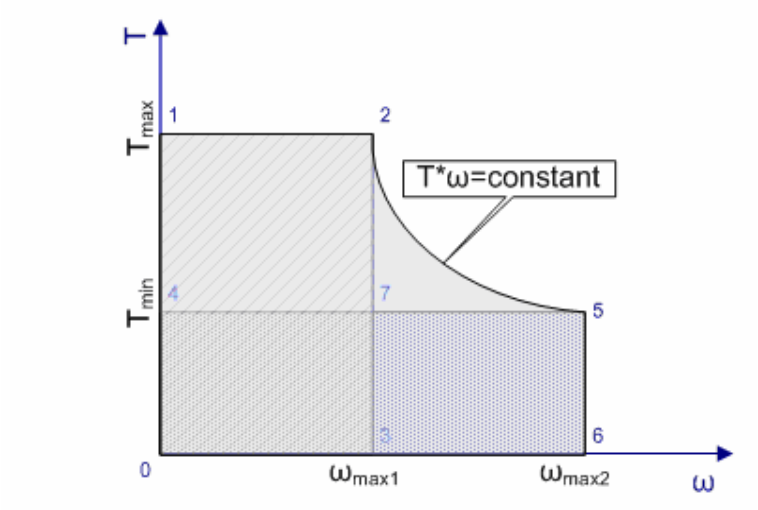


Figure 1.11, BLDC with two speed DVT torque speed curve

The new added gear ratio is chosen such that the load region index is maximized. The torque speed curve for the high and low ratios are the same as those of two speed DVT, and the area covered by torque speed curve associated with the middle ratio is the enclosed rectangle area 0-8-9-10-0 in Figure 1.12.

The total covered area for this three speed design is an enclosure of 0-1-2-11-9-12-5-6-0. The torque speed curve of this three ratio design overlaid on the baseline torque speed curve can be shown in Figure 1.12. The power index of the three speed DVT is the same as the two speed DVT, but the load region index is larger. Thus, the load matching index is larger and is closer to 1, the actual value of the load matching index is 0.968.When another gear ratio is added, a four speed DVT is formed. The high ratio and the low ratio are the same as the two speed and three speed DVT.

The two ratios between the high and low ratio are chosen such that the load region index is maximized. The four speed DVT has a load matching index of 0.975.

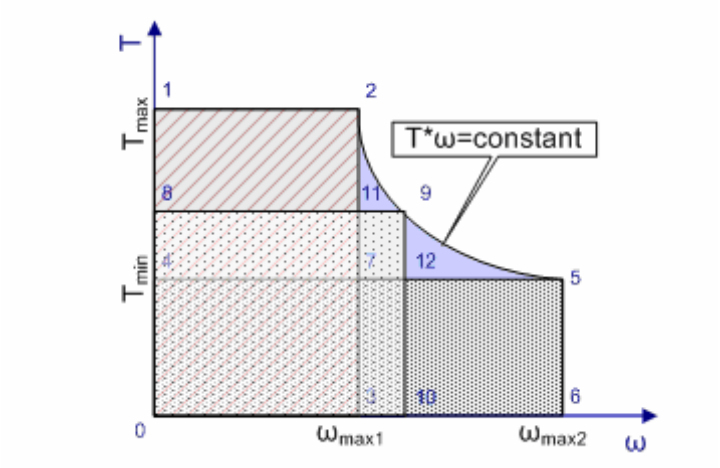


Figure 1.12, BLDC with three speed gear box torque speed curve

If more intermediate gear ratios are added, more area will be covered, thus increasing the load matching index, we can say that the design matches the baseline hydraulic actuator better. Ideally, if the number of gear ratios could be infinitely large, as in a continuously variable transmission, the load matching index would become closer to 1. However, as the number of gear ratios increases, the complexity of the gear box increases as well. There is a tradeoff between number of gears and the complexity of the system. A heuristic rule is that a design should have load matching index closer to 1 with less gear ratios.

Based on the above analysis, the load matching index for different designs can be calculated by Equation (1.5). These results are shown in Table 1.2.

From Table 1.2, fixed ratio gear box is greatly over designed and thus demonstrated the issues with fixed ratio electrical servo-actuators. The other three designs are candidates for further comparison. Two speed DVT is eliminated because it is significantly under designed, which can be verified by examining the torque speed curve as shown in Figure 1.11.

Table 1.2, Load matching index for different TBA design

Design No.	Name	Load matching index
Baseline	Baseline hydraulic actuator	1
1	Fixed ratio gear box	~2
2	Two-speed DVT	0.943
3	Three-speed DVT	0.968
4	Four-speed DVT	0.975

Three speed DVT increases the load matching index by 0.025 by adding one more gear ratio on two speed DVT.Four speed DVT increases the load matching index by 0.007 by adding one more gear ratio on three speed DVT. Three speed DVT was chosen against four speed DVT because the three speed DVT has a larger increase on the load matching index by adding only one more gear ratio.

Following the same analysis, if the TBA has infinitely many gear ratios, or if it can change gear ratios continuously, the TBA can match the baseline hydraulic actuator torque speed curve with high accuracy. In principle, this can be achieved with a continuously variable transmission (CVT) design. A big advantage associated with gear variation of CVT is that the gear shifting is essentially continuous and the transient disturbance should be negligible. This design is not the focus of this research, and more information can be found in [1].

TBA Prototype and the Experimental System

The TBA prototype and associated experimental system includes four subsystems: a PC, a BLDC servo, a DVT, and a load dynamometer (DM). The whole system is shown in Figure 1.13.

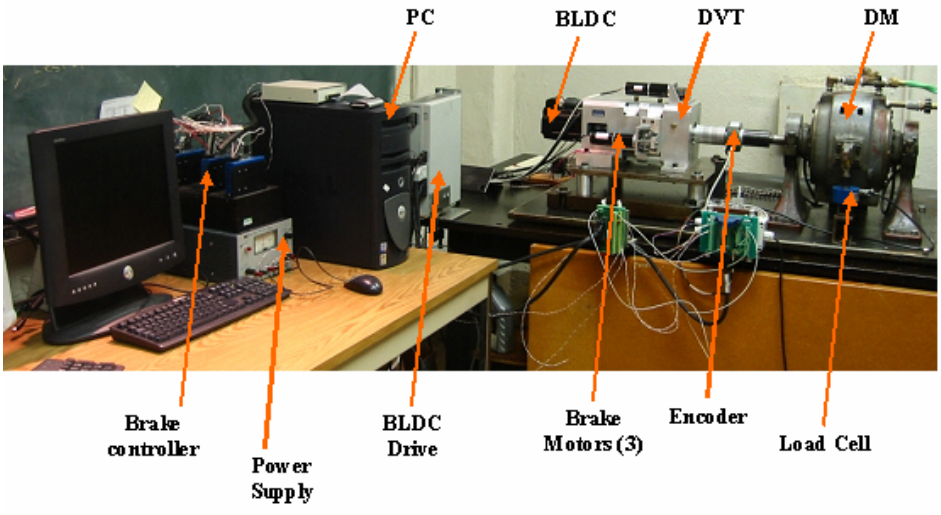


Figure 1.13, TBA system hardware layout

The PC is used to control the BLDC servo and DVT shifting. The PC hosts all the control software and data acquisition hardware for D/A, A/D, encoder, and digital I/O.

The BLDC has a rated power of ~2 KW with a rated torque of ~4 N-m. The dynamometer is used to emulate the load torque produced by the robot arm. The torque generated by the dynamometer is determined by the input speed and the reference voltage. One limitation of this dynamometer is that it cannot generate any load when the speed input is zero, and the torque generated at low speed is limited. In this research, experimental result showed that the dynamometer can track the desired load torque when the input speed is adequate. The experimental results will be shown in Chapter 5.

The three speed DVT is the core design of the TBA prototype. It has three planetary gear sets serially connected together. The three speed DVT mechanical construction diagram is shown in Figure 1.14. The mechanical connection relationship of the three planetary gear sets is:

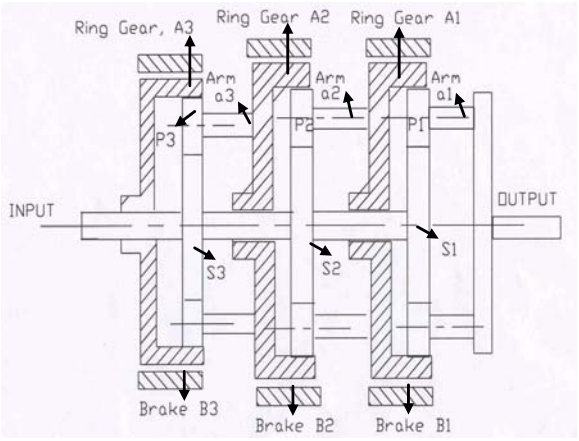


Figure 1.14, Three speed DVT

1. The carrier, or arm, of the previous planetary gear set is rigidly connected to the ring gear of the next planetary gear set (a3 to A2, and a2 to A1);
2. All sun gears are rigidly connected with each other (S3 to S2 to S1).

This configuration was used mainly because it provides the required gear ratios with three almost identical planetary gear sets. The three gear ratios are 2.8, 4, and 7. The similar size of all three planetary gear sets makes the brake band and the DVT housing design easier. Figure 1.14 shows the layout of the three speed DVT design.

The three speed DVT always functions under the condition that only one brake is engaged. When one of the three brakes is engaged, it becomes a one degree of freedom mechanism, and an output torque can be generated with the input torque and the brake torque. While no brake is engaged, the DVT is a mechanism with two degrees of freedom, which means that applying a single input cannot produce an output torque to carry the load. When more than two brakes are engaged, it is over constrained, and this condition causes abnormal brake wearing, and thus should be avoided.

By energizing a particular brake, thus locking the corresponding ring gear, the DVT functions as a gear box with a fixed gear ratio. Therefore, the DVT can function as three different gearboxes when different brakes are engaged, thus providing multiple-speed operation and corresponding expansion of the torque speed curve of the actuator.

Scope and Organization of the Dissertation

The main purpose of this research is to explore various control issues of the TBA system, including control algorithms and their digital implementation.

The fundamental goal is to identify control methods that provide high performance servo control while attenuating the disturbances due to discrete gear shifting action. In association with this research, a PC-based control software platform which provides the real time performance required by TBA control is developed. The remainder of this dissertation is organized as follows:

In Chapter 2, the TBA prototype overview will be given and mathematical models of TBA system will be derived in a modular fashion. In Chapter 3, TBA system stability is discussed, necessary conditions for the TBA system stability are established and proofs are given. A stable switch strategy is analyzed and the proof of its transition stability is given. In the last part of Chapter 3, a feedback based controller design is also presented and its equivalent PID controller is given. In Chapter 4, TBA control software design is discussed in detail; implementation issues using a real time operating system - Linux/RTAI is analyzed. In Chapter 5, Matlab/Simulink simulations of various controllers are

developed, and simulation results analyzed. Experimental results are also presented and analysis of the control system performance is given. In Chapter 6, major contributions and conclusions are given based on the experimental and simulation results. Future research needs are also listed in this chapter. This research contributes to the literature and research knowledge base in the following ways:

1. Exploration and solution of the control problems of the TBA in the hybrid system control context.
2. Expansion of the fundamental understanding of the practical control issues in TBA.
3. Design, analysis, and real time implementation of the TBA control system and identification of the most suitable control strategy for the TBA.
4. Methods that can be extended to use in a much broader range of hybrid dynamic systems.

CHAPTER 2

Mathematical Modeling

The TBA prototype system, which is composed of a BLDC motor, a three speed DVT and a robot arm, is a nonlinear dynamic system. An exact mathematic model would be very complicated because of the nonlinear properties such as saturation, backlash, and coulomb friction. This research represents the first level of investigation into TBA control and as a result these nonlinear properties will be ignored. It is assumed that a simplified model is sufficient for the initial controller design.

The purpose of this chapter is to derive a mathematical model which can represent the main dynamic properties of the TBA and yet is convenient for the controller design and analysis. The process used is a bottom up procedure; specifically, three component models, a BLDC servo model, a three speed DVT model, and a robot arm model, are derived and simplified individually. In the end, a simplified form of a final TBA dynamic equation is given.

BLDC Model

As mentioned in Chapter 1, a BLDC motor has a permanent magnet rotor and a stator winding. By alternating the electrical current direction in the stator winding, a rotating magnetic field is generated in the air gap, thus leading to the rotational motion of the motor shaft. Compared with a regular DC servo motor, a BLDC servo motor has a number of advantages, including a higher power to

weight ratio, smaller friction, less maintenance, non arcing commutation, and better heat dissipation.

Even though commercial BLDCs share the same operation methods in high performance practice, the construction details are treated as proprietary information, and are different from manufacturer to manufacturer. The performance of a BLDC also depends on its construction. Some commonly used design parameters are: permanent magnet properties, number of pole pairs and the stator winding technique. A general comparison of BLDC performance with respect to magnet material, number of poles and physical dimensions can be found in [3]. As a general rule, as the number of pole pairs increases, the motor torque to weight ratio and efficiency increase as well.

Despite the rich diversity of BLDC motor configurations, a functional construction diagram can be represented as in Figure 2.1. This construction has three phase uniformly distributed windings and four pole pairs. Hall-effect sensors are used to measure the rotor position for correct commutation.

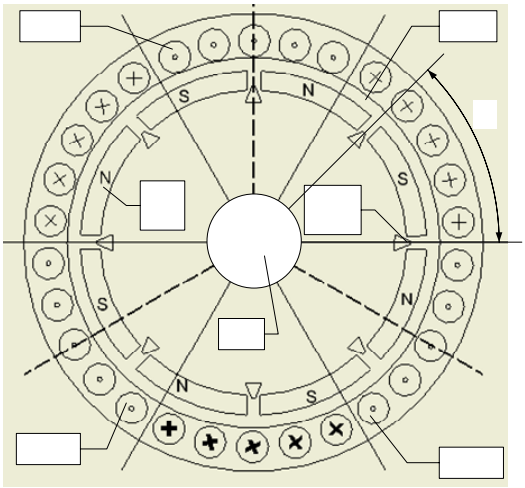


Figure 2.1, A functional BLDC construction diagram with four pole pairs

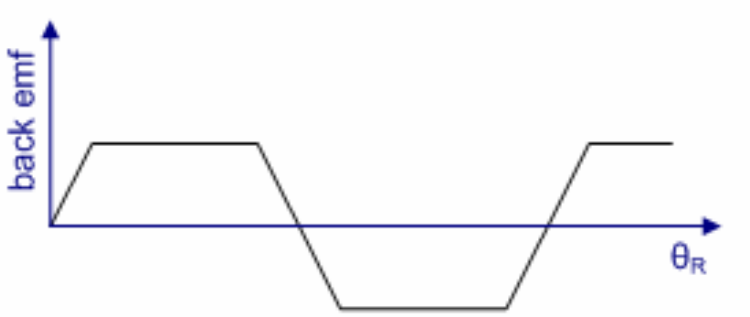


Figure 2.2, Trapezoidal back EMF

In general, a BLDC has a uniformly distributed stator winding, which has a trapezoidal shaped back electromagnetic force (EMF) as shown in Figure 2.2.

In order to have a stable steady state power output, current in the stator winding must alternated according to the rotor position[2]. The stator winding of a BLDC can also be sinusoidally distributed. The operation and control of this type of machine is essentially the same as a regular AC synchronous machine as discussed in Chapter 1.

This section presents the derivation of a unified simplified model for both types of BLDC motors, such that it is unnecessary to know the detailed construction of the BLDC motor in use. It is also assumed that this simplified model is sufficient for controller design and analysis. For a sinusoidally wound stator, the dynamic equation of this type of BLDC motor, when expressed in the rotor reference frame, can be simply represented as shown in Equation (2.1).

$$\begin{aligned}
\frac{di_q}{dt} &= \left(-Ri_q - nL_d\omega_R i_d - nk_e\omega_R + v_q\right)\Big/L_q \\
\frac{di_d}{dt} &= \left(-Ri_d - nL_q\omega_R i_q + v_d\right)\Big/L_d \\
\frac{d\omega_R}{dt} &= \left(n\sqrt{\tfrac{3}{2}}\left(k_e i_q + (L_d - L_q)i_q i_d\right) - T_{load} - b_m\omega_R\right)\Big/J
\end{aligned} \qquad (2.1)$$

R – Resisitance, n- number of pole pairs

Equation (2.1) is a nonlinear dynamic system because of the product terms of speed and current.

The following four relations can be used to simplify the model to obtain a simplified linear model of the motor:

1. Employing a high gain PI current feedback controller [4] forces the currents in Equation (2.1) to follow the reference currents promptly.

$$i_d = i_{dref} \quad (2.2)$$

2. The reference current in d-axis is controlled to zero.

$$i_{dref} = i_d = 0 \quad (2.3)$$

3. When the air gap is uniform, the following relation holds.

$$L_d = L_q = L \quad (2.4)$$

4. For the d-q voltage, the following relation holds.

$$v_d = v_q = \sqrt{\tfrac{3}{2}}V \quad (2.5)$$

By substituting Equation (2.2) through Equation (2.5) to Equation (2.1). The final simplified form of the dynamic equations is:

$$\begin{aligned} \frac{di_q}{dt} &= \left(-Ri_q - nk_e\omega_R + \sqrt{\tfrac{3}{2}}V\right)\Big/L \\ 0 &= \left(-nL_q\omega_R i_q + \sqrt{\tfrac{3}{2}}V\right)\Big/L \\ \frac{d\omega_R}{dt} &= \left(n\sqrt{\tfrac{3}{2}}k_e i_q - T_{load} - b_m\omega_R\right)\Big/J \end{aligned} \quad (2.6)$$

The first and third equations in Equation (2.6) can be solved independently, which can be rewritten in the following form, shown in Equation (2.7).

$$\frac{di_q}{dt} = \left(-Ri_q - nk_e\omega_R + V_e\right)\Big/ L$$
$$\frac{d\omega_R}{dt} = \left(K_e i_q - T_{load} - b_m\omega_R\right)\Big/ J \qquad (2.7)$$
$$V_e \equiv \sqrt{\tfrac{3}{2}}V, K_e \equiv n\sqrt{\tfrac{3}{2}}k_e$$

Equation (2.7) is in the same form as a regular DC motor model as shown in Equation (1.1).

For a trapezoidal back EMF type BLDC, the general form of a three phase, single pole pair BLDC motor model can be found in [2]. A three phase, multi pole pair BLDC motor model is given by the following:

$$\begin{bmatrix} \dot{i}_{s1} \\ \dot{i}_{s2} \\ \dot{i}_{s3} \end{bmatrix} = \frac{e_p}{L_s + M}\begin{bmatrix} e(n\theta_R) \\ e(n\theta_R - 2\pi/3) \\ e(n\theta_R - 4\pi/3) \end{bmatrix}\omega_R - \frac{R_s}{L_s + M}\begin{bmatrix} i_{s1} \\ i_{s2} \\ i_{s3} \end{bmatrix} + \frac{1}{L_s + M}\begin{bmatrix} u_{s1} \\ u_{s2} \\ u_{s3} \end{bmatrix}$$
$$\dot{\omega}_r = -\frac{n}{2}\frac{k_t}{J}\left[e(n\theta_R)i_{s1} + e(n\theta_R - 2\pi/3)i_{s2} + e(n\theta_R - 4\pi/3)i_{s2}\right] - \frac{T_{load}}{J} \qquad (2.8)$$
$$\dot{\theta}_R = \omega_R$$

where

$$e(\theta_R) = \frac{\partial(\lambda_{\theta_R})}{\partial\theta_R}$$

λ_{θ_R} – flux linkage, T_e – motor torque

u_{s1}, u_{s2}, u_{s3} – stator votage

i_{s1}, i_{s2}, i_{s3} – stator cuurent

L_s, M – self and mutual inductance

θ_R, ω_R – rotor angular position and velocity

R_s – stator resistence, T_{load} – load torque

τ_p – torque constant, n – number of pole pairs

It is shown that, combined with a motor drive, the dynamic equation can be simplified into this form [2]:

$$J\frac{d\omega_R}{dt} = k_t I_p - T_{load} - b_m \omega_R$$

where,

I_p - magnitude of the field current.

k_t - torque constant, b_m - friction coefficient

(2.9)

Notice that Equation (2.9) has the same form as the second equation in Equation (2.7), so the two types of BLDC motors have the same mechanical dynamic equation, which is similar as a regular DC motor shown in Equation(1.1). The construction of a BLDC resembles an inside out regular DC motor, while the control of a BLDC requires an electronic device that can alternate the current in the stator windings in order to obtain proper commutation. This device is generally referred to as a BLDC drive. With a motor drive, the BLDC motors of either type can be represented by Equation (2.9), and this simplified model will be used to represent the motor in the TBA prototype.

Three Speed DVT Model - A Hybrid Dynamic System

The three speed DVT is composed of three planetary gear sets. The connection between adjacent gear sets is the previous carrier gear messed to the next ring gear as shown in Figure 1.14. This configuration provides three desired gear ratios with identical ring diameters, which is proven to be a significant advantage for the braking system mechanical design, and the overall TBA design.

The mathematical model of a DVT can be derived using Newton's second law [1], which produces two coupled second order systems that can be

expressed in a single fourth order differential equation. In order to solve this equation, all braking torques must be known, which is only true when there is a slip between the braking band and the ring gear.

When the brake is fully engaged and the ring gear is locked, the braking force is a static friction whose magnitude and direction are determined by the servo motor input torque and the load torque, which can not be treated as known torques.

In this dissertation, a new DVT model is derived based on the Lagrange's equation of motion by assuming that the braking torque is large enough such that the ring gear can stop in a very short time and thus slip can be neglected. The following assumptions are also used to derive the mathematic model:

1. Backlash is neglected.
2. Gears and shafts are treated as rigid bodies.
3. Friction effects have the form:

$$F = -b\dot{x} \qquad (2.10)$$

The modeling process starts with a single planetary gear set with the ring gear locked, and then a full DVT model is derived by following a similar procedure. In supplement of the development of the DVT model, it is useful to review a number of concepts and definitions [5].

Pfaffian form - virtual displacement

$$\begin{aligned} &a_x(x,y,z,t)\delta_x + a_y(x,y,z,t)\delta_y + a_z(x,y,z,t)\delta_z = 0 \\ &a_x(x,y,z,t), a_y(x,y,z,t), a_z(x,y,z,t) \text{ - contraints} \\ &\delta_x, \delta_y, \delta_z \text{ - virtual displacements} \end{aligned} \qquad (2.11)$$

Holonomic and scleronomic constraints

When Equation (2.11) is time integrable, the constraint equation can be reduced to the holonomic constraint given in Equation (2.12).

$$f(x, y, z, t) = 0 \qquad (2.12)$$

When there is no explicit time variable in the holonomic form, the new form is called a scleronomic constraint as shown in Equation (2.13).

$$f(x, y, z) = 0 \qquad (2.13)$$

Rayleigh's dissipation function

If dissipation force can be expressed in the following form

$F_{x_i} = -b_{x_i}\dot{x}_i, F_{y_i} = -b_{y_i}\dot{y}_i, F_{z_i} = -b_{z_i}\dot{z}_i$

then dissipation energy can be expressed as follows (2.14)

$$F = \frac{1}{2}\sum_{i=1}^{n}\left(b_{x_i}\dot{x}_i^2 + b_{y_i}\dot{y}_i^2 + b_{z_i}\dot{z}_i^2\right)$$

Lagrange's Equation of Motion

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_k}\right) - \frac{\partial L}{\partial q_k} + \frac{\partial F}{\partial \dot{q}_k} = Q_k + \sum_{l=1}^{M} a_{lk}\lambda_k \ , \ (k = 1, 2, ..., N)$$

$L \equiv K - V$, K-kinetic energy, V - potential energy

F - dissipation energy

q_k - generalized coordinate (2.15)

Q_k -generalized force

λ_k -Lagrange's multiplier

a_{lk} - Lagrange's multiplier coeefficient

When the constraints are sceleronomic as shown in Equation (2.13), a_{lk} in Equation (2.15) can be calculated with Equation (2.16).

$$a_{lk} = \frac{\partial f_l}{\partial q_k} \qquad (2.16)$$

f_l - ℓth constraint

Hybrid Dynamic System (HDS)

HDS - A dynamic system which has both continuous dynamics and discrete dynamics. The mathematical representation is given by Equation (2.17).

A three speed DVT is a hybrid dynamic system in the sense of Equation (2.17). When the DVT stays in one gear, it is one continuous dynamic system; after the shift, it is another continuous dynamic system.

$$\dot{x} = f(x(t), \sigma(x,t), u(t)), \text{ with } x(0) = x_0, \sigma(x_0, 0) = \sigma_0$$

where

$x(t)$ – n×1 state vector

u(t) - m×1 control vector

$\sigma(x,t)$ – an m×1 discrete event vector

$f : (R^n, D^m, R^m) \to R^n$, D is a set of all possible discrete events

(2.17)

The state change is a discrete action that switches one continuous dynamic system into another continuous dynamic system. This switching action can cause instability and performance degradation, as will be discussed in Chapter 3. As far as the modeling is concerned, the hybrid system can be treated as several independent continuous dynamic systems supervised by a high level discrete state regulator; the behavior of this regulator can be modeled as a finite state machine shown in Figure 2.3.

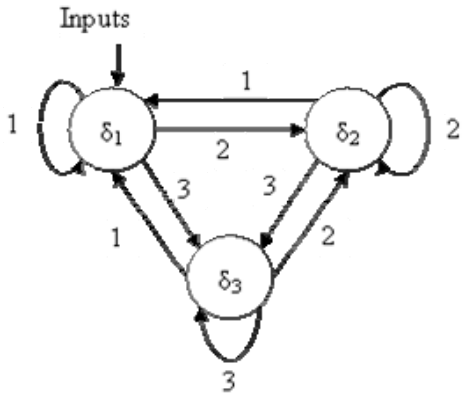


Figure 2.3, The gear shift system modeled as a finite state machine

The input alphabet is 1, 2 and 3, and the input word can be any combination of 1, 2 or 3. This finite state machine actually plays a core role in a discrete state generator, and the detailed TBA implementation will be introduced in Chapter 3.

Definition 2.1: TBA gear numbers

1st gear (δ_1):

The DVT is said to be in the 1st gear if the ring gear A1 in Figure 1.14 is locked by engaging brake B1 as in Figure 1.14.

2nd gear (δ2):

The DVT is said to be in the 2nd gear when the ring gear A2 in Figure 1.14 is locked by engaging brake B2 as in Figure 1.14.

3rd gear (δ3):

The DVT is said to be in the 3rd gear when the ring gear A3 in Figure 1.14 is locked by engaging brake B3 as in Figure 1.14.

The convention for TBA gear definition follows that of an automobile transmission. The 3rd gear corresponds to the lowest gear ratio, and the 1st gear corresponds to the highest gear ratio.

Therefore, when the BLDC is operating at a constant speed and torque, the 3rd gear produces the highest output speed and lowest output torque, and the 1st gear produces the lowest speed and highest torque.

A Single Planetary Gear Model

A single planetary gear set is a mechanism with two degrees of freedom. A functional diagram of a single planetary gear set is given in Figure 2.4.

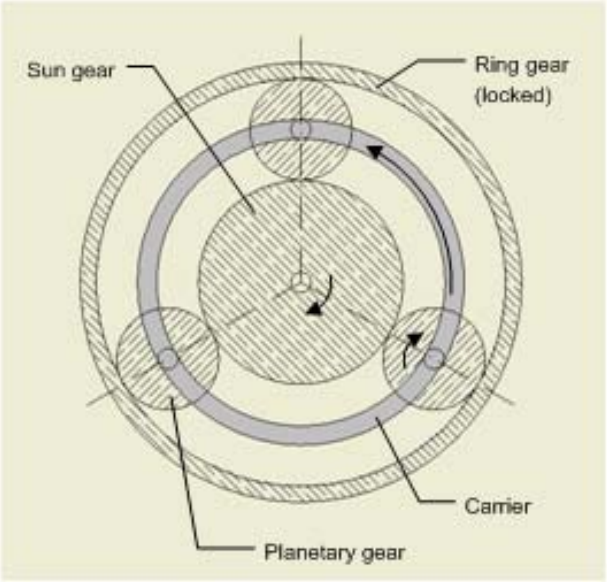


Figure 2.4, A single planetary gear set

By locking the ring gear, the planetary set becomes a one degree of freedom mechanism.

A dynamic model for the single planetary gear set is presented with the ring gear locked, for complete derivation of the model, please see the Appendix.

The energy terms of the planetary gear set are given in Equation (2.18).

$$\begin{aligned} K &= \tfrac{1}{2}\left(J_s\dot{\theta}_s^{\,2} + J_c\dot{\theta}_c^{\,2} + 3\left(J_p\dot{\theta}_p^{\,2} + m_p\dot{\theta}_c^{\,2}r_c^{\,2}\right) + J_r\dot{\theta}_r^{\,2}\right) \\ V &= 0 \\ F &= \tfrac{1}{2}\left(b_s\dot{\theta}_s^{\,2} + b_c\dot{\theta}_c^{\,2} + 3b_p\dot{\theta}_p^{\,2} + b_r\dot{\theta}_r^{\,2}\right) \end{aligned} \qquad (2.18)$$

With constraints given as follows:

$$\begin{aligned} -\dot{\theta}_s r_s + \dot{\theta}_c r_s - \left(\dot{\theta}_c + \dot{\theta}_p\right) r_p &= 0 \\ \dot{\theta}_r r_r - \dot{\theta}_c\left(r_r + r_s\right) + \dot{\theta}_s r_s &= 0 \end{aligned} \qquad (2.19)$$

The following generalized coordinates and forces are defined:

$$\begin{aligned} q_1 \equiv \theta_s,\ q_2 \equiv \theta_c,\ \ q_3 \equiv \theta_p,\ q_4 \equiv \theta_r \\ Q_1 \equiv T_{in},\ \mathrm{Q}_2 \equiv -T_{out},\ \ \mathrm{Q}_3 \equiv 0,\ \ \mathrm{Q}_4 \equiv -T_b \end{aligned} \qquad (2.20)$$

By substituting Equation (2.19) into Equation (2.16), the Lagrange multipliers can be calculated. Then substitute these results along with Equation (2.18) and Equation (2.20) into Equation (2.15), and the dynamic equations of a single planetary gear set can be given in Equation (2.21).

$$\begin{aligned}
&J_s\ddot{\theta}_s + b_s\dot{\theta}_s = T_{in} + (\lambda_1 - \lambda_2) r_s \\
&\left(3m_p r_c^2 + J_c\right)\ddot{\theta}_c + b_c\dot{\theta}_c = -T_{out} - \lambda_1 r_s + \lambda_2 (r_s + r_r) \\
&3J_p\ddot{\theta}_p + b_p\dot{\theta}_p = \lambda_1 r_p \\
&J_r\ddot{\theta}_r + b_r\dot{\theta}_r = -T_b - \lambda_2 r_r
\end{aligned} \qquad (2.21)$$

In steady state, the static braking torque is large enough such that the ring gear is stopped, and Equation (2.21) can be further simplified as Equation (2.22).

$$\frac{(r_s + r_r)}{r_s} J_s\ddot{\theta}_s + \frac{(r_s + r_r)}{r_s} b_s\dot{\theta}_s + \left(3m_p r_c^2 + J_c\right)\ddot{\theta}_c + b_c\dot{\theta}_c - \frac{r_r}{r_p}\left(3J_p\ddot{\theta}_p + b_p\dot{\theta}_p\right) = T_{in}\frac{r_s + r_r}{r_s} - T_{out} \qquad (2.22)$$

Combined with Equation (2.19), the dynamic equation of motion of the planetary gear set in final form can be represented as a second order differential equation:

$$\begin{aligned}
&\left(J_s - 3\mu(\mu+2)\gamma(\gamma-1)J_p + \gamma J_{eq}\right)\ddot{\theta}_s + \left(b_s + b_c\gamma - b_p\mu(\mu+2)\gamma(\gamma-1)\right)\dot{\theta}_s = T_{in} - \gamma T_{out} \\
&\textit{where} \\
&\mu \equiv \frac{r_s}{r_p}, \gamma \equiv \frac{r_s}{r_r + r_s}, J_{eq} \equiv \left(3m_p r_c^2 + J_c\right)
\end{aligned} \qquad (2.23)$$

The coefficient of the acceleration term in Equation (2.23) is a constant determined by the gear geometry and inertial properties, thus, it can be simplified:

$$\begin{aligned}
&J\ddot{\theta}_s + b\dot{\theta}_s = T_{in} - \gamma T_{out} \\
&\textit{where,} \\
&b \equiv b_s + b_c\gamma - b_p\mu(\mu+2)\gamma(\gamma-1) \\
&J \equiv J_s - 3\mu(\mu+2)\gamma(\gamma-1)J_p + \gamma J_{eq}
\end{aligned} \qquad (2.24)$$

A Three Speed DVT Model

As a single planetary gear set, a three speed DVT is a mechanism with two degrees of freedom when no brake is engaged, and it becomes a one degree of freedom mechanism when one of the three ring gears is locked.

The model process for a three speed DVT is similar except that the inertia and friction properties are different. Dynamic models for the three speed DVT are presented with different brakes engaged.

For complete derivation of the model, please see the Appendix.

The energy terms of the DVT are given as:

$$
\begin{aligned}
K &= \tfrac{1}{2}\left(J_{s1}\dot{\theta}_{s1}^{\,2} + J_{r1}\dot{\theta}_{r1}^{\,2} + 3J_{p1}\dot{\theta}_{p1}^{\,2} + 3m_{p1}r_{c1}^{\,2}\dot{\theta}_{c1}^{\,2} \right) + \\
&\quad \tfrac{1}{2}\left(J_{s2}\dot{\theta}_{s2}^{\,2} + J_{r2}\dot{\theta}_{r2}^{\,2} + 3J_{p2}\dot{\theta}_{p2}^{\,2} + 3m_{p2}r_{c2}^{\,2}\dot{\theta}_{c2}^{\,2} \right) + \\
&\quad \tfrac{1}{2}\left(J_{s3}\dot{\theta}_{s3}^{\,2} + J_{r2}\dot{\theta}_{r3}^{\,2} + 3J_{p3}\dot{\theta}_{p3}^{\,2} + 3m_{p3}r_{c3}^{\,2}\dot{\theta}_{c3}^{\,2} \right) \\
V &= 0 \\
F &= \tfrac{1}{2}\left(b_{s1}\dot{\theta}_{s1}^{\,2} + b_{c1}\dot{\theta}_{c1}^{\,2} + 3b_{p1}\dot{\theta}_{p1}^{\,2} + b_{r1}\dot{\theta}_{r1}^{\,2} \right) + \\
&\quad \tfrac{1}{2}\left(b_{s2}\dot{\theta}_{s2}^{\,2} + b_{c2}\dot{\theta}_{c2}^{\,2} + 3b_{p2}\dot{\theta}_{p2}^{\,2} + b_{r2}\dot{\theta}_{r2}^{\,2} \right) + \\
&\quad \tfrac{1}{2}\left(b_{s3}\dot{\theta}_{s3}^{\,2} + b_{c3}\dot{\theta}_{c3}^{\,2} + 3b_{p3}\dot{\theta}_{p3}^{\,2} + b_{r3}\dot{\theta}_{r3}^{\,2} \right)
\end{aligned}
\qquad (2.25)
$$

Due to the fact that all three sun gears are rigidly connected, and the carrier is rigidly connected with the next adjacent ring gear, the following kinematic constraints result:

$$
\begin{aligned}
&\dot{\theta}_{s1} = \dot{\theta}_{s2} = \dot{\theta}_{s3} \equiv \dot{\theta}_{s} \\
&\dot{\theta}_{c1} = \dot{\theta}_{r2} \\
&\dot{\theta}_{c2} = \dot{\theta}_{r3}
\end{aligned}
\qquad \textbf{(2.26)}
$$

Substitution of Equation **(2.26)** into Equation (2.25) produces the simplified energy term expression given as:

$$
\begin{aligned}
K &= \tfrac{1}{2}\left(J_{s1}\dot{\theta}_s^{\,2} + J_{r1}\dot{\theta}_{r1}^{\,2} + 3J_{p1}\dot{\theta}_{p1}^{\,2} + 3m_{p1}r_{c1}^{\,2}\dot{\theta}_{r2}^{\,2} \right) + \\
&\quad \tfrac{1}{2}\left(J_{s2}\dot{\theta}_s^{\,2} + J_{r2}\dot{\theta}_{r2}^{\,2} + 3J_{p2}\dot{\theta}_{p2}^{\,2} + 3m_{p2}r_{c2}^{\,2}\dot{\theta}_{r3}^{\,2} \right) + \\
&\quad \tfrac{1}{2}\left(J_{s3}\dot{\theta}_s^{\,2} + J_{r2}\dot{\theta}_{r3}^{\,2} + 3J_{p3}\dot{\theta}_{p3}^{\,2} + 3m_{p3}r_{c3}^{\,2}\dot{\theta}_{c3}^{\,2} \right) \\
V &= 0 \\
F &= \tfrac{1}{2}\left(b_{s1}\dot{\theta}_s^{\,2} + b_{c1}\dot{\theta}_{r2}^{\,2} + 3b_{p1}\dot{\theta}_{p1}^{\,2} + b_{r1}\dot{\theta}_{r1}^{\,2} \right) + \\
&\quad \tfrac{1}{2}\left(b_{s2}\dot{\theta}_s^{\,2} + b_{c2}\dot{\theta}_{r3}^{\,2} + 3b_{p2}\dot{\theta}_{p2}^{\,2} + b_{r2}\dot{\theta}_{r2}^{\,2} \right) + \\
&\quad \tfrac{1}{2}\left(b_{s3}\dot{\theta}_s^{\,2} + b_{c3}\dot{\theta}_{c3}^{\,2} + 3b_{p3}\dot{\theta}_{p3}^{\,2} + b_{r3}\dot{\theta}_{r3}^{\,2} \right)
\end{aligned}
\qquad (2.27)
$$

By using the same argument as in the single planetary gear modeling, the dissipation energy can be neglected, and a single equivalent dissipation term can be added in the final dynamic equation afterwards.

Equation (2.25) can be further simplified as shown as:

$$
\begin{aligned}
K &= \tfrac{1}{2}\left(\left(J_{s1} + J_{s2} + J_{s3} \right)\dot{\theta}_s^{\,2} + J_{r1}\dot{\theta}_{r1}^{\,2} + 3J_{p1}\dot{\theta}_{p1}^{\,2} + \left(J_{r2} + 3m_{p1}r_{c1}^{\,2} \right)\dot{\theta}_{r2}^{\,2} \right) + \\
&\quad \tfrac{1}{2}\left(3J_{p2}\dot{\theta}_{p2}^{\,2} + \left(J_{r3} + 3m_{p2}r_{c2}^{\,2} \right)\dot{\theta}_{r3}^{\,2} \right) + \\
&\quad \tfrac{1}{2}\left(3J_{p3}\dot{\theta}_{p3}^{\,2} + 3m_{p3}r_{c3}^{\,2}\dot{\theta}_{c3}^{\,2} \right) \\
V &= 0, F = 0
\end{aligned}
\qquad (2.28)
$$

The constraints of the DVT system can be shown as:

$$
\begin{aligned}
\dot{\theta}_s r_{s1} - \dot{\theta}_{r2}\left(r_{c1} - r_{p1} \right) + \dot{\theta}_{p1} r_{p1} &= 0 \\
-\dot{\theta}_s r_{s1} + \dot{\theta}_{r2}\left(r_{r1} + r_{s1} \right) - \dot{\theta}_{r1} r_{r1} &= 0 \\
\dot{\theta}_s r_{s2} - \dot{\theta}_{r3}\left(r_{c2} - r_{p2} \right) + \dot{\theta}_{p2} r_{p2} &= 0 \\
-\dot{\theta}_s r_{s2} + \dot{\theta}_{r3}\left(r_{r2} + r_{s2} \right) - \dot{\theta}_{r2} r_{r2} &= 0 \\
\dot{\theta}_s r_{s3} - \dot{\theta}_{c3}\left(r_{c3} - r_{p3} \right) + \dot{\theta}_{p3} r_{p3} &= 0 \\
-\dot{\theta}_s r_{s3} + \dot{\theta}_{c3}\left(r_{r3} + r_{s3} \right) - \dot{\theta}_{r3} r_{r3} &= 0
\end{aligned}
\qquad (2.29)
$$

The same relationships exist among the angular positions and among the angular accelerations as well. The general coordinates and general forces are:

$$
\begin{aligned}
&q_1 \equiv \theta_s,\ q_2 \equiv \theta_{p1}, q_3 \equiv \theta_{r2},\ \ q_4 \equiv \theta_{r1},\ \ q_5 \equiv \theta_{p2}, q_6 \equiv \theta_{r3},\ \ q_7 \equiv \theta_{p3}, q_4 \equiv \theta_{c3} \\
&Q_1 \equiv T_{in}, Q_2 \equiv 0,\ Q_3 \equiv -T_{b2}, Q_4 \equiv -T_{b1}, Q_5 \equiv 0,\ \ Q_6 \equiv -T_{b3}, Q_7 \equiv 0,\ \ Q_8 \equiv -T_{out}
\end{aligned}
\qquad (2.30)
$$

Substitution of Equation (2.28), Equation (2.29), and Equation (2.30) into Equation (2.15) gives the overall form dynamic equations of the DVT as follows:

$$
\begin{aligned}
&J_s\ddot{\theta}_s = T_{in} + r_{s1}\lambda_1 - r_{s1}\lambda_2 + r_{s2}\lambda_3 - r_{s2}\lambda_4 + r_{s3}\lambda_5 - r_{s3}\lambda_6 \\
&3J_{p1}\ddot{\theta}_{p1} = r_{p1}\lambda_1 \\
&\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} = -T_{b2} - \left(r_{c1} - r_{p1}\right)\lambda_1 + \left(r_{s1} + r_{r1}\right)\lambda_2 - r_{r2}\lambda_4 \\
&J_{r1}\ddot{\theta}_{r1} = -T_{b1} - r_{r1}\lambda_2 \\
&3J_{p2}\ddot{\theta}_{p2} = r_{p2}\lambda_3 \\
&\left(3m_{p2}r_{c2}^2 + J_{c2} + J_{r3}\right)\ddot{\theta}_{r3} = -T_{b3} - \left(r_{c2} - r_{p2}\right)\lambda_3 + \left(r_{s2} + r_{r2}\right)\lambda_4 - r_{r3}\lambda_6 \\
&3J_{p3}\ddot{\theta}_{p3} = r_{p3}\lambda_5 \\
&\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} = -T_{out} - \left(r_{c3} - r_{p3}\right)\lambda_5 + \left(r_{s3} + r_{r3}\right)\lambda_6
\end{aligned}
\qquad (2.31)
$$

Now, let's derive the dynamic model for each of the three DVT gear ratios. When the DVT is in the 3rd gear, which is equivalent to locking the 1st ring gear, and constraints are given as:

$$
\begin{aligned}
&\ddot{\theta}_{r1} = 0 \\
&T_{b2} = T_{b3} = 0
\end{aligned}
\qquad (2.32)
$$

When Equation (2.32) is substituted into Equation (2.31), the dynamic equations for the DVT with the first ring gear locked are then given by:

$$
\begin{aligned}
&J_s\ddot{\theta}_s = T_{in} + r_{s1}\lambda_1 - r_{s1}\lambda_2 + r_{s2}\lambda_3 - r_{s2}\lambda_4 + r_{s3}\lambda_5 - r_{s3}\lambda_6 \\
&\frac{3J_{p1}}{r_{p1}}\ddot{\theta}_{p1} = \lambda_1, \frac{3J_{p3}}{r_{p3}}\ddot{\theta}_{p3} = \lambda_5 \\
&\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} = -\left(r_{c1} - r_{p1}\right)\lambda_1 + \left(r_{s1} + r_{r1}\right)\lambda_2 - r_{r2}\lambda_4 \\
&-\frac{T_{b1}}{r_{r1}} = \lambda_2, \frac{3J_{p2}}{r_{p2}}\ddot{\theta}_{p2} = \lambda_3 \\
&\left(3m_{p2}r_{c2}^2 + J_{c2} + J_{r3}\right)\ddot{\theta}_{r3} = -\left(r_{c2} - r_{p2}\right)\lambda_3 + \left(r_{s2} + r_{r2}\right)\lambda_4 - r_{r3}\lambda_6 \\
&\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} = -T_{out} - \left(r_{c3} - r_{p3}\right)\lambda_5 + \left(r_{s3} + r_{r3}\right)\lambda_6
\end{aligned}
\qquad (2.33)
$$

The final form of the dynamic equations for the DVT in 3rd gear is a second order ordinary differential equations (Equation (2.34)). Notice that a dissipation term has been added.

$$J_1\ddot{\theta}_s + b_1\dot{\theta}_s = T_{in} - \gamma_1 T_{out}$$

where,

$$\gamma_1 \equiv \frac{\left(r_{s3} + \left(\frac{r_{s1} r_{r2}}{(r_{r1}+r_{s1})(r_{r2}+r_{s2})} + \frac{r_{s2}}{(r_{r2}+r_{s2})} \right) r_{r3} \right)}{(r_{r3}+r_{s3})}$$

$$J_1 \equiv \left(-p_1 p_2 J_s + (1-p_1 p_2)\alpha_1 J_{eq2} + 3p_1\mu_1(\beta_1 - 1)\mu_1 J_{p1} + 3\mu_2(\alpha_1 - 1)\mu_2 J_{p2} + \right. \quad (2.34)$$

$$\left. 3p_3\mu_3(\gamma_1 - 1)m_3 J_{p3} + p_2(1-p_1)\beta_1 J_{eq1} + (p_3 - p_1 p_2)\gamma_1 J_{eq3}\right)/(-p_1 p_2)$$

$$p_1 \equiv \frac{(r_{s1}+r_{r1})}{r_{r1}},\ p_2 \equiv \frac{(r_{s2}+r_{r2})}{r_{r2}},\ p_3 \equiv \frac{r_{r3}}{(r_{s3}+r_{r3})},\ \mu_1 \equiv \frac{r_{s1}}{r_{p1}},\ \mu_2 \equiv \frac{r_{s2}}{r_{p2}},\ \mu_3 \equiv \frac{r_{s3}}{r_{p3}}$$

$b_1 -$ equivalent coefficient of friction

$$J_{eq1} \equiv 3m_{p1}r_{c1}^2 + J_{c1} + J_{r2},\ J_{eq2} \equiv 3m_{p2}r_{c2}^2 + J_{c2} + J_{r3},\ J_{eq3} \equiv 3m_{p3}r_{c3}^2 + J_{c3}$$

Similarly, the dynamic equation of the DVT in the 2nd gear is given by:

$$J_2\ddot{\theta}_s + b_2\dot{\theta}_s = T_{in} - \gamma_2 T_{out},\ where$$

$$\gamma_2 \equiv \frac{r_{s2}}{(r_{r2}+r_{s2})}\frac{r_{r3}}{(r_{r3}+r_{s3})} + \frac{r_{s3}}{(r_{r3}+r_{s3})},$$

$$J_2 \equiv J_s + (1-p_1)\beta_2 J_{r1} + \left(1 - \frac{1}{p_2}\right)\alpha_2 J_{eq2} + \left(1 - \frac{p_3}{p_2}\right)\gamma_2 J_{eq3} + \quad (2.35)$$

$$3\mu_1^{\,2} J_{p1} - \frac{3}{p_2}\mu_2^{\,2}(\alpha_2 - 1)J_{p2} - 3\frac{p_3}{p_2}\mu_3^{\,2}(\gamma_2 - 1)J_{p3}$$

$b_2 -$ equivalent coefficient of friction

And the dynamic equation of the DVT in the 1st gear is given by:

$$J_3\ddot{\theta}_s + b_3\dot{\theta}_s = T_{in} - \gamma_3 T_{out},\ where,$$

$$\gamma_3 \equiv \frac{r_{s3}}{(r_{r3}+r_{s3})},\alpha_3 \equiv -\frac{r_{s2}}{r_{r2}},b_3 - \text{equivalent coefficient of friction} \quad (2.36)$$

$$J_3 \equiv J_s + (1-p_1p_2)(\alpha_3 p_1 + \beta_2)J_{r1} + (1-p_2)\alpha_3 J_{eq1} + (1-p_3)\gamma_3 J_{eq3} -$$

$$3p_1\mu_1^{\,2}(\alpha_3 - 1)J_{p1} + 3\mu_2^{\,2}J_{p2} - 3p_3\mu_3^{\,2}(\gamma_3 - 1)J_{p3}$$

Equations (2.34), (2.35) and (2.36) can be expressed in a unified form as:

$$\dot{x}(t) = f_i\left(x(t), T_{in}, T_{out}\right),\ (i = 1, 2, 3),$$
where, (2.37)
$$x(t) = \begin{Bmatrix} \theta_s(t) \\ \dot{\theta}_s(t) \end{Bmatrix}$$

Since at any time, only one brake is engaged, only one of the three dynamic equations is the governing equation. For example, when DVT is in 3rd gear with the first brake (B1 in Figure 1.14) engaged, the DVT system's motion is governed by Equation (2.34), other two dynamic equations as Equation (2.35) and (2.36) are in an inactive state. By engaging a different brake, the governing dynamic equation of the DVT changes accordingly. The three brakes are controlled by a high level gear shift control strategy, or supervisory control strategy, which will be discussed in detail in Chapter 3. The addition of this supervisory action to the dynamic model as given in Equation (2.37) allows a unified form of the DVT dynamic equation to be:

$$\dot{x}(t) = f\left(x(t), \delta\left(t, x(t)\right), T_{in}, T_{out}\right) \qquad (2.38)$$

Equation (2.38) is an exact form of a hybrid dynamic system. Hence, this DVT model is a hybrid dynamic system with three subsystems that can be individually expressed as 2nd order ODEs.

A Robot Arm Model

A prototype robot manipulator, the Titan II manufactured by Schilling Robotics, is used as the design baseline. The robot manipulation can be

modeled as a rigid robot arm with a simple support on one end and a load attached on the other, as shown in Figure 2.5.

The dynamic equation is given as:

$$T = (M_p + \tfrac{1}{3}M_a)l^2\ddot{\theta} + (M_p + \tfrac{1}{2}M_a)lg\cos\theta + b_a\dot{\theta} \quad (2.39)$$

where

T – torque input, b_a – friction

M_a – arm mass , M_p – load mass

l – arm length, g – gravity constant,

θ – angular position with respect to horizontal orientation

This is a nonlinear system, and the output feedback linearization method was used to generate a linear model as will be discussed in the following section.

The TBA Prototype Model

The TBA model can be derived based on the dynamic equations of the three modules: the BLDC servo, the DVT and the robot arm. Since the DVT is a hybrid system, the TBA system as a whole is also a hybrid system, which includes three continuous dynamic systems with discrete state changes. When no state change occurs, the continuous dynamic system models of different modules as given in Equation (2.9), (2.34), (2.35), (2.36) and (2.39) can be summarized as:

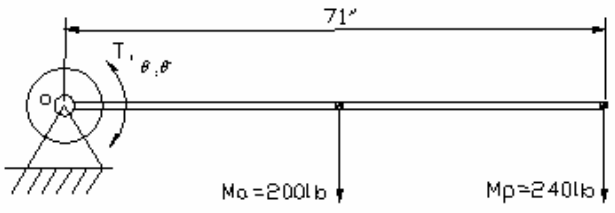


Figure 2.5, A prototype robot arm

$$
\begin{aligned}
&J\frac{d\omega_R}{dt} = K_t i - b_m\omega_R - T_{load} \\
&\frac{d\theta_R}{dt} = \omega_R \\
&J_i\ddot{\theta}_s + b_i\dot{\theta}_s = T_{in} - \gamma_i T_{out} \\
&T = (M_p + \tfrac{1}{3}M_a)l^2\ddot{\theta}_a + (M_p + \tfrac{1}{2}M_a)lg\cos\theta_a + b_a\dot{\theta}_a
\end{aligned}
\qquad (2.40)
$$

The motor shaft is rigidly attached to the DVT input shaft, and the output shaft is rigidly attached to the robot arm through a fixed gear reduction 200 to 1. This final gear is not included in the TBA prototype, but in fact would be present in the robot shoulder pitch axis design. The final gear box ratio is chosen based on the load and speed requirements of the TBA and the robot joint [1]. For a final gear ratio of 200, the following relations hold:

$$
\begin{aligned}
&\dot{\theta}_m = \dot{\theta}_R = \omega_R = \dot{\theta}_s = 200\,\dot{\theta}_a / \gamma_i \\
&\ddot{\theta}_m = \ddot{\theta}_s = 200\,\ddot{\theta}_a / \gamma_i \\
&\theta_m = \theta_s = 200\theta_a / \gamma_i \\
&T_{load} = T_{in} \\
&T_{out} = T / 200
\end{aligned}
\qquad (2.41)
$$

Thus, Equation (2.40) can be simplified into the following form:

$$
\left(J + J_i + \gamma_i\frac{(M_p + \frac{1}{3}M_a)l^2}{200}\right)\ddot{\theta} + \left(b_m + b_i + \frac{\gamma_i}{200}b_a\right)\dot{\theta} = K_t i - \frac{(M_p + \frac{1}{2}M_a)lg}{200}\cos\left(\frac{\gamma_i}{200}\theta\right)\gamma_i \qquad (2.42)
$$

If the mass of the load and the arm, the length of the arm, and the joint position can be measured with high accuracy, Equation (2.42) can be simplified as a linear model by using the feedback linearization [6-8]. Let

$$
u_i(t) = K_t i(t) - \frac{(M_p + \frac{1}{2}M_a)lg}{200}\gamma_i\left(\cos\left(\frac{\gamma_i}{200}\theta(t)\right) + \frac{\theta(t)}{2\pi}\right) \qquad (2.43)
$$

The robot arm angular position is in fact limited within the range of $[0, 2\pi]$.

A prerequisite condition for a feedback linearization to be valid is that the control input cannot reach the physical limitation. In this project, the control signal is the current of the BLDC servo, as shown in Equation (2.43). The feedback is adequately scaled down to avoid the saturation as shown in Equation (2.43). With the new feedback controller, Equation (2.42) can be further simplified as:

$$
\begin{aligned}
&J_{Ti}\ddot{\theta}(t)+B_{Ti}\dot{\theta}(t)+M_i\theta(t)=u_i(t)\\
&\textit{where}\\
&J_{Ti}\equiv J+J_i+\frac{\gamma_i(M_p+\frac{1}{3}M_a)l^2}{200}\\
&B_{Ti}\equiv b_m+b_i+\frac{\gamma_i}{200}b_a\\
&M_i\equiv\frac{(M_p+\frac{1}{2}M_a)lg\gamma_i}{400\pi}
\end{aligned}
\qquad (2.44)
$$

The corresponding state space form is:

$$
\begin{aligned}
&\dot{x}(t)=A_i x(t)+B_i u_i(t)\\
&y(t)=C_i x(t)\\
&\textit{where,}\\
&x_1(t)\equiv\theta(t)=\theta_s(t),\ x_2(t)\equiv\dot{\theta}(t)=\dot{\theta}_s(t),\\
&y(t)\equiv\dot{\theta}_a(t)\\
&A_i\equiv\begin{bmatrix}0 & 1\\ -\frac{M_i}{J_{Ti}} & -\frac{B_{Ti}}{J_{Ti}}\end{bmatrix},B_i\equiv\begin{Bmatrix}0\\1\end{Bmatrix},C_i\equiv\begin{bmatrix}\gamma_i & 0\end{bmatrix}\\
&\dot{x}(t)\equiv\begin{Bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{Bmatrix},x(t)\equiv\begin{Bmatrix}x_1(t)\\ x_2(t)\end{Bmatrix}
\end{aligned}
\qquad (2.45)
$$

Again, this can fit in an hybrid dynamic system expression as:

$$
\begin{aligned}
&\dot{x}(t)=f\left(x(t),\delta\left(t,x(t)\right),u(t)\right)\\
&\textit{where}\\
&x(t)-2\times1\text{ continuous state vector}\\
&\delta\left(t,x(t)\right)-\text{ a discrete state variable}\\
&u(t)-\text{ a scalar control input}
\end{aligned}
\qquad (2.46)
$$

In summary, the TBA model can be simplified into a hybrid system with three subsystems, which can be modeled as a 2^{nd} order ODE given by Equation (2.45) or in a more compact form as shown in Equation (2.46).

CHAPTER 3

TBA Controls

Background

The TBA prototype system is a hybrid dynamic system, which is composed of three “switchable” continuous dynamic systems. A high level supervisory controller is used for gear shift control to achieve a discrete state change. In general, the continuous dynamic subsystems can be linear or nonlinear systems. In this research, only the linear subsystems are considered since the TBA model can be linearized as discussed in Chapter 2.

The state space model for a general hybrid system involving linear continuous dynamic subsystems is given as:

$$\begin{aligned} \dot{x}(t) &= A_i x(t) + B_i u_i(t) \\ y(t) &= C_i x(t) \end{aligned} \qquad (3.1)$$

As shown in Chapter 2, B_i is same for all three states of the TBA system.

The control problem of a hybrid system has drawn great attention in the past decade [9-19]. For a hybrid system, there are three requirements for the controller:

1. The hybrid system must be stable.
2. The switch strategy must be transition stable.
3. Performance requirements must be satisfied.

In this research, the first two requirements are analyzed in detail, and the third requirement is obtained by a feedback controller.

In a continuous dynamic control system, when systems are unstable, various techniques can be used to make them stable, such as PID, feedback control, etc. In a hybrid system, because of the discrete state change, the system stability does not solely depend on the subsystem stability. Two important situations exist: on one hand, the state changes could potentially make a stable system unstable with an inappropriate shift [15], while on the other hand, a proper state change could stabilize a hybrid system involving unstable continuous subsystems.

A commonly used method to verify the stability of a shift system is the Lyapunov function based method [14, 18, 20-24]. A Linear Matrix Inequality technique has also been used to analyze the stability of a hybrid system [17] [25].

Another issue associated with a hybrid system is the transient disturbance due to the state change.

Supervisory control has been used to synchronize gear shifts vs. engine speed to achieve a smooth shift transition [26]. An optimization method has been used on an automotive gear box to achieve some optimal control index [27]. Another optimal control method based on linear quadratic optimization, so called “bumpless transfer”, has been used in a helicopter control to achieve smooth control signal transition under state change [28]. An optimization based method has been also used to solve a system with a varying sample rate [29].

Other techniques, like robust control and dynamic programming, have also been used for specific hybrid systems [1, 18, 19, 22, 30].

By constructing a completely discrete abstraction using hybrid automata theory or other pure logic based theories [31-33], various techniques have been developed to study the hybrid system for stability, reachability and controller synthesis based on automata theory on a discretized hybrid system.

From a continuous dynamic system point of view, a general hybrid dynamic closed loop control system can be represented by Figure 3.1. This diagram can be divided into three major blocks: a hybrid plant, a supervisory controller, and a closed loop controller.

The hybrid plant is the system to be controlled; it has continuous dynamics and discrete state changes. Some examples of uses of these types of hybrid plants are: a furnace oven to maintain a constant temperature by turning on and off the heaters, a chemical reactor container to keep the right amount of reactants by turning on and off the valves, and, in this research, a TBA to match the load requirement by shifting gears. A hybrid system block accepts inputs from the controllers and generates outputs to be used as the input to the event generator block.

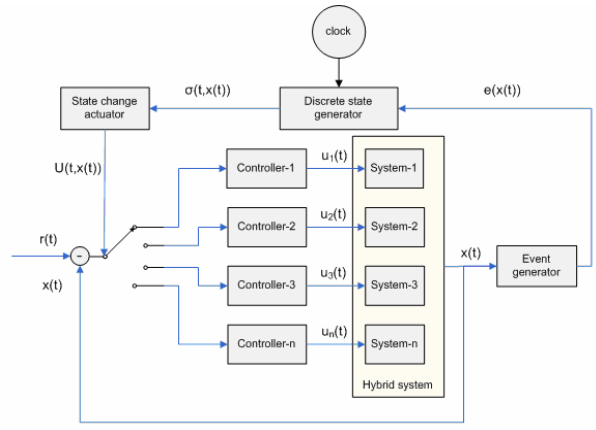


Figure 3.1, A typical hybrid dynamic system control block diagram

The supervisory controller block can be divided into three sections: an event generator, a discrete state generator, and a state change actuator. The event generator takes the input from the hybrid system and generates meaningful event signals to the discrete state generator. In the oven example, the event generator takes the temperature measurement as input and generates a high, normal or low temperature event as output. In the chemical reactor, it takes the amount of reactant measurement as input, and generates a more, normal, or less reactant event. In the TBA, the event generator takes the position and torque measurements as inputs and generates a switch event when the system trajectory crosses a certain switch boundary as discussed later in the chapter. In the above examples, only one event is generated at a time, but in more complicated systems, multiple events can be generated.

The discrete state generator takes three inputs: the events generated by the event generator, a timer input, and the current system state. Then it uses an intelligent procedure to determine what the new state of the system will be, and to generate state control actions. This block is a pure discrete system, so all of the discrete theory can be readily used for its analysis. In the oven example, the discrete state generator generates a command to turn on or off the heater based on the events from the event generator. In the chemical reactor example, it generates a command to turn on or off the valves based on the event generator. In the TBA, the function of this block is a little more complex; it takes three inputs: the event, the timer, and the current state. Multiple actions are generated since the engagement of a gear requires the disengagement of a currently engaged gear.

The output of the discrete state generator includes the disengagement of the current gear, engagement of the new gear, and no change in the other gear state. In summary, the state change generator generates all the actions associated with a certain state change and sends its output to the state change actuator block.

The state change actuator takes the output of the discrete state generator block as commands and translates them into meaningful control signals to fulfill the state change action. The commonly used control signals can be either analog or digital. For example, in the chemical reactor example, if a control valve is the actuator, an analog signal is used to control the amount the valve is opened; if it is an on-off type valve, a simple digital signal is enough.

In the TBA example, three digital signals are used to control the braking actions. Specifically, when the digital line voltage is low, the associated brake motor is idle, and no torque is applied to the brake band; when it is high, the brake motor is activated, and the prescribed torque is applied to the brake band so that the corresponding ring gear is stopped. The closed loop controller is used to improve system stability and performance including during state changes. In this research, for each individual state, linear system based theory can be used since the TBA system can be linearized as shown in Chapter 2.

In general, three basic problems of hybrid dynamic system control are formulated as follows [15]:

Problem 1. Find conditions that guarantee that the switched system is asymptotically stable for any switched signals.

Problem 2. Describe those classes of switched signals for which the switched system is asymptotically stable.

Problem 3. Construct a switching signal for which the switched system is asymptotically stable.

These three problems deal with different types of hybrid systems, and the focuses of the problems are different.

Problem 1 generally deals with a hybrid dynamic system with continuous dynamics that are asymptotically stable by open loop or closed loop control. The focus is to find a condition such that the hybrid system is guaranteed to be asymptotically stable for all switched signals. The results of Problem 1 can be found in many publications, and many hybrid dynamic system problems have this property. A commonly used method is to find a common Lyapunov function. A gradient based technique to find the common Lyapunov function is presented in [20, 34]. In this research, Problem 1 is applicable and is first investigated, for the purpose of examining the conditions under which the TBA is guaranteed to be stable for any gear shift sequence.

Problem 2 deals with a larger range of hybrid systems which cannot be stabilized by arbitrary switch signals. This raises the following questions:

1. Is there a specific switch signal sequence that can stabilize the system?
2. How can such a shifting sequence be found?
3. Given a switch signal sequence, is the hybrid system stable?

The first question is an analysis problem. Different Lyapunov-based methods have been used to find an answer to the first question [12, 19-21, 23,

24, 35]. The second question relates to design. A technique to design such a switch sequence is reported in [35]. The third question relates to verification, and it is valid when a class of predefined switch signals is available for analysis. A trial and error method is generally used to find a suitable switch sequence from the predefined signals. The predefined class is obtained based on a good understanding of the specific hybrid system.

Problem 2 can be divided into the following categories: a) The continuous dynamic systems are stable individually; b) The continuous dynamic systems are not stable individually. Because the hybrid system stability and the stability of its continuous subsystems are independent [12], the subsystem stability is neither sufficient nor a necessary condition of the hybrid system stability.

One interesting example is that certain switch signals can make a hybrid system with stable continuous subsystems unstable. In this case, only proper switch signals can make the hybrid system stable.

Let's look at an example. Suppose a hybrid system is composed of two stable second order ODEs, and further assume the trajectory or phase portraits of the subsystems are shown as the top two curves in Figure 3.2.

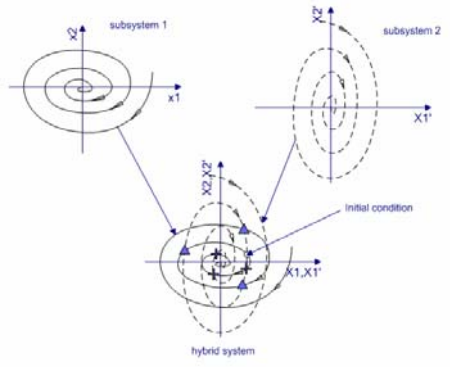


Figure 3.2, Unstable hybrid system with stable dynamic subsystems

The trajectory of the hybrid system is shown as the bottom curve in Figure 3.2 by overlaying the two subsystems trajectories. This example does not give any specific mathematic equations for the two subsystems; instead, it only intends to show that inappropriate state changes can make hybrid systems unstable even though the continuous subsystems are stable.

The first observation from Figure 3.2 is that the two continuous dynamic subsystems are stable, with subsystem 1 on the left and subsystem 2 on the right.

On the bottom overlaying hybrid system trajectory in Figure 3.2, the circle stands for the system initial condition, which is put on the intersection of the two subsystem trajectories. Suppose subsystem 2 is initially active, if there is no switch signal, the phase portrait of the hybrid system will be the same as that of the subsystem 2. The hybrid system is stable, since the hybrid system is essentially the continuous dynamic subsystem 2.

If the switch signals happen at the locations marked with triangles as shown in the bottom figure, the state variables of the system quickly grow, and the system is said to be unstable. A common property of these switch signals is that they all happen at locations where, after each switch, the system trajectory is further away from the origin.

If the switch signals happen at the locations marked with crosses, the system is stable since the state variables quickly approach the origin from the initial condition. Compared with the unstable system, the stable system has all switches occurred at locations where, after the switch, the trajectory is closer to the origin.

Based on the above analysis, in order to obtain a stable hybrid system, a proper switch signal must be used even the subsystems are stable. A common method to evaluate a switch signal is the energy based method, such as a quadratic form Lyapunov function [24].

The second category is an open question. The necessary and sufficient condition for the existence of a switch signal that stabilizes this type of hybrid system is only proven for certain cases. A necessary and sufficient condition for a hybrid system with multiple 2^{nd} order LTI systems is established in [24], and a sufficient condition for a simpler system is reported in [23].

Problem 3 is a design problem, which is the most challenging problem among the three basic problems for hybrid systems. In general, even though significant results have been reported in [9-11, 17-19, 22, 25, 27], this problem is essentially an open problem. Specific solutions can be found for certain applications. Different hybrid system simulation frameworks have been established to analyze hybrid systems. Hybrid automata based discrete abstractions of hybrid systems are reported in [9, 10, 31, 36]; linear system theory, affine system theory, and optimal control theory based analyses are reported in [11, 12, 22, 37, 38].

The remainder of this chapter will first discuss issues related with the supervisory controller. Then, a stability analysis in terms of hybrid system stability and switch signal transition stability will be given for the TBA system. In the last part, closed loop controller design methods will be analyzed. Finally, a suitable TBA control method which guarantees system stability and switch transition stability will be given.

Supervisory Control

The study of the hybrid system is essential in designing supervisory controllers for a continuous system [36]. The essential issue in designing a supervisory controller is to design an appropriate switch strategy. The following are necessary components in the evaluation of a switch strategy

1. Switch signal transition stability
2. Hybrid system stability

The first component is the evaluation of the stability of the switch signals. Basically, one must find conditions such that the switch signal is invariant under disturbances. Due to the absence of a general solution, this research adopts a heuristic approach in order to evaluate the transition stability of the switching signals. The second component falls into one of the three problems of a hybrid system. In this research, stability analysis is developed by attacking Problems 1 and 2 as presented earlier in this chapter, and the goal is to find the switch signals such that the TBA system is stable.

Switch Strategy- Transition Stability

The switch strategy controls the ways in which the switch signals are generated. Switch signals can be divided into three categories: time based, state variable based (or simply state based) and hybrid. Other categorization methods are possible; for example, in [19], the switch signals are divided into time based and event based. The following definitions are useful for development of the supervisory controller.

Definition 3.1: Time-based switch signal [19]

Time-based switch signal - A switch signal is called a time based switch signal if it is a function of time:

$$\delta(t) = \Delta(t) \qquad (3.2)$$

This type of shift signal is usually adopted when all the dynamics and control signals are known a priori, such that the states of the system can be calculated beforehand. Generally speaking, systems that can be controlled in an open loop scheme can use this type of switch signal.

In fact, a time based switch law can be treated as one of the other two categories. It is separated because it is the simplest switch law, and it is always transition stable. But the real world applications of a time based switch law are limited, since it is very difficult to find a system whose state variables only depend on time. So, strictly speaking, a time based switch signal can only achieve the least approximate result among the three.

Definition 3.2: State-based switch signal

State-based switch signal - A switch signal is state-based if it only depends on the state variables of the continuous subsystems, the first derivative of these state variables and/or the output of the system. The time variable does not explicitly appear in the state based switch signal function. The state based switch signal can be shown as:

$$\delta(t) = \Delta\left(x(t), \dot{x}(t), y(t)\right) \qquad (3.3)$$

This type of switch law is the widely used in the literature, and it works well in many real applications. The three examples introduced early in this chapter all

use state based switch signals. There is one limitation; the state based switch law is not guaranteed to be transition stable. The reason is that the variations in system state variables might generate unwanted switch signals, especially near the switch boundaries.

Definition 3.3: Hybrid switch signal

Hybrid switch signal - If the shift signal is based on both time and state of the system, it is called hybrid switch signal. The function of the switch signal can be represented as:

$$\delta(t) = \Delta\left(t, x(t), y(t), \dot{x}(t)\right) \qquad (3.4)$$

A broader definition of the hybrid switch signal is: any switch signal that does not fit the first two definitions.

As we can see, the hybrid switch law definition implies the first two definitions and, thus, can be treated as a general form of a switch signal.

Properties of a switch signal:

Time invariant:

A switch signal function of the type shown in Equation (3.4) is said to be time invariant in the specified ranges if the following condition is satisfied.

There exists a $T > 0$,

$x(t), x(t+T) \in [x_l, x_h]$,

$y(t), y(t+T) \in [y_l, y_h]$, *and*

$\dot{x}(t), \dot{x}(t+T) \in [\dot{x}_l, \dot{x}_h]$

such that

$$\Delta\left(t+T, x(t+T), y(t+T), \dot{x}(t+T)\right) = \Delta\left(t, x(t), y(t), \dot{x}(t)\right)$$

This property defines an invariant set for the ideal dynamic system model.

State space invariant:

A switch signal of the type shown in Equation (3.4) is said to be state space invariant in the specified ranges if the following conditions are satisfied:

There exists

$\|\Delta x(t)\| > 0,\ \|\Delta y(t)\| > 0, \|\Delta \dot{x}(t)\| > 0$

$x(t), x(t) + \Delta x(t) \in [x_1, x_2],$

$y(t), y(t) + \Delta y(t) \in [y_1, y_2],$ *and*

$\dot{x}(t), \dot{x}(t) + \Delta \dot{x}(t) \in [\dot{x}_1, \dot{x}_2]$

such that

$\Delta(t, x(t) + \Delta x(t), y(t) + \Delta y(t), \dot{x}(t) + \Delta \dot{x}(t)) = \Delta(t, x(t), y(t), \dot{x}(t))$

This property defines an invariant set for a dynamic system under state perturbation.

Theorem 3.1:

A switch signal of the form as shown in Equation (3.4) is said to be transition stable on certain ranges of the time and state variables if it is both time invariant and state space invariant on the same ranges.

Proof: The proof of this theorem is trivial since there are only two factors that will cause a switch signal as defined in Equation (3.4) to be transition unstable. These two factors are the time variable and the state space variables, so if the signal is both time invariant and state space invariant, it is guaranteed to be invariant on the specified time and state ranges; in other words, it is transition stable. □

Transition stability is very important for the implementation of a switch signal, or switch strategy for the TBA system.

TBA switch signals design

The TBA system involves three subsystems governed by 2nd order ODE, and the two state variables of the TBA are position and velocity. This section analyzes switch signal designs for all three types of switch signals, and, in the end, a suitable switch signal for the TBA is given. The TBA must shift gears to accommodate a varying load torque. In determining the shift strategy for the TBA system, a heuristic rule can be used as follows: when the torque is high, the TBA should shift to a lower gear with a higher ratio; otherwise, it should shift to a higher gear with a lower ratio. Using this heuristic rule, three different shift strategies were developed and tested; a time based strategy, a state space based strategy and a hybrid based strategy. The first method uses a time based shift signal. A time based switch signal is only valid with complete a priori knowledge of TBA system and the load properties. In this case, a time based switch signal is guaranteed to be transition stable. Even though uncertainty associated with the real trajectory and real load certainly exists, the time based switch signals can still have good performance if such uncertainty is within a reasonable range. Again, in order to use the time based switch signals in the TBA, the system parameters should be clearly defined, which generally requires a smooth trajectory design, an accurate load estimation method, and a simulation model. The second shift strategy is torque based. Since the equation for the torque, shown in Equation (2.39), depends on the state variables and their derivatives, this strategy can be treated as a state based switch signal strategy. Three variables are considered in the TBA shifting signal expression: position, velocity, and acceleration. So, the shift signal can be expressed as:

$$\delta(t) = \Delta\left(\theta(t), \dot{\theta}(t), \ddot{\theta}(t)\right) \qquad (3.5)$$

For simplicity, let's look at an example of a shift signal with only two state variables: angular position and velocity.

A two dimensional (2-D) state space trajectory with a shift boundary is shown in Figure 3.3. The shift boundary curve is shown as:

$$C_{sb} = \left\{x(t) : f_{sb}\left(x(t)\right) = 0, x(t) \in R^2\right\} \qquad (3.6)$$

The shift boundary curve divides the state space into two adjacent regions as shown as:

$$\begin{aligned} S_a &= \left\{x(t) : f_{sb}\left(x(t)\right) < 0, x(t) \in R^2\right\}, \\ S_b &= \left\{x(t) : f_{sb}\left(x(t)\right) > 0, x(t) \in R^2\right\} \end{aligned} \qquad (3.7)$$

Suppose the system trajectory moves upward along the trajectory curve as defined by:

$$C_{tr} = \left\{x(t) : \dot{x}(t) = f_i\left(\delta_i\left(t, x(t), y(t)\right), x(t)\right), x(t) \in R^2\right\} \qquad (3.8)$$

Further, suppose the trajectory intersects with the shift boundary at point P_s as shown in Figure 3.3. Before the intersection, the system is in state S_a with switch signal δ_a; after the intersection, the system enters a new state S_b with a new shift signal δ_b.

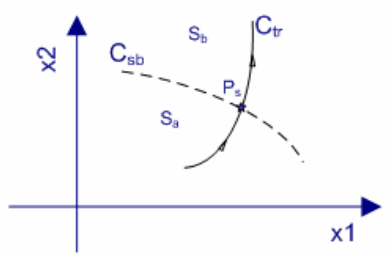


Figure 3.3, Two dimensional switch curve

Although the above switch strategy appears to be valid and convenient, it has a potential problem. The switch signal is not transition stable. To prove this, Theorem 3.2 and Theorem 3.3 are formulated.

Theorem 3.2:

A shift signal generated on a shift boundary as Equation (3.6), which divides the state space as Equation (3.7), is not state space invariant in the ranges including the shift boundary.

Proof:

Suppose a range is defined as follows.

$$\left\{x(t):\tau \le f_{sb}\left(x(t)\right) \le 0, \in R^2\right\}, \tau < 0$$

At the lower bound, the system is in state S_a with switch signal δ_a, after a state change $\left\|\Delta x(t)\right\| > 0$ towards the boundary. No matter how small $\left\|\Delta x(t)\right\|$ is, there exists a corresponding small τ, such that the system crosses the boundary and enters a new state S_a with a new switch signal δ_b. □

Theorem 3.3: A shift signal generated on shift boundary as in Equation (3.6), which divides the state space as Equation (3.7), is not time invariant on the ranges including the shift boundary, unless the system dwells on the boundary.

Proof: Suppose a range defined as follows,

$$\left\{x(t):\tau \le f_{sb}\left(x(t)\right) \le 0, \in R^2\right\}, \tau < 0$$

At the lower bound, the system is in state S_a with switch signal δ_a, after a time increase $T > 0$. Suppose the trajectory moves towards the boundary, and

the system does not dwell on the boundary. No matter how small T is, there exists a corresponding small τ, such that the system crosses the boundary and enters a new state S_b with a new switch signal δ_b.

The unstable transition actions of the switch signal near the switch boundary can be shown in Figure 3.4. Because the system trajectory has some perturbations in the vicinity of the shift boundary, the switch signal will change back and forth as the trajectory crosses the shift boundary from different directions. This phenomenon is often referred to as limit cycle behavior of the switch signal. In the TBA system, this will cause the brake to engage and disengage frequently, thus generating unwanted disturbances. A new shift strategy, illustrated in Figure 3.5, is designed to solve this problem.

In this new shift strategy, another shift boundary is added such that the state space is divided into three regions with two states. The states and switch signal for the TBA system are now defined as follows:

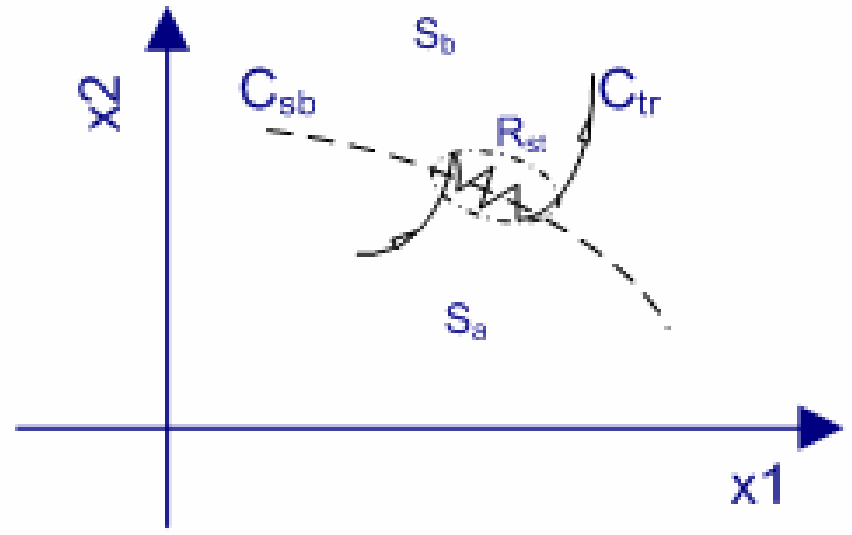


Figure 3.4, Two dimensional switch curve with disturbance

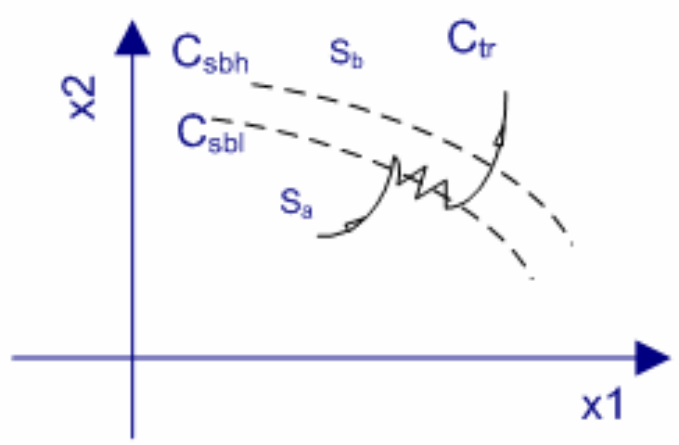


Figure 3.5, A shift strategy with two shift boundaries

Below the lower boundary C_{sbl}, system is in state S_a with switch signal δ_a. Above the upper boundary, system is in state S_b with switch signal δ_b. Between the two boundaries, the system state and switch signal are the same as those before the crossing of the boundaries; therefore, the system may have different states in this region determined by how the trajectory enters the region. This region is called a grey region. For example, if the trajectory crosses the lower boundary into the grey region, the state is S_a with switch signal δ_a; if the trajectory crosses the upper boundary into the grey region, the state is S_b with switch signal δ_b.

The two shift boundaries are defined by:

$$C_{sbl} = \left\{ x(t) : f_{sbl}\left(x(t)\right) = 0, x(t) \in R^2 \right\},$$
$$C_{sbh} = \left\{ x(t) : f_{sbh}\left(x(t)\right) = 0, x(t) \in R^2 \right\} \qquad (3.9)$$

A new switch signal is generated only if the trajectory enters the grey region by crossing one shift boundary and leaves the region by crossing the other boundary.

By the new control strategy, for the trajectory shown in Figure 3.5, only one switch signal is generated. The trajectory starts in state S_a with switch signal δ_a, and then it crosses the lower bound from below the boundary. By the new control strategy, no new switch signal is generated since the system is still in the state S_a. Similarly, no new switch signal is generated in the subsequent intersections between the trajectory and the lower boundary.

A new switch signal is generated when the trajectory leaves the grey region by crossing the upper shift boundary; after the crossing, the system enters into the new state S_b with switch signal δ_b.

Only one new switch signal is generated in Figure 3.5; thus, the limit cycle behavior is avoided. Now, let's formulate a theorem to prove that this is universally true.

Theorem 3.4:

A shift signal generated based on the shift strategy described as Figure 3.5 with shift boundary as Equation (3.9), is transition stable.

Proof:

This proof is carried out on three regions: the region below and including the lower boundary, the region above and including the upper boundary, and the grey region.

In the region below and including the lower boundary, first let's examine the time invariant property. Suppose a range is defined as follows,

$$\left\{ x(t) : \tau \le f_{sb}\left(x(t)\right) \le 0, \in R^2 \right\}, \tau < 0$$

At the lower boundary, the system is in state S_a with switch signal δ_a. After a time increase $T > 0$, suppose the trajectory moves towards the lower boundary. No matter how small τ is, there exists a corresponding T such that the system crosses the lower boundary, and no new switch signal is generated.

Next, let's examine the state space invariant property.

Suppose a range defined as follows,

$$\left\{ x(t) : \tau \le f_{sbl}\left(x(t)\right) \le 0, \in R^2 \right\}, \tau < 0$$

At the lower bound, the system is in state S_a with switch signal δ_a. After a state change $\left\|\Delta x(t)\right\| > 0$ toward the boundary, no matter how small τ is, there exists a corresponding $\left\|\Delta x(t)\right\|$, such that the system crosses the lower boundary, and no new switch signal is generated.

In the region above and including the upper boundary, an approach similar to that of the lower region can be used to show that the shift strategy is both time invariant and state space invariant in this region.

The difference is that the trajectory enters the grey region from the upper boundary, and the range is defined as:

$$\left\{ x(t) : 0 \le f_{sbu}\left(x(t)\right) \le \tau, \in R^2 \right\}, \tau > 0 \quad (3.10)$$

Inside the grey region, according to the definition, no new shift signal will be generated, so it is transition stable. The conclusion is that the shift strategy is transition stable in all three regions; thus, it is transition stable for the hybrid system. □

A final observation for this shift strategy is that the two boundaries are parallel to each other, and the width of the grey region depends on heuristic results based on the properties of the hybrid system. Even though the new state space based switch signal established a transition stable shift strategy, it has a problem. The width of the grey region must be large enough to accommodate all of the disturbances. In the TBA system, because of the large load torque disturbance in the vicinity of the shift time, the grey region width is too large to be of any practical use. For example, in order to accommodate the disturbance by the above shift strategy, the width of the grey region is about 4 N-m out of a 12 N-m range. This large grey region width causes a long shift delay.

In this research, a hybrid switch strategy is used to design a transition stable strategy with a relatively narrow grey region. Specifically, a time constraint between adjacent shifts is added to the switch signal expression. In other words, no shift is allowed for a specified amount of time after the start of the previous shift. This time constraint is determined by the time required to finish the shift action, which can be obtained by actually running the system. The final choice of this time constraint is 0.5 seconds. There is another benefit to having this time constraint. That is, any shift close to the commanded position can be avoided, since these types of shifts generate unwanted disturbances.

Hybrid system stability - TBA

A hybrid system differs significantly from a continuous dynamic system with respect to stability issues. The commonly used terminologies in continuous

dynamic systems are: Lyapunov stable, asymptotically stable, and exponentially stable. These are still valid in a hybrid system, while the definitions are a little different in the sense that the state switch should be included in the definitions.

Consider a hybrid system in the form of Equation (3.11),

$$\dot{x}(t) = f\left(\delta\left(t, x(t)\right), x(t)\right), \quad \delta\left(t_0, x(t_0)\right) = \delta_0, x(t_0) = x_0 \qquad (3.11)$$

with a solution,

$$x(t) = \psi\left(t, (t_0, \delta_0, x_0)\right) \qquad \textbf{(3.12)}$$

and an equilibrium point,

$$f\left(\delta\left(t_e, x_e\right), x_e\right) = 0. \qquad \textbf{(3.13)}$$

For the switch signal $\delta\left(t, x(t)\right)$ with initial conditon δ_0, the following stability definitions of the system are given.

Definitions 3.4: Stabilities: The system is said to be stable under the switch signal at the equilibrium point if, for each $\varepsilon > 0$***, there is a*** $\sigma(t_0, \varepsilon), \tau \geq 0$***, such that,*** $\|x_0 - x_e\| < \sigma\left(t_0, \varepsilon\right) \Rightarrow \left\|\psi\left(t_0 + \tau, (t_0, \delta_0, x_0)\right) - x_e\right\| < \varepsilon$.

The system is said to be uniformly stable under the switch signal at the equilibrium point if, for each $\varepsilon > 0$***, there is a*** $\sigma(\varepsilon), \tau \geq 0$***, such that,***

$$\|x_0 - x_e\| < \sigma\left(t_0, \varepsilon\right) \Rightarrow \left\|\psi\left(t_0 + \tau, (t_0, \delta_0, x_0)\right) - x_e\right\| < \varepsilon.$$

The system is said to be asymptotically stable under the switch signal at the equilibrium point if it is uniformly stable and, there exists, the following condition holds:

$$\|x_0 - x_e\| < \sigma(t_0, \varepsilon) \Rightarrow \lim_{\tau \to \infty} \|\psi(t_0 + \tau, (t_0, \delta_0, x_0)) - x_e\| = 0.$$

The system is said to be exponentially stable under the switch signal at the equilibrium point if it is uniformly stable and, for each $\varepsilon > 0$***, there exists a*** $\sigma > 0$ ***,such that,***

$$\|x_0 - x_e\| < \sigma \Rightarrow \|\psi(t_0 + \tau, (t_0, \delta_0, x_0)) - x_e\| \le e^{-\varepsilon(t_0+\tau)}, \tau \ge 0$$

The asymptotic and exponential stabilities are the most desired stability types for a hybrid system.

Let's look at the Lyapunov stability theorem for a continuous dynamic system. For a continuous dynamic system given by Equation (3.14),

$$\begin{aligned} &\dot{x} = f(x),\ x(t_0) = x_0 \\ &f : R^n \to R^n \end{aligned} \qquad (3.14)$$

Let $x_e = 0$ be an equilibrium point of Equation (3.14), and $V(x): R^n \to R$ be a continuous differentiable function.

Theorem 3.5: Lyapunov stable [39] - Under the above conditions, if

$$\begin{aligned} &i) \quad V(0) = 0 \\ &ii) \quad V(x) > 0, x \ne 0 \\ &iii) \quad \dot{V}(x) \le 0, x \ne 0 \end{aligned}$$

Then the equilibrium point is Lyapunov stable.

Theorem 3.6: Asymptotically stable [39] - Under the same conditions as Theorem 3.5, if

$$\begin{aligned} &i) \quad V(0) = 0 \\ &ii) \quad V(x) > 0, x \ne 0 \\ &iii) \quad \dot{V}(x) < 0, x \ne 0 \end{aligned}$$

Then the equilibrium point is asymptotically stable.

The above two theorems establish a method to test the stability of a continuous dynamic system as given by Equation (3.14).

A hybrid system involves several such continuous dynamic systems; therefore, new theorems are presented to test the stability [14, 15, 20, 24]. These theorems are included here for completeness.

First, let's check if the TBA system is quadratically stablizable under certain switch signals. Let's look at other supporting theorems[19].

Lemma 5.5 [19]:

System (5.1) is quadratically stablizable if there exist gain matrices F_i such that the matrix pencil $\left\{\sum_{i\in M}\varpi_i\left(A_i+B_iF_i\right):\varpi_i\geq 0,\sum\varpi_i=1\right\}$ ***contains a Hurwitz matrix.***

The following definition is provided for Lemma 5.5.

$$
\begin{aligned}
&System\ (5.1)\ \text{is}\\
&\dot{x}(t)=A_ix(t)+B_iu_i(t)\\
&y(t)=C_ix(t)
\end{aligned}
$$

Theorem 3.7: If hybrid system as Equation (3.15) has controllable continuous subsystems,

$$
\begin{aligned}
&\dot{x}(t)=A_ix(t)+Bu_i(t)\\
&y(t)=C_ix(t)
\end{aligned}
\qquad (3.15)
$$

Furthermore, if there exists a single state feedback which can stabilize all subsystems, then the hybrid system is quadratically stable using the same state feedback.

Proof:

$$\sum_{i\in M}\varpi_i\left(A_i+B_iF_i\right)=\sum_{i\in M}\varpi_i\left(A_i+BF\right)=\sum_{i\in M}\varpi_iA_i+BF$$

By Lemma 5.5 [11], $\sum_{i\in M}\varpi_iA_i+BF$ has to be stable for some i.

In order to show that at least one of the above expressions is stable, we only need to show that there exists a special i, such that $\left(\sum_{i\in M}\varpi_iA_i,B\right)$ is controllable. In fact, such case is easy to find by setting only one ϖ_i to 1, say, ϖ_1=1, and the rest to 0s; then, $\left(\sum_{i\in M}\varpi_iA_i,B\right)$ becomes (A_1,B), which is controllable by the theorem condition. □

The TBA prototype has three continuous dynamic subsystems, and a common feedback controller can be found to stabilize all three subsystems. According to Theorem 3.7, the TBA system is stable under synchoronous switch signal. The above theorem succefully proven that the TBA system is quadratically stablizable. But in practice, the gear shift is determined on the load toruque, which limits the use of this synchronous switch stabliltiy.

Now let's check if the TBA system is stable under asynchoronous switch, the following theorems in literature are provided for proving this stability.

Theorem III.1 [14]: Let D be a compact linear polysystem, the following are equivalent:

1. ***The origin is a uniformly exponentially stable equilibrium,***
2. ***The origin is a uniformly asymptotically stable equilibrium,***
3. ***There exists a C^1 positive definite function $V:R^n\to R$, homogeneous of degree two, such that $x\mapsto\nabla V(x)Ax$ is negative definite for all $A\subset D$,***
4. ***There exists a C^∞ positive definite function $V:R^n\to R$, such that $\nabla V(x)Ax$ is negative definite for all $A\subset D$.***

Notice that condition 3 is equivalent to stating that there exists a common Lyaponov function, because,

$$\dot{V}(x) = \frac{\partial V}{\partial x}\dot{x} = \frac{\partial V}{\partial x}Ax \equiv \nabla VAx$$

$$\text{let V=x}^{\text{T}}Px > 0,\ \text{P>0}$$

$$\dot{V}(x) \equiv \frac{d\left(x^T Px\right)}{dt}$$

$$= \dot{x}^T Px + x^T \dot{P}x + x^T P\dot{x}$$

$$= x^T A^T Px + x^T PAx$$

$$= x^T (A^T P + PA)x$$

$$< 0$$

$$or, A^T P + PA = -Q,\ Q > 0$$

Theorem 3.1 [24]: A necessary and sufficient condition for the dynamic systems $\sum A_1$ and $\sum A_2$ ***to have a CQLF (Common Quadratic Lyapunov Function) is that the pencils*** $\sigma_a\left[A_1, A_2\right],\ \sigma_a\left[A_1, A_2^{-1}\right]$ ***are both Hurwitz.***

The following definitions apply to Theorem 3.1.

$$\sum A_1 : \dot{x}(t) = A_1 x(t)$$

$$\sigma_a\left[A_1, A_2\right] \equiv \alpha A_1 + (1-\alpha)A_2, \alpha \in [0,1]$$

$$CQLF - \text{common quadratic Lyapunov function}$$

Lemma 3.1 [24]: Let $\sum A_1, \sum A_2,$ and $\sum A_3$ ***be stable second order LTI systems that pairwise satisfy the conditions of Theorem 3.1 [24], and, with*** $a_{21i} \neq 0, i \in \{1,2,3\}$ ***, let*** $\varepsilon_{Ai} \cap \varepsilon_{Aj} = \varnothing$ ***for some*** $i, j \in \{1,2,3\}, i \neq j$ ***. Then a symmetric positive definite matrix P exists such that*** $V(x) = x^T Px$ ***is a CQLF for all three systems*** $\sum A_1, \sum A_2,$ and $\sum A_3$ ***.***

The following definitions apply to Lemma 3.1.

a_{21i} is the element of matrix A_i on the 2nd row, and 1st column

$$\varepsilon_{Ai} = \left\{ P_i : \det(A_i^T P_i + P_i A_i) > 0 \right\}$$

$$P_i^T = P_i > 0$$

Theorem III.1 establishes an equivalent relation between exponential and asymptotic stability and the existence of a common Lyapunov type equation. Theorem 3.1[24] and Lemma 3.1[24] together give a sufficient condition for a CQLF for a hybrid system involving three 2nd order LTI systems, which is exactly the configuration of the TBA model.

Equipped with the above three theorems, let's look at the TBA example. The TBA subsystem dynamic equations are given by Equation (3.16).

$$\dot{x}(t) = A_i x(t)$$

when load is zero

$$A_1 = \begin{bmatrix} 0 & 1 \\ -0.4262 & -0.0062 \end{bmatrix}, A_2 = \begin{bmatrix} 0 & 1 \\ -0.3123 & -0.0084 \end{bmatrix}, A_3 = \begin{bmatrix} 0 & 1 \\ -0.1828 & -0.0142 \end{bmatrix} \quad (3.16)$$

when load is maximum

$$A_1 = \begin{bmatrix} 0 & 1 \\ -0.3159 & -0.0013 \end{bmatrix}, A_2 = \begin{bmatrix} 0 & 1 \\ -0.2315 & -0.0018 \end{bmatrix}, A_3 = \begin{bmatrix} 0 & 1 \\ -0.1359 & -0.0031 \end{bmatrix}$$

It is trivial to show that all the above systems are stable and controllable, and they also satisfies the conditions of Theorem 3.1 [24]. Due to the small damping coefficient (0.001), the conditions specified in Lemma 3.1 [24] are not satisfied. Therefore, the open loop system is not guaranteed to be stable under any switching signals. One remedy to the above problem is to design different controller for different subsystem such that the following two conditions hold:

a) The closed loops are stable; b) The closed loop transfer functions share the same expression.

The state feedback control system diagram is given in Figure 3.6.

The new system dynamic equation is given by:

$$\dot{x}(t) = (A - bF)x(t),$$
$$b = B_1 = B_2 = B_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad (3.17)$$
$$F = \begin{bmatrix} f_1 & f_2 \end{bmatrix}$$

If the above two conditions hold, the conditions in Lemma 3.1 [24] are satisfied, thus the hybrid system is stable under asynchorounous switch.

Controller Design and Implementation

The purpose of this section is to find a suitable and practical controller for the TBA and to design it such a controller that satisfies the conditions in Theorem 3.7. In addition, this controller must be able to attenuate the transient response caused by the mismatch between the motor speed and the DVT speed before and after a specific gear shift.

Such transient disturbances could lead to high accelerations of the manipulator that would be detrimental to precise manipulator motion, and they could also cause some difficulties in gear shift control [25, 26].

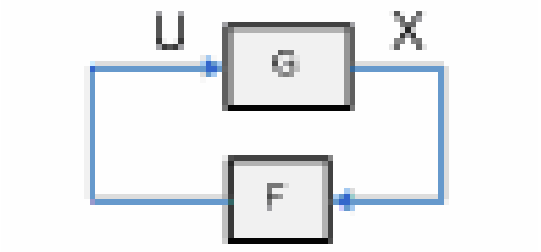


Figure 3.6, State feedback control diagram

Our approach is focused on state feedback control, which can then be transformed into an equivalent proportional-integral-derivative (PID) controller. Other controller design techniques such as optimal control and robust control will be introduced and analyzed. Like most control system applications, the TBA control design starts with design requirements.

TBA control design requirements

Due to the complexity of this system, some of the requirements are quantitative, while others are qualitative, and the main requirements are:

1. The shift strategy must be transition stable.
2. The hybrid system must be stable.
3. The transient response due to the gear shift should be adequately attenuated.
4. Steady state errors should be less than 1 degree (the resolution of the encoder is ~0.36 degree) for compatibility with manipulator control.
5. The servo control loop sample rate should be no less than 1000Hz (the bandwidth of the TBA prototype is about 1500Hz).
6. The gear shift should be finished in no less than 30 ms (the time constant of the brake action is about 30 ms).

State Feedback Control

As discussed in the previous section, a common state feedback controller for all three subsystems of the TBA must be found in order to achieve stability under asynchronous switch signals.

The subsystems are controllable, so the closed loop gain can be arbitrarily set. Suppose a pair of complex numbers $-5 \pm j3$ is arbitrarily selected as the closed loop poles. Then the required closed loop characteristic equation is given by:

$$s^2 + 10s + 34 = 0 \qquad (3.18)$$

The resulting feedback controllers are given by:

when load is zero,

$F_1 = [33.5738 \quad 9.9938], F_2 = [33.6877 \quad 9.9916], F_3 = [33.8172 \quad 9.9858]$

when load is maximum,

$F_1 = [33.6841 \quad 9.9987], F_2 = [33.7685 \quad 9.9912], F_3 = [33.8541 \quad 9.9969]$ (3.19)

where

$F_1, F_2, F_3 -$ feedback controller as F in Figure 3.6

The resulting subsystems share the same characteristic equations. Although this approach is effective in eliminating the differences among the systems, it has an obvious drawback: an infinite number of feedback controllers are required to accommodate various loads.

Further examination of the feedback controllers as shown in Equation (3.19) reveals that that the feedback controllers for all three systems vary little from zero load to maximum load. By using a single feedback controller designed on a single load, i.e. zero load, it is easy to show that all the subsystems are stable under different loads. By Theorem 3.7, using this feedback controller for all three subsystems will guarantee TBA system stability. Under the feedback controller F_1 described in Equation (3.19) for zero load, the systems are given by Equation (3.20).

$$\dot{x}(t) = A_{fi}x(t)$$

when load is zero,

$$A_{f1} = \begin{bmatrix} 0 & 1 \\ -34.0000 & -9.9999 \end{bmatrix}, A_{f2} = \begin{bmatrix} 0 & 1 \\ -33.8861 & -10.0022 \end{bmatrix}, A_{f3} = \begin{bmatrix} 0 & 1 \\ -33.7566 & -10.0080 \end{bmatrix}$$

when load is maximum,

$$A_{f1} = \begin{bmatrix} 0 & 1 \\ -33.8897 & -9.9951 \end{bmatrix}, A_{f2} = \begin{bmatrix} 0 & 1 \\ -33.8053 & -9.9956 \end{bmatrix}, A_{f3} = \begin{bmatrix} 0 & 1 \\ -33.7097 & -9.9969 \end{bmatrix} \quad (3.20)$$

where,

$$A_{fi} = A_i + BF_1$$

By using the state feedback, a common Lyapunov function does exist by using the method in [24]. The Lyapunov function sets are given in Figure 3.7 and Figure 3.8. A quick look at the A matrices with the state feedback reveals that all the subsystems have similar A matrices.

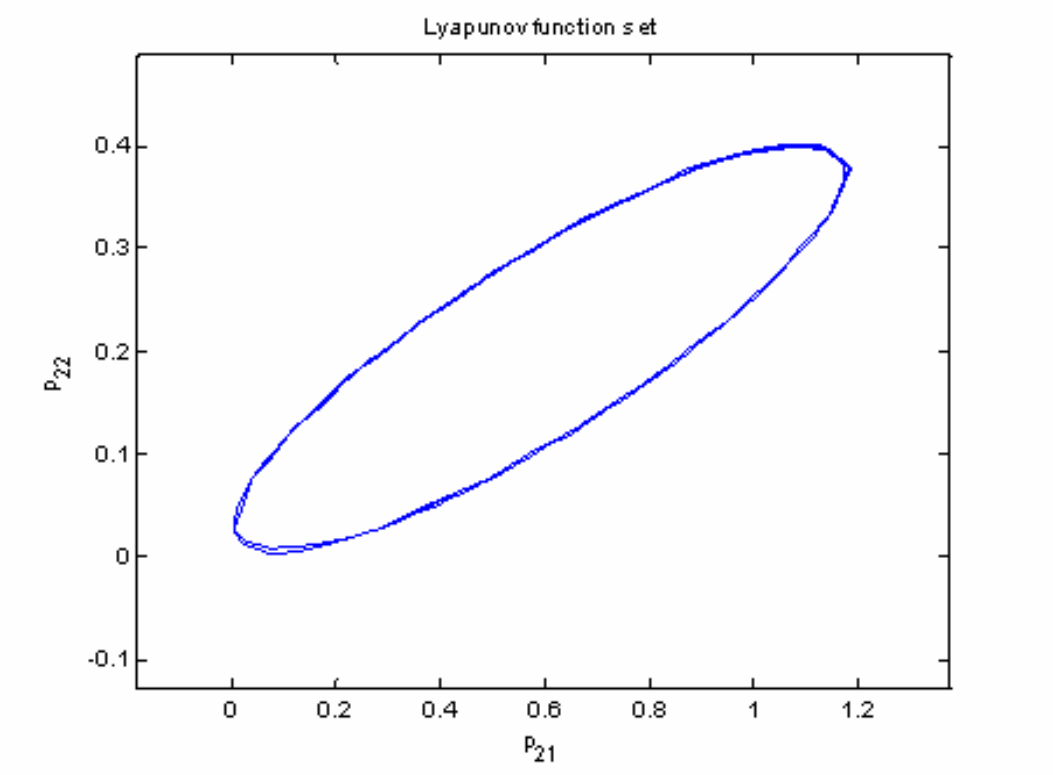


Figure 3.7, Lyapunov function sets for zero load

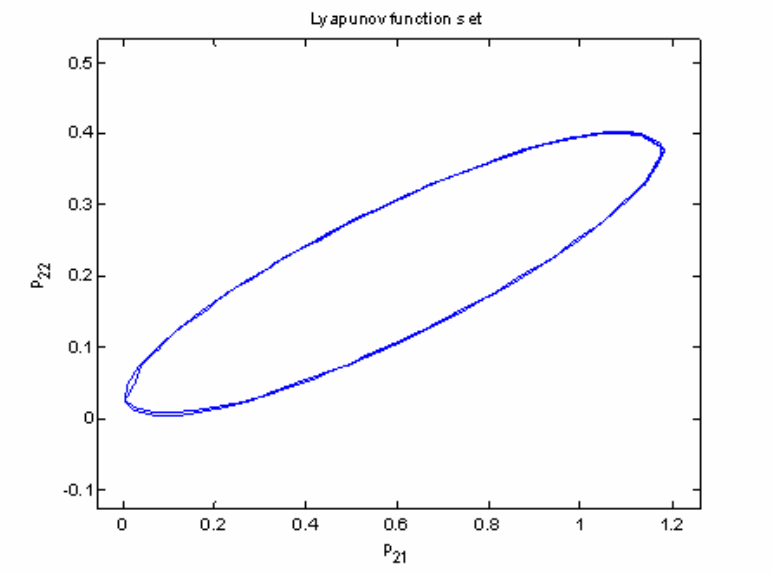


Figure 3.8, Lyapunov function sets for maximum load

Based on Figure 3.7 and Figure 3.8, it is clear that all three Lyapunov sets share a common area (inside the ellipsis) for both zero and maximum load.

The system shown in Figure 3.9 is equivalent to Figure 3.6, because the TBA system involves three 2nd order continuous subsystems.

For the TBA prototype used in this research, only position feedback is available and thus F is equivalent to a PD controller. The PID controller has several design conveniences based on the following considerations:

1. A majority of servo motor control applications use PID control [40].
2. Small calculation overhead makes it suitable for real time control.
3. Practice and theory show that PID servo motor control can achieve position and velocity tracking simultaneously [2].

The third result is very important since it provides a solution to the transient response issues that result from the speed mismatch before and after the shift.

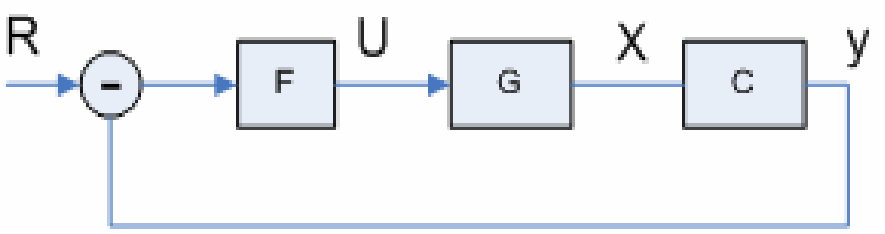


Figure 3.9, Equivalent control subsystem diagram

If the output speed can be tracked well enough, the transient response can be suppressed by designing a smooth speed trajectory.

The detailed implementation of the PID control top level diagram is given in Figure 3.10.

A continuous time PID controller algorithm is shown in Equation (3.21).

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{de(t)}{dt} \qquad (3.21)$$

where

K_P, K_I, K_D gains

$e(t) = r(t) - y(t)$ error

u(t), r(t), y(t) control, reference, and output respectively

There are two digital implementation forms for the continuous PID controller [41]: a position form and a velocity form. A position form and a velocity form are given in Equation (3.22) and (3.23) respectively.

$$u(k) = u(0) + K_p e(k) + K_i T_s \sum_{i=1}^{k} e(i) + K_d \frac{e(k) - e(k-1)}{T_s}$$

K_P, K_I, K_D -gains

e(k) = r(k)-y(k) -error (3.22)

u(k), r(k), y(k) -control, reference, and output respectively

T_s -sample period

$$u(k+1) = u(k) + K_p(e(k) - e(k-1)) + K_i T_s e(k) + K_d(e(k) - 2e(k-1) + e(k-2))/T_s \qquad (3.23)$$

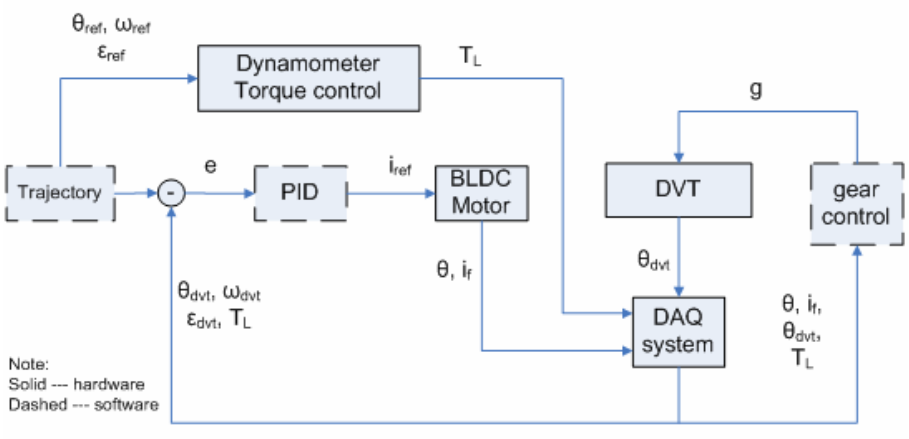


Figure 3.10, Top level control system diagram

It is easy to see that the number of additions at a single step is linearly increased with time for the position form, while for the velocity form, it is constant. The velocity PID form is used in the research.

Due to the existence of the integral term and the servo motor speed/current limit, windup situations need to be avoided. In this research, conditional integration is used. The analysis of this saturation effect is not the interest of the research; interested readers are encouraged to find reference in [42]. Other anti-windup methods can be found in [40]. As mentioned previously, it is proven that a servo motor with a PID controller can achieve position tracking and velocity tracking simultaneously [2, 40]. This research extends the theory in [2] by closing the loop at the DVT output instead of the servo motor output. The idea is that if the DVT position and speed can be tracked well enough, the transient disturbance due to the state change can be attenuated as well. A TBA prototype servo and DVT control system diagram is shown in Figure 3.11. In this research, the dynamic payload experienced by the robot arm is emulated by a dynamometer.

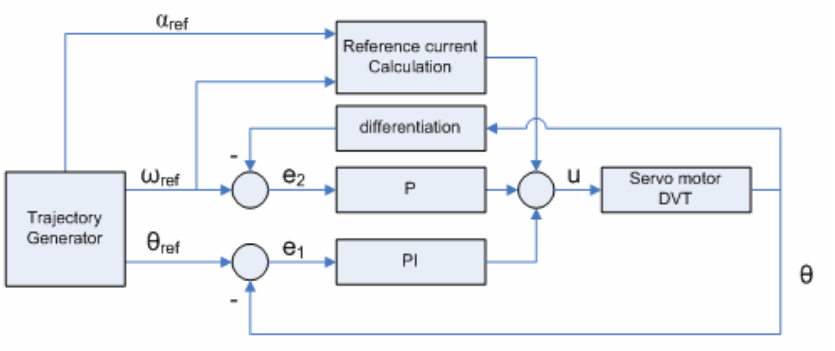


Figure 3.11, Servo motor control diagram

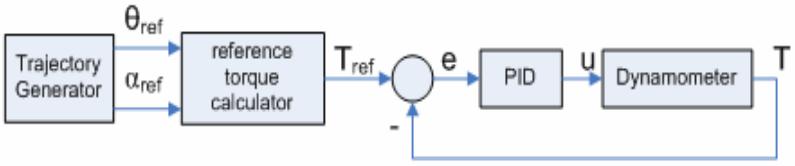


Figure 3.12, Dynamometer torque control diagram

However, the static gravity of the robot arm load cannot be emulated because the particular dynamometer used cannot generate torque output when the speed is zero. The dynamometer output torque is a nonlinear function of speed and reference voltage. Open loop control of the dynamometer would require a complete calibration of the dynamometer.

In this research, a closed loop design is adopted to track the reference torque, which is generated based on the trajectory and load model.

Thus, a full calibration of the load cell attached to the dynamometer is adequate. The control system diagram is shown in Figure 3.12.

In summary, the proposed PID control system for the TBA is proven to be stable (including transition stable).

It is also proven that this proposed controller can suppress the transient disturbances expected from the gear shifting.

The final TBA control system diagram based on PID control is given in Figure 3.13.

As in Figure 3.13, the proposed TBA control system includes a trajectory generator, a supervisory controller, and two closed loop control subsystems: servo motor control and dynamometer control.

The trajectory generator takes a position (set point) as input, and generates a trapezoidal type trajectory, the output of the trajectory generator is the position command sent to the servo control (r(t)), and position, acceleration commands sent to dynamometer control (r'(t)). The servo motor closed loop control takes the generated trajectory as input, and tracks the commanded trajectory. The dynamometer closed loop control generates the load torque profile based on the generated trajectory, and the load profile is calculated based on Equation (2.39). The actual torque is measured by a load cell and sent to the supervisory controller to determine the appropriate gear shift.

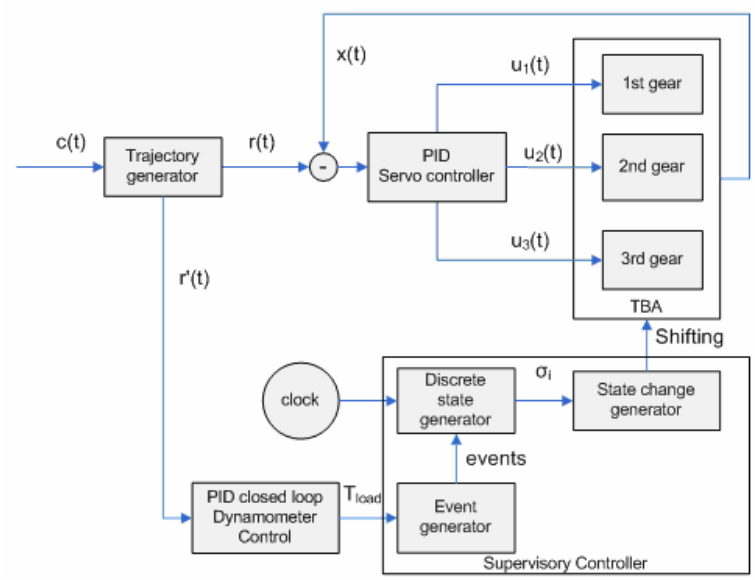


Figure 3.13, TBA Control System Diagram

The TBA supervisory controller has three blocks: an event generator, a discrete state generator and a state change generator. The supervisory controller takes a clock signal and the torque generated by the dynamometer as input, and sends out explicit shifting commands to engage/disengage brakes based on the proposed control strategy as shown in the next paragraph. Before introducing the individual blocks, the TBA control strategy is first presented:

1. The TBA shift strategy has two shift interfaces:

   Interface 1: gear shift between 3rd gear and 2nd gear.

   Interface 2: gear shift between 2nd gear and 1st gear.

2. Each shift interface has two shift boundaries:

   Interface 1: $T_{load} \geq 5$, and $T_{load} \leq 4$ Nm

   Interface 2: $T_{load} \geq 8$, and $T_{load} \leq 6$ Nm

3. Time between adjacent gear shifts must be greater than 0.5 seconds.

The shift strategy is shown in Figure 3.14.

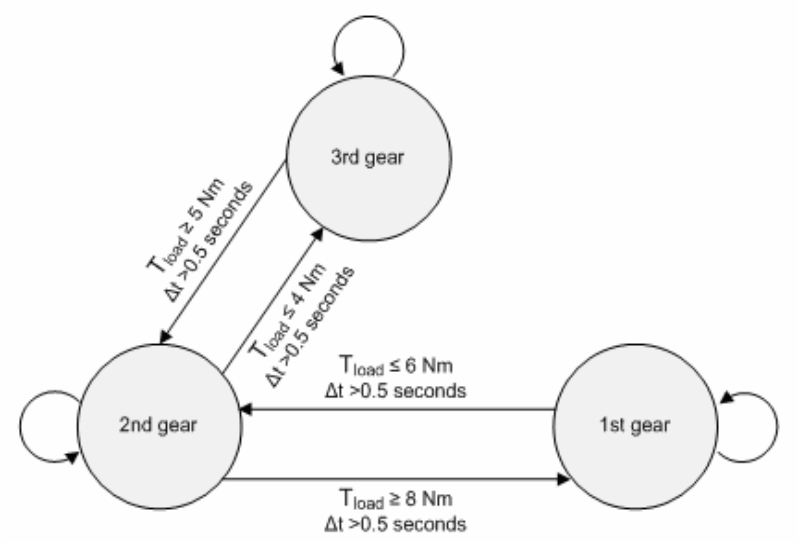


Figure 3.14, TBA gear shift strategy (prototype)

There are a couple of clarifications need to be made here: first, the shift boundaries and Δt are chosen by experiments. There may be other different set of values under which the TBA system functions normally. Second, the TBA prototype could potentially have six different gear shift actions: 1->2,2->1,2->3,3->2, 3->1,and1->3.

In this research, direct shifts between 1st gear and 3rd gear are not considered since there will be no such gear shifts in the prototype. In real TBA application, such shifts are possible due to the inertia load induced by large load accelerations. The shift strategy is shown in Figure 3.15.

By using the shift strategy shown in Figure 3.14, and a single PID controller for the TBA control, by Theorem 3.4: and Theorem 3.7:, the TBA control strategy as shown in Figure 3.14 is transition stable, and the TBA system is guaranteed to be stable under asynchronous shifts.

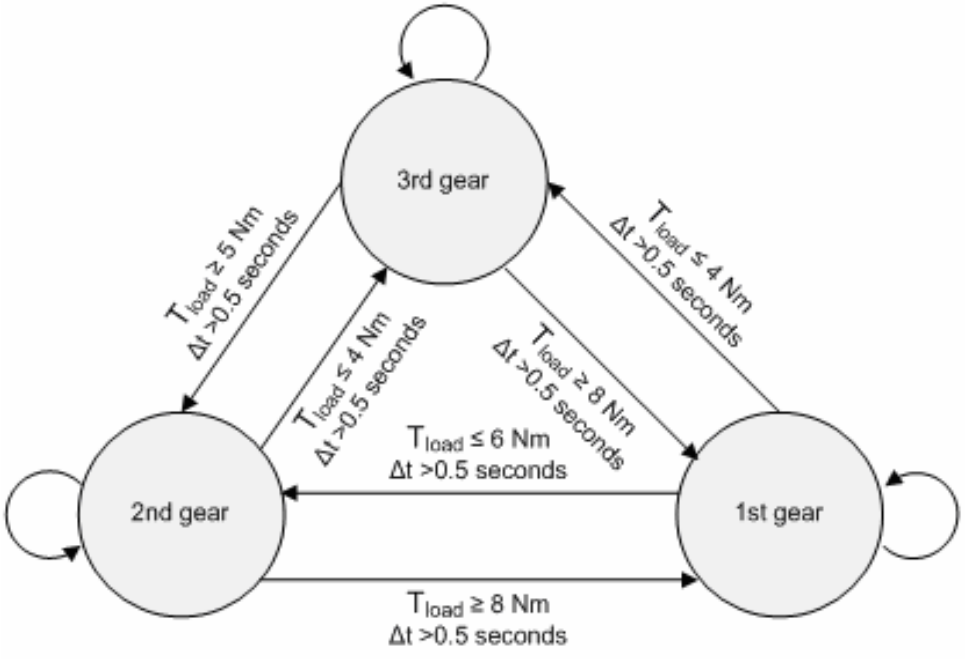


Figure 3.15, TBA gear shift strategy (real system)

Now, let's have a close look at the individual blocks in the supervisory controller. The event generator takes the actual load torque generated by the dynamometer and generates shift boundary crossing events based on the gear shift strategy shown in Figure 3.14.

The crossing events are only generated when the load torque crosses the boundary in certain direction. For example, when the load torque crosses the 4N-m boundary from below, no crossing event will be generated; while it crosses the same boundary from above, a crossing event will be generated and sent to the discrete state generator.

The discrete state generator takes two inputs: the crossing events, and the time, and determines the next TBA gear state. The process to determine the next TBA gear can be described as: If either no crossing event is received or the time from the previous shift is less than 0.5 seconds, the output of the discrete state is the current state; if both a crossing event is received and the time condition are satisfied, it generates a new state signal, which is then sent to the state change generator.

The state change generator receives the state signal from the discrete state generator, and generates all the control commands for the three brake motors.

The process can be described as: Once it receives a state signal, it compares the state signal with the current state of the TBA. If the two are same, no control command will be sent; if they are different, new shift commands will be generated and sent to the brake motors.

The control of the engage/disengage action is achieved by sending digital signal to the amplifiers of the motors. As presented in Chapter 1, only one brake can be engaged at a time, so every gear shifting action involves three different brake actions: disengaging the current brake, engaging the objective brake, and keeping the other brake disengaged.

The above control strategy can be explained more clearly by using a design example as follows:

1. The robot arm is in a vertical down position initially.
2. The robot arm moves up from the initial vertical down position to a final vertical up position following a trapezoidal shaped trajectory.

Under the above two conditions, the rotational load torque is zero initially, as the robot arm moves up to the horizontal position, the load torque increases from zero to the maximum; then as the arm continues to move up to the vertical up position, the load torque decreases from the maximum to zero. The functions of the proposed control strategy can be described as follows:

1. Initially, the load torque is less than 5 N-m, and TBA is in the 3rd gear with the lowest ratio.
2. When load torque is greater than or equals to 5 N-m, and the time since the previous shift is greater than 0.5 seconds, TBA shift to the 2nd gear.
3. When load torque is greater or equals to 8 N-m, and the time since the previous shift is greater than 0.5 seconds, TBA shift to the 1st gear.
4. When load torque is less than or equals to 6 N-m, and the time since the previous shift is greater than 0.5 seconds, TBA shift to the 2nd gear.

5. When load torque is less than or equals to 4 N-m, and the time since the previous shift is greater than 0.5 seconds, TBA shift to the 3rd gear.
6. The TBA will stays in the 3rd gear until the robot arm is vertical up.

The above gear shift strategy is shown in Figure 3.14. The experimental results of the above design example are given in Chapter 5.

Optimal Control and Robust Control Discussion

At the outset of this research, a key objective was to investigate various control approaches and to investgate their applicability to TBA control. The following discussion presents the results of looking at optimal and robust control theories.

Optimal Control

The analysis in this section serves as an initial investigation into the application of optimal control theory to the TBA control problem. Due to the fact that a hybrid system includes both continuous dynamics and discrete dynamics, the optimal control problem has much richer contents compared with a continuous dynamic system. Two types of optimal problems have drawn special interests: optimal performance by optimal switching signals and optimal performance in terms of disturbance rejection and tracking error. The first problem has been successfully applied in some applications. For example, in a launch vehicle control system, the optimal fuel consumption is achieved by turning on and off the rocket engine at the right time. In the automotive engine, the optimal controller changes the engine working regions to achieve minimum

fuel consumption [43]. In a helicopter control example [37, 38], a linear quadratic regulator is used to achieve “bumpless transfer” while the helicopter undergoes a controller switch. In the TBA, two types of optimal control are valid and have practical uses. These two optimization problems are:

1. Maximize the servo motor power to achieve the fastest possible operation.
2. Minimize the transient response during the gear shift.

This research only focused on the design of an optimal controller. For stability issues of a general hybrid system optimal control problem, see [19, 22]. In this research, the second optimization problem is formulated and a solution is given. First, let’s have a look at the general optimal control TBA system diagram shown in Figure 3.16.

The control objective is to find an optimal controller F_o to obtain the minimal control index. Since every gear shift includes two TBA gear states, it is valid to study a system with any two of the three states of the TBA first, and then use a similar approach to design optimal controllers for all other state changes.

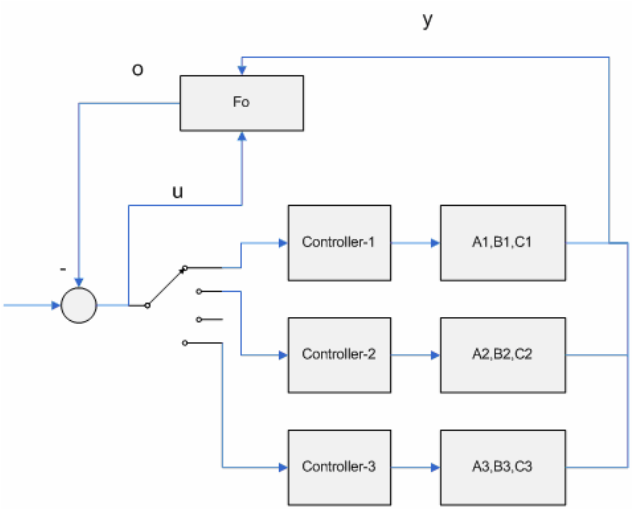


Figure 3.16, General TBA optimal control diagram

The control diagram with two states is shown in Figure 3.17. An optimal control problem is,

$$\begin{aligned}\dot{x}(t) &= A_1 x(t) + B u_1(t)\\ y_1 &= C_1 x\\ J &= \frac{1}{2}\int_0^{t_f}\left(w_1\left(y_1(t)-y_2(t)\right)^2 + w_2\left(u_1(t)-u_2(t)\right)^2\right)dt + \frac{1}{2}w_3\left(y_1(t_f)-y_2(t_f)\right)^2\end{aligned} \quad (3.24)$$

The index objective is,

$$\begin{aligned}\tilde{J} &= J + \int_0^{t_f}\lambda(t)^T\left(\left(A_1 x(t) + B u_1(t)\right) - \dot{x}(t)\right)dt\\ &= \int_0^{t_f}\left(\frac{1}{2}\left(w_1\left(C_1 x(t) - C_2 x_2(t)\right)^2 + w_2\left(u_1(t)-u_2(t)\right)^2\right) + \lambda(t)\left(\left(A_1 x(t) + B u_1(t)\right) - \dot{x}(t)\right)\right)dt\\ &\quad + \frac{1}{2}w_3\left(C_1 x(t_f) - C_2 x_2(t_f)\right)^2\end{aligned} \quad (3.25)$$

$\lambda(t) - Lagrange\ multiplier\ or\ co-state$

The optimal solution conditions are given in Equation (3.26).

$$\begin{aligned}&\frac{\partial\tilde{J}(t)}{\partial u_1(t)} = 0, \quad \frac{\partial\tilde{J}(t)}{\partial x(t)} = 0,\\ &\frac{\partial\phi(t)}{\partial x(t_f)} = 0, \quad \frac{\partial\tilde{J}(t)}{\partial\lambda(t)} = 0, where\\ &\phi(t) \equiv \frac{1}{2}w_3\left(C_1 x(t_f) - C_2 x_2(t_f)\right)^2 + \int_0^{t_f}\lambda(t)\dot{x}(t)\,dt\end{aligned} \quad (3.26)$$

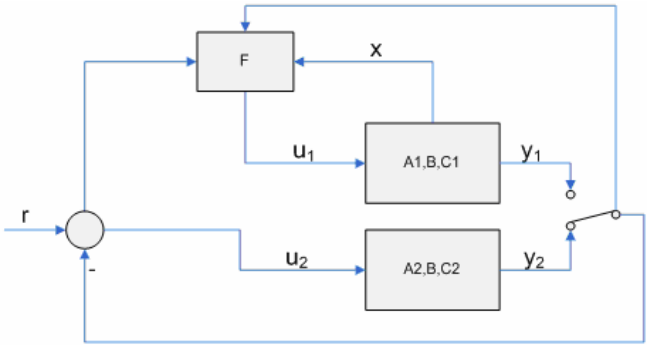


Figure 3.17, Second optimal control problem system diagram

From $\frac{\partial \tilde{J}(t)}{\partial u_1(t)} = 0$, the function of $u_1(t)$ can be expressed as in Equation (3.27).

$$u_1(t) = \begin{bmatrix} 1 & -\frac{B^T}{w_2} \end{bmatrix} \begin{bmatrix} u_2(t) \\ \lambda(t) \end{bmatrix} \quad (3.27)$$

The co-state $\lambda(t)$ can be solved by a state space formulation given in Equation (3.25).

$$\begin{bmatrix} \dot{x}(t) \\ \dot{\lambda}(t) \end{bmatrix} = \begin{bmatrix} A_1 & \frac{B^T B}{w_2} \\ -w_1 C_1^T C_1 & -A_1^T \end{bmatrix} \begin{bmatrix} x(t) \\ \lambda(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_2(t) + \begin{bmatrix} 0 \\ w_1 C_1^T C_2 \end{bmatrix} x_2(t)$$
$$x(t_o) = x_0 \quad (3.28)$$
$$\lambda(t_f) = w_3 C_1^T \left(C_1 x(t_f) - C_2 x_2(t_f) \right)$$

This is the so-called two point boundary problem; the solution of this problem is related to a Riccati differential equation and another differential function after using a technique called a “sweep” [44]. These equations are :

$$-\dot{M}(t) = M(t) A_1 + A_1^T M(t) - M(t) \frac{B^T B}{w_2} M(t) + w_1 C_1^T C_1$$
$$-\dot{g}(t) = w_1 C_1^T C_2 x_2(t) + M(t) B u_2(t)$$
$$with \quad (3.29)$$
$$M(t_f) = w_3 C_1^T C_1 x(t_f)$$
$$g(t_f) = w_3 C_1^T C_2 x_2(t_f)$$
$$\lambda(t) = M(t) x(t) - g(t)$$

In Equation (3.29), $\lambda(t)$ can be solved in a backward iteration, although the control signal $u_2(t)$ and $x_2(t)$ must be known, which limits the use of this method.

In summary, a global optimization problem results in a two point boundary problem, which can not be solved in this application. However, it is possible to find a class of suboptimal solutions by using some assumptions. For example, if assuming $t_f \to \infty$, the optimization problem is referred to as an infinite time horizon problem, and the solution can be solved without using a co-state variable [37]. This infinite time horizon assumption can be treated as a relaxation of the global optimization conditions. The solution of suboptimal problems presents a set of completely different problems for the TBA system, which is not the thrust of this reseach due to the hardware limitation in the TBA prototype system and will be left for future research. Even though the suboptimal solution could produce a solvable optimization problem, some practical problems need further investigation:

1. The resulting control is generally high order, and digital implementation of the high order controller generates big error while using the approximation method to change transfer function from s domain to z domain.
2. Large calculation overhead needs special attention in applications requiring real time performance.

Robust Control

Robust control method offers a viable approach to DVT controller design. In this research, the robust control serves as an initial investigation of the robust control problems in the TBA prototype.

Specifically, robust controller design to achieve input disturbance rejection is presented and analyzed, even though in general, robust control method may

used to solve two types of problems: disturbance rejection and reduced sensitivity to parameter uncertainty. The system diagram of input disturbance rejection problem using robust control method is shown in Figure 3.18.

As shown in Chapter 2, TBA system has three 2nd order subsystems. The plant (P) in Figure 3.18 is one of the three TBA subsystem models. In the following design example, the controller design is accomplished by using Matlab μ-synthesis toolbox. In this specific case, the plant is the TBA in its 2nd gear with the dynamic equation shown in Equation (2.35), K is the controller to be designed. Other signal names and weighting matrices in Figure 3.18 are:

1. nt - sensor noise
2. ut - weighted control signal
3. dt - disturbance
4. Wu - input weighting function, a high pass filter,
5. Wd, Wn – disturbance, noise weighting function, high pass filters
6. We – output weighting function, a low pass filter

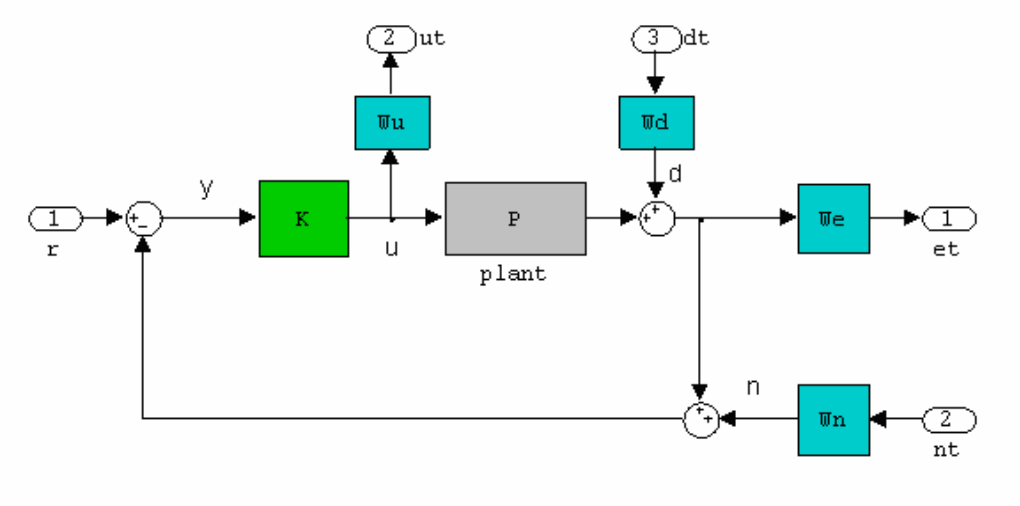


Figure 3.18, TBA robust for disturbance attenuation

The output weighting function (We) specifies the system performance, that is, at low frequency, the open loop system have much larger gains than at the high frequency, such that the Matlab H_∞ algorithm only needs to take care of the low frequency response. Other weighting functions (Wu, Wd, and Wn) specify the interested bands for different inputs and output. A general explanation on how to select the weighting functions can be found in [45]. The sigma values of the We and Wu (Wd, Wn) are shown in Figure 3.19. Notice that the sigma value is the same as Bode magnitude for the above matrices, the Matlab μ-synthesis toolbox uses sigma value instead of Bode plots in order to deal with the multi Inputs multi outputs system. The open loop system is formed when the controller (K) is removed from the closed loop system shown in Figure 3.18. The open loop system has four inputs and three outputs. The inputs are u, r, dt, nt and the output are y, et, and ut. The open loop system has six simple poles and is not stable:

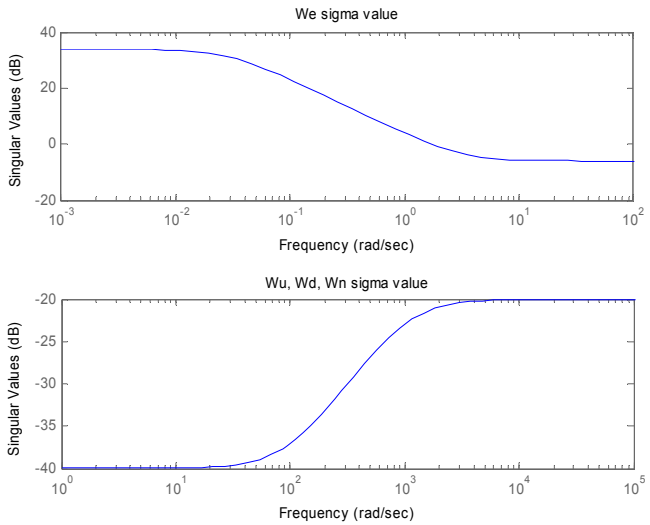


Figure 3.19, Open and closed loop magnitudes

p_{o1}=-39.7116, p_{o2}=-7786.731,

p_{o3}=-39.738, p_{o4}~p_{o6}=-1000.000

where, p_{oi} – open loop poles (i=1,2,...,6)

By using the Matlab μ-synthesis toolbox, a sixth-order controller was designed, and has the following transfer function:

$$\frac{(s+0.012)(s+39.738)(s+1000.000)^2(s+1000.189)}{(s+0.0001)(s-591.540)(s+991.924)^2(s+1002.994)(s+1.249*10^6)}$$

The resulting closed loop system has the following twelve simple poles:

p_{c1}=-1240862.662, p_{c2}=-7786.731, p_{c3}=-39.738,

p_{c4}=-3.000, p_{c5}=-0.030, p_{c6}=-0.0001,

p_{c7}=-993.216, p_{c8}=-1006.337, p_{c9}=-1003.955,

p_{c10}~p_{c12}=-1000.000

where, p_{ci} – closed loop poles (i=1,2,...,12)

Figure 3.20 shows the open and closed loop magnitude respectively.

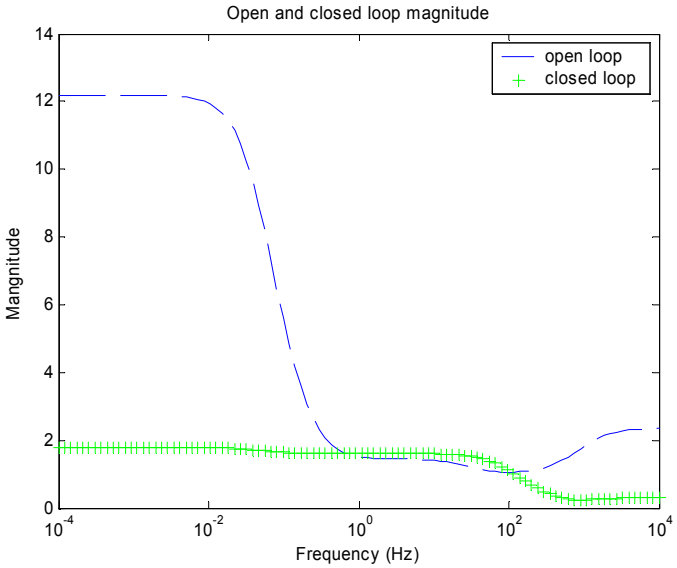


Figure 3.20, Open and closed loop magnitudes

Based on the above results and Figure 3.20, the following can be concluded:

1. Open loop system is unstable, its bandwidth is small, open loop gain is larger than 1 at high frequency.
2. Closed loop system is stable, its bandwidth is wider, and system output rolls off at high frequency.

Even though the robust control method successfully produced a stable closed loop system that satisfied the performance and disturbance rejection requirements, some practical problems need further investigation as in the optimal control problem:

1. Digital implementation of the high order controller generates big error while using the approximation method to change transfer function from s domain to z domain.
2. Large calculation overhead needs special attention in applications requiring real time performance.

Summary

This chapter studied the control problems associated with the TBA prototype and its experimental system using hybrid supervisory control theory.

The proposed PID based controller is proven to be stable under asynchronous gear shift. The proposed control strategy is transition stable under load torque disturbances. The proposed control system is implemented in a PC-based real time control framework, which will be discussed in detail in Chapter 4.

The experimental results in Chapter 5 will verify the correctness of the TBA control system framework and the controller design methodology presented here above.

Initial studies of optimal control and robust control for the TBA system are also presented in this chapter, these studies and results provide useful reference for future work about control of the TBA.

CHAPTER 4

Control System Software

In the software development, the TBA prototype and its experimental system is divided into four subsystems. These subsystems are the servo motor control (SMC), DVT, dynamometer control (DMC), and PC controller/data acquisition system (PCC). The SMC, the DVT, and the DMC subsystems are functionally independent of each other in the sense that each of them can be individually controlled with a PC.

On the top level, these subsystems need communication and coordination in order to fulfill the functionality requirements of the TBA system. Figure 4.1 shows the TBA subsystems and their interconnection relations.

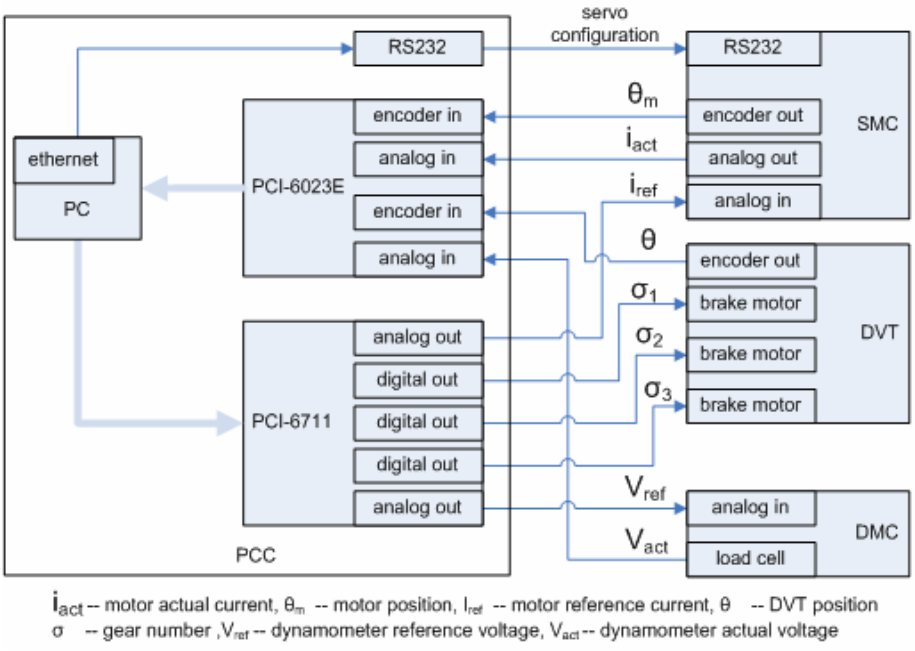


Figure 4.1, TBA subsystems interconnection relation

There are two closed loop control subsystems in the system: the SMC and the DMC. The PCC subsystem, equipped with a serial port, Ethernet, A/D, D/A, a timer/counter, and computation power, plays a central role in the TBA control system design, which can be shown in the following TBA system function flow during a single control step.

In each control step, the PCC first reads the real time clock ticks to determine the current loop start time. If the timer satisfies the commanded system loop period condition, the PCC generates a desired trajectory to send to the SMC to control the BLDC servo motor.

The PCC reads the position of the DVT output shaft and uses it as the feedback signal to generate the BLDC and dynamometer control signal in order to track the trajectory and the desired load profile.

The PCC also reads the actual torque output from the dynamometer in order to generate the next gear shift signal. At the same time, the PCC reads other relevant system information such as the BLDC torque and position, and it also sends the experimental data to a high level control PC for data display over a dedicated local area network (LAN).

The high level control computer is a Linux PC and is not shown in the system diagram.

TBA Design Requirement

Like most practical control system software implementation, the TBA control software design starts with a requirement analysis. Due to the complexity

of this system, some of the requirements are quantitative, while others can only be given as qualitative constraints. The main requirements and constraints are:

1. The servo control loop must be able to run at a rate no less than 1000Hz, and the deterministic loop period should be guaranteed with admissible delay.
2. Multithread architecture should be adopted for parallel execution.
3. The software should be capable of setting different threads to different priority levels as required by the task.
4. The system should have adequate flexibility, be easy to maintain, and provide friendly documentation measures.
5. The system should provide most commonly used PC hardware support.
6. The system should have C/C++ programming support.
7. The system should have Ethernet communications.
8. The system overall price should be within reasonable limits.

The control sample rate is determined based on the time constant of the TBA. The calculations show that the TBA time constant is in the range of 0.0020~0.0025 seconds, which correlated with the dynamic bandwidth of the system. Generally, high sample rates have positive effects like reduced quantization error, but noise captured by high sample rates must be attenuated, usually with a low pass filter.

Due to requirement 1, a Real Time System (RTS) is required for TBA control. The core of a RTS is a real time operating system (RTOS) environment where real time constraints can be met.

Operating System - Introduction

In a real world control system with a PC, the operating system plays a very important role. It not only provides the computation power for any algorithm, but it also manages all the hardware drivers and user applications to make them work together seamlessly, sharing computer resources like memory, I/O, interrupts, etc.

A PC operating system can be divided into two major categories: a real time operating system (RTOS) and a general purpose operating system (GPOS). They can be differentiated by the following features: preemptive kernel, priority scheduling, interrupt handling, etc.

GPOS and RTOS

A generally accepted definition of a GPOS states that the correct result depends only on the logical correctness of the computation.

For example, for a printer connected to a PC, it generally does not matter whether it takes 50 or 100 milliseconds to start the printing job, as long as all the text and graphics are printed correctly.

In this type of situation, a general purpose operating system (GPOS) is adequate.

On the top level, the different types of GPOS are similar, although they can be vastly different in the implementation.

Figure 4.2 shows the three layer architecture model for a standard Linux operating system.

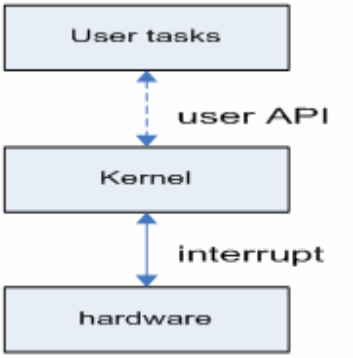


Figure 4.2, Standard Linux three layer model

One common feature of GPOS is that it is interrupt driven [46], or event driven. Specifically, when an interrupt line is raised, the operating system sees it immediately and responds to it in a timely manner, which generally involves stopping the current process or waiting until the current interrupt is handled.

PC GPOS can be further divided into three main categories: Microsoft Windows family operating systems, various Unix flavored operating systems, and different Linux implementations, which are all delivered as GPOS. As will be discussed later, a Linux operating system can be adapted into an RTOS, thus allowing a fair competition for system time between interrupt handling and user applications.

Compared with the GPOS, a definition of an RTOS can be conceivably expressed as the following.

Definition 4.1: RTOS - the correct result not only depends on the logical correctness of the computation, but also on the time required for the result to be generated.

The following two examples are typical applications where an RTOS is required. The first example is a computer controlled radar tracking system. In this

system, the position of a target must be calculated accurately and the result must be obtained in a timely manner, otherwise, the tracking information is not very useful.

Another example is a computer controlled robot arm emergency stop function. It monitors the status of prescribed safety rules, and if any of the rules are violated, the robot arm is shut down.

Suppose that during a robot arm operation, someone accidentally enters the restricted area, thus triggering the safety rule violation condition. Then the control software sends out a command to shut down the arm actuator, and probably applies the brakes, also.

If this command is not sent out and executed quickly enough, the person may be struck and injured by the robot arm. Under GPOS, there is one solution to this example, which is to assign an interrupt handler to any safety violation.

But this approach is generally not recommended since there is a limited number of interrupt lines, most of which have already been occupied by commonly used hardware.

Generally, in these situations, control software developed in a GPOS cannot perform the tasks well simply because the time of execution of any instruction is unpredictable or non-deterministic.

Let’s take the standard Linux kernel as an example to see why a GPOS is not suitable for this type of application.

In a GPOS, a task can only have two states: stopped or running. One clarification must be made here. In a multitask GPOS, all of the running tasks

could have a third state while waiting for their time slice to come; this state is generally not treated as a new state because the user program does not have control over it. Figure 4.3 shows the task status in Linux.

As mentioned earlier in this section, interrupts are handled first, before any user programs, because the GPOS generally gives higher priority to hardware interrupts than to the user program. Therefore, the user program can be executed only when all interrupt lines are inactive.

Furthermore, there is no way to predict when an interrupt line will be raised, which makes any user software with high time constraints subject to non-deterministic time delays in applications with a large number of hardware interrupt routines. Another limitation is that a user program does not have direct control of the CPU clock or the system timer under a GPOS. It is the kernel that has exclusive control over these two high accuracy clocks, even in a multitask system. For example, Linux has a multitask kernel which is attained by splitting the system time into time slices by using a clock interrupt, and the kernel assigns the highest priority to the system clock interrupt. One dedicated time slice is assigned to one task only. A task can only be executed in its assigned time slice, during which other tasks are either suspended or waiting to start execution.

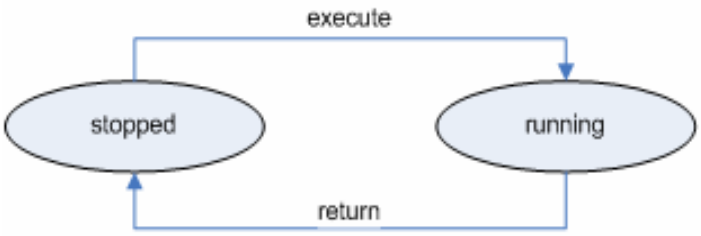


Figure 4.3, GPOS task status

In Linux, the time slice is generally on the order of tens of milliseconds [47], which means that any task with a deadline that occurs before its predefined time slice will be executed too late or not at all.

A GPOS uses the time sharing policy to guarantee that all tasks have some system time to be executed. This feature of a GPOS is referred as a fair scheduling policy. In a GPOS, features like the fair scheduling policy and the non-preemptive kernel can lead to non-deterministic time delays [48]. On the other hand, an RTOS is capable of meeting the deadline requirements, as in the radar tracking and robot emergency stop examples shown earlier in the chapter.

On the top level, an RTOS provides a priority based scheduling policy and a preemptive kernel, and these two features together ensure that the higher priority tasks are executed promptly, with some jitter delays resulting from the context switch. The low priority tasks are put in the suspended state (or ready state), and their execution is resumed after all higher priority tasks are finished. So, a task in an RTOS has one more state compared with that in a GPOS, and this additional state is called the block state. Figure 4.4 shows the task states in an RTOS.

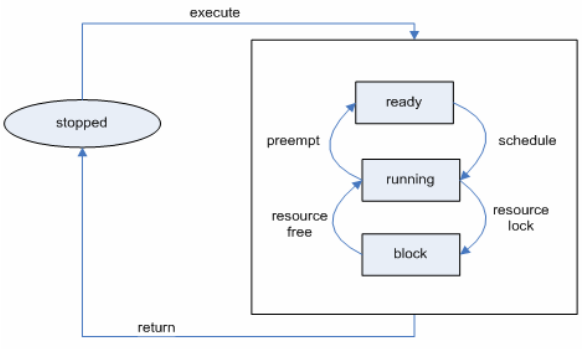


Figure 4.4, Process state diagram in an RTOS

In an RTOS, when a process is activated, it is first put in a ready state in a waiting queue. If there is no other equal or higher priority process in the queue, it is put in the first place. If there is no other equal or higher priority process running in the processor, it is in the running state immediately. All lower priority processes currently running will be preempted and put back in the ready state while waiting for the higher priority process to release the processor.

There is one problem, however. If a higher priority process happens to share a resource which is locked by a lower priority process, the higher priority process can not be executed right away, and it must wait until the lower priority process frees the resource, thus causing the higher priority process to suffer from a time delay.

This situation can be worse when the lower priority process is preempted by other higher priority processes, no matter whether these processes have higher or lower priority than the previous higher priority process. The higher priority process, which shares the resource locked by a lower priority process, suffers from a non-deterministic time delay. The reason for this delay is that the resource could never be released since the lower priority process is preempted by other higher priority processes, which then cause the higher priority process to wait on the resource and miss its deadline. This situation is known as priority inversion.

Generally, two POSIX protocols [49] can be adopted to deal with the priority inversion. One is known as priority inheritance, a protocol that allows two processes which share the same resource to be treated with same priority during

the resource synchronization period; thus, the only time missed by the higher priority process is a single resource access time from the lower priority thread. This time is generally sufficiently small in an RTOS.

The other protocol is known as priority protection, which changes the processes priority to the highest priority of all the resource locks it has, such that multiple processes sharing a common resource have the same priority. However, the small block of time still exists in this protocol as in the first protocol.By having a preemptive kernel, priority based scheduling and some priority inversion avoidance protocol, the RTOS can meet most stringent real time system constraints. There are cases where a PC-based RTOS is not sufficient.

The reason is that all preemptive and rescheduling actions need some CPU time, which is known as context switch time, or jitter. During the context switch time, the processor will store the memory to be preempted and used for later process reentry. Typical jitter time for a PC based RTOS is in the magnitude of several microseconds [50]. In the TBA projects, since the control loop time period is in the milliseconds level, a PC based RTOS is adequate. VxWorks, QNX, RTAI, RtLinux and INtime are some examples of commonly used RTOS for PC based control system.

The first two are true RTOS in the sense that they are delivered as RTOS; the last three, however, are patched RTOS based on a GPOS. RtLinux and RTAI are based on Linux, and INtime is based on Microsoft Windows. Even though different types of RTOS are different in implementation, they all provide the same satisfactory real time performance. In this research, RTAI/Linux is used.

RTAI/Linux

The Real Time Application Interface is a hard real time extension to the Linux kernel, contributed in accordance with the Free Software guidelines. It provides the features of an industrial grade RTOS and is seamlessly accessible from the powerful and sophisticated GNU/Linux environment.

This project has been founded by the Department of Aerospace Engineering of Politecnico di Milano (DIAPM). Over the years, it has become a community effort involving international developers, coordinated by DIAPM's Prof. Paolo Mantegazza [51].

RTAI uses a Linux patch which enables the GPOS Linux to fulfill tasks with real time constraints when the RTAI is loaded.

There are two patches that have been used. The first one is called the Real Time Hardware Abstraction Layer (RTHAL), which is developed for RTLinux. The second one is called the Adaptive Domain Environment for Operating Systems (ADEOS), which is developed under the GNU General Public License (GPL) to provide a flexible environment for sharing hardware resources among multiple operating systems, or among multiple instances of a single OS. New versions of RTAI have been ported from RTHAL to ADEOS.

RTAI/Linux Architecture

RTAI can be treated as a GPOS Linux kernel module. When it is loaded, the system is an RTOS; otherwise, the system is just a Linux GPOS. The Linux/RTAI system architecture can be shown in Figure 4.5.

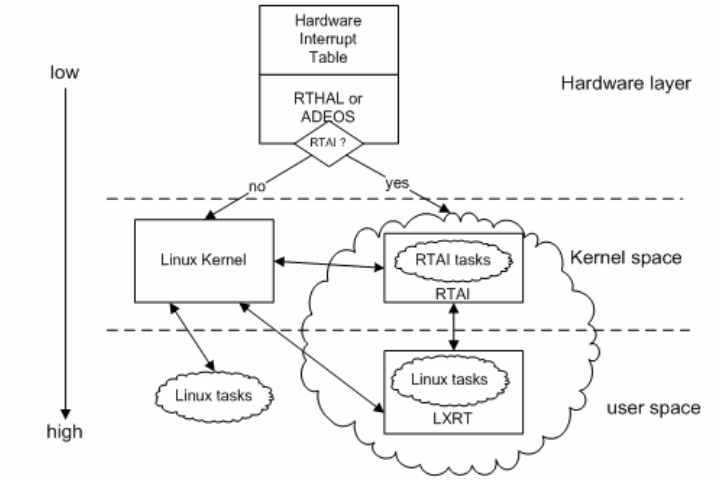


Figure 4.5, RTAI/Linux system architecture

On the lowest level are the hardware interrupt and the RTHAL or ADEOS patch. The patch collects all the pointers to the internal data structures and functions that affect the real time operation into a single structure (rt_hal), traps all these function calls to the member of rt_hal data structure, and makes the pointers in rt_hal point to the redefined RTAI functions.

A definition of rt_hal data structure is as shown as follows [52]:

```
struct rt_hal {
struct desc_struct *idt_table;
void (*disint)(void);
void (*enint)(void);
unsigned int (*getflags)(void);
void (*setflags)(unsigned int flags);
void (*mask_and_ack_8259A)(unsigned int irq);
void (*unmask_8259A_irq)(unsigned int irq);
void (*ack_APIC_irq)(void);
void (*mask_IO_APIC_irq)(unsigned int irq);
void (*unmask_IO_APIC_irq)(unsigned int irq);
unsigned long *io_apic_irqs;
```

```
void * irq_controller_lock;
void *irq_desc;
int *irq_vector;
void *irq_2_pin;
void *ret_from_irq;
};
```

The middle level is a coexisting Linux kernel and RTAI environment. The user program can be developed in the kernel space and loaded into the kernel as a module, like a hardware driver, such that hard real time (HRT) can be achieved with optimal performance.The highest level is the user space, where all user programs can use the standard library functions.

The RTAI provides a symmetric real time programming environment in kernel space and user space. The user space real time program environment is called LXRT, an extension on RTAI, which provides good average real time performance with some unbearable spikes [53].

In this project, RTAI 3.0 and Linux 2.4.25 kernel with RTHAL patch are used.

LXRT

LXRT enables a symmetric use of RTAI functions in the user space and kernel space, both HRT and soft real time (SRT). With LXRT, it is very convenient for any C programmer to implement and test a real time control system in the user space without dealing with the kernel space problems. It is convenient also because most of the GNU standard library, for example, math library functions as defined in math.h, can be called in the user application.

For HRT in user space, there is one constraint: any call to a Linux kernel service should be avoided. For example, the POSIX file operation functions as defined in stdio.h should not be called. These Linux services will cause a context switch between the RTAI and the Linux kernel. In a control system with a high control loop rate, for example ~20 KHz, the context switch time might violate the real time constraints. The worst case occurs when the Linux kernel service is blocked; then the HRT task will certainly miss the deadline.

So, as a rule of thumb, Linux kernel service calls should be avoided inside a HRT task. One way to use service call is to have a server – client configuration by using the IPC calls provided by RTAI/LXRT, for example, RTAIFIFO, RTAI shared memory, etc. Another advantage of LXRT is that the user space real time program can be ported into kernel space real time fairly easy because all the RTAI/LXRT function calls have the same names and definitions as the ones in the kernel space.

Programming languages and tools

The programming languages utilized in this research are C and C++. The control system software is strictly C style programming simply because the RTAI and LXRT are both written in C, and our DAQ drivers are also written in C. The high level GUI is based on QT and QWT, which are both written in C++.

The drivers package for the A/D, D/A boards' used in this research is called Linux control and measurement device interface (Comedi) [54], which is developed and maintained by a group of free software enthusiasts including David Schleef, and Frank Mori Hess.

Comedi is a collection of drivers for a variety of common data acquisition plug-in boards. The drivers are implemented as a core Linux kernel module providing common functionality and individual low-level driver modules. The Comedi project develops open-source drivers, tools, and libraries for data acquisition. The drivers provide features like integrated real time support for most hardware, a high-level library (comedilib), application-level device independence, and compatibility with Linux 2.0, 2.2, 2.4, 2.6 kernels, and support RTAI.

The source code is managed by Concurrent Version Systems (CVS), which provides remote check in, check out source codes and user privilege management. CVS is an open source software package and can be downloaded free of charge from http://ftp.gnu.org/non-gnu/cvs/. Doxygen, which is used as the software documentation tool, is an open source software package developed under a GPL license. Doxygen can be downloaded at www.doxygen.org.

TBA Software Design

In this section, the detailed design of the TBA control software is introduced and analyzed. The proposed control software has a multithread framework under a real time operating system. As discussed earlier, RTAI/Linux is used as the real time operating system. Even though RTAI/Linux is claimed to be hard real time operating system, it still has an overhead for context switches in a PC environment. For example, a sampling rate of 10 kHz can be subject to as much as a 30% time delay [55]. This also justifies the chosen sample rate of 1000 Hz, which has a 3% time delay at most.

A Windows/LabVIEW Implementation

A LabVIEW implementation under Windows XP was used earlier in the TBA project to establish the technical feasibility of the TBA [1, 2], and the functional diagram of this implementation is shown in Figure 4.6.

Although this implementation was sufficient to prove the functional feasibility of the TBA, it has several drawbacks:

1. LabVIEW control software is developed under Windows XP, which is a GPOS.
2. The BLDC servo motor control is restricted with serial port only.
3. The control program is a single thread design.

A non-deterministic and long control loop sample time (~100ms), as well as the non-deterministic control loop time, violated the constant control loop time assumption in the controller digital implementation. For example, a digital form PID is shown in Equation (3.23). Experimental results showed that, with the LabVIEW implementation, a maximum servo control loop rate of 10 Hz can be used. Considering that the servo motor has a rated speed of 5000 rpm, the servo control loop frequency is far too low. The long sample rate will introduce large quantization error and performance degradation [56].

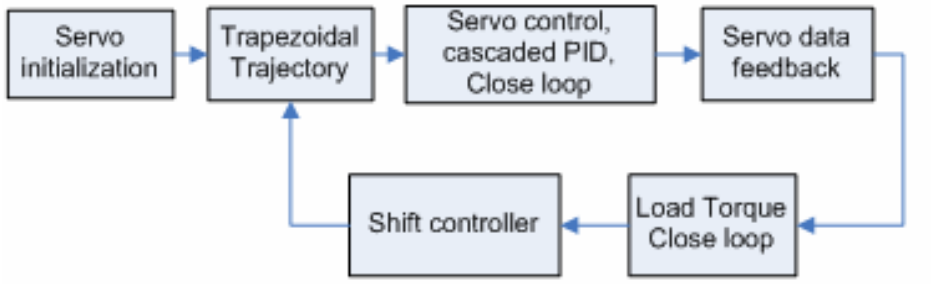


Figure 4.6, Windows/LabVIEW diagram

As a result of the above drawbacks, the performance of the TBA system could not satisfy the design requirements. For example, the steady state position error is about 4 degrees, and the transient response from the gear shift is too large.

The DVT output velocity is subject to up to 50% change in about 30~50 milliseconds in all the gear shift actions [1]. In order to achieve better performance and shorter control loop time, a real time multithread design is adopted in this research.

Multithread Design with RTAI

A multithread design is used since there are several subsystems and each has a different real time requirement. The TBA real time control software structure is shown in Figure 4.7.

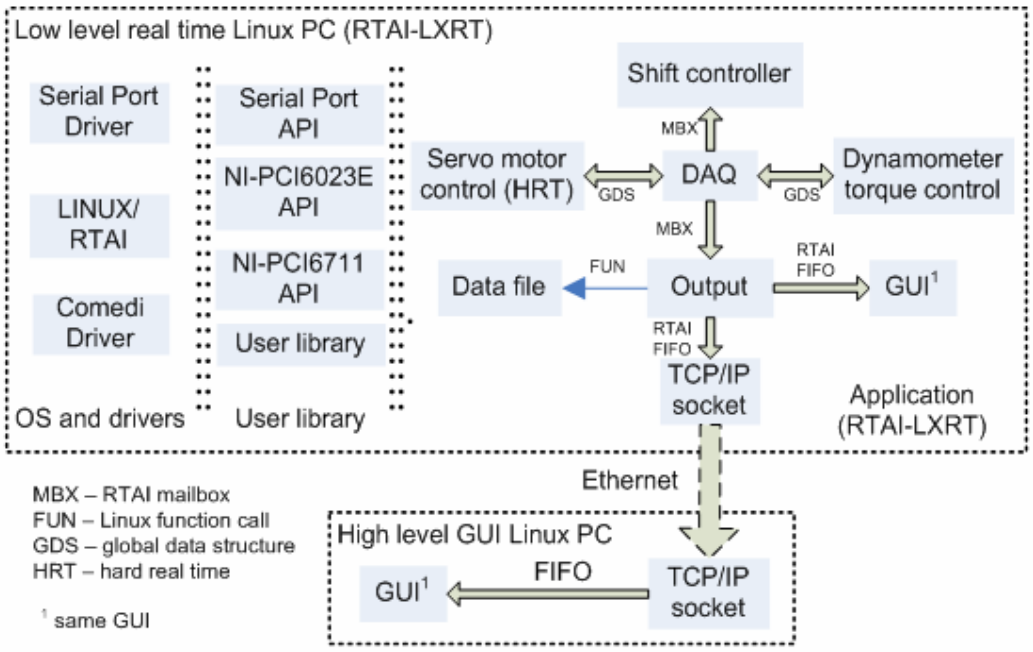


Figure 4.7, TBA real time control software implementation diagram

Two Linux PCs are used. One is a low level control PC (LLPC) with the RTAI, Comedi and all control algorithms. The other is a high level GUI PC (HLPC), which does not have the RTAI, and is used for data display and some system initialization.

There are three layers in LLPC software. The lowest level consists of the Operating System (OS), RTAI, and Comedi drivers; the serial port driver is essentially a part of the Comedi driver. The middle level consists of user libraries and hardware driver APIs developed on top of Comedi and the RTAI; new control algorithms can be added in the user library.

The highest level in the LLPC is the user applications. There are four threads that run simultaneously in the LLPC: servo motor control (SMC), shift control (SHC), dynamometer torque control (DMC), and output (OPT) thread. The DAQ block is not a thread, but in fact is embedded in a corresponding thread. Preemptive priority scheduling is used to guarantee the real time constraint of the SMC. The SMC runs at priority 99 with a hard real time constraint; the other three threads run at priority 98 with a soft real time constraint. Other scheduler schemes are also possible [15].

The SMC is a closed loop thread which controls the servo motor; it also receives load torque data from the DMC and, combined with its own feedback data, determines the gear number and load torque profile and sends these commands to the SHC and SMC.

The SHC receives commands from the SMC and takes the action to engage or disengage gears.

The DMC receives torque commands from the SMC and closed loop control dynamometer torque, and it also sends the data to the SMC.Due to the sharing of data structures among the threads, the RTAI semaphore is used for collision avoidance.

Another functionality of the software is to save and display experimental data. The data are sent to three different blocks: data file, LLPC GUI and Ethernet for HLPC GUI.

The saving of data to files is achieved by calling the function write(), which is inside the SMC block; this violates the HRT constraint [15], but experiments show satisfactory results. An alternative solution could be to send data using a RTAI FIFO, then write to a file in another thread, which requires synchronization between fast and slow threads.

In the LLPC, there is a GUI block which is essentially same as that in the HLPC. The reason to use a GUI in the LLPC is for data display convenience without considering the various issues related to Ethernet.

A Graphical User Interface

A graphical user interface is implemented for displaying the experimental data. It has been decided that the GUI does not need to be real time, and all the real time data must be retrievable from the data stored in the LLPC.

The open source software, Qt Widgets for Technical Applications (QWT), which is developed on top of QT/X11, is used in this research.

QWT is maintained by Josef Wilgen et al., and the latest version is 4.2.0. A screenshot of the GUI is shown in Figure 4.8.

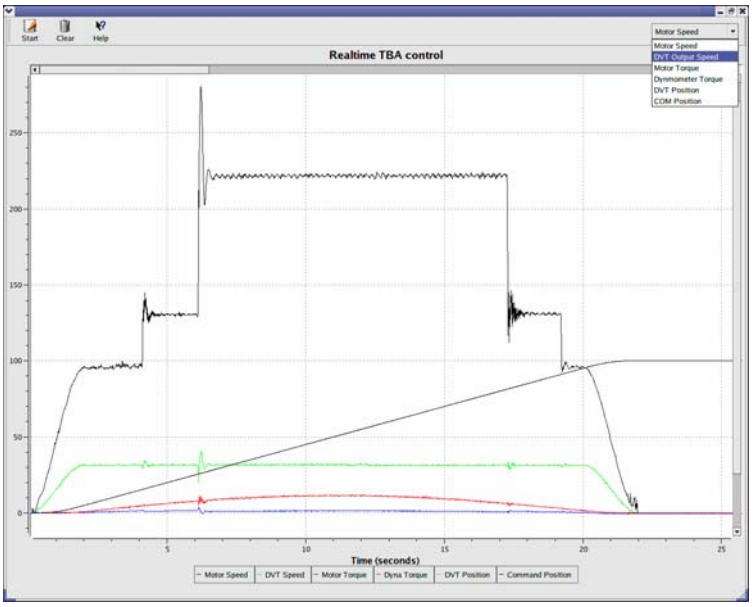


Figure 4.8, Screen shot of GUI

The GUI provides several features:

1. Real time display of six experimental data.
2. An interface to a RTAI FIFO.
3. The ability to dynamically zoom in and out a curve.
4. Arbitrary selection of a curve to be displayed.
5. Fast data display by updating only the new data in the picture.

In summary, the software design meets all of the stated requirements and provides a flexible structure for new controller implementation. It is designed as a HLPC and LLPC configuration, which makes it possible to separate the development system and low level control PC.

Furthermore, compared with the LabVIEW/Windows implementation, the new RTAI real time multithread implementation improves system performance significantly as will be discussed in Chapter 5.

CHAPTER 5

SIMULATIONS AND EXPERIMENTS

In this chapter, the simulation and experimental results using the proposed TBA control and real time implementation will be presented and analyzed. The Matlab/Simulink model simulation serves to evaluate TBA performance under the proposed control method. The experiments serve to evaluate the performance of the TBA prototype under the control method and software implementation. At the end of the chapter, a conclusion of the proposed control for the TBA is given.

Matlab/Simulation

The Simulink models were built on a modular basis as shown in Figure 5.1. There are six blocks in the model: a trajectory block, a controller block, a motor block, a DVT block, an arm block, and a data output block.

BLDC Motor

The parameters of the BLDC motor used in the simulation are listed in Table 5.1.

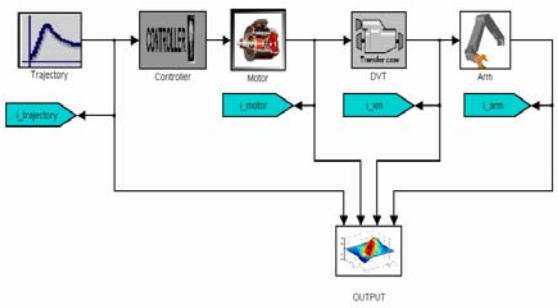


Figure 5.1, Top level Simulink model (Linear Model)

Table 5.1, Motor parameters

Parameters		BLDC motor
Rated Speed		5000 rpm
Rated Torque		4.36 N-m
Rated Power		2829
Torque Constant		0.274 N-m/amp
Resistance		4.511Ω
Inductance		12mH
DC-Link Voltage		250Vdc
d-q Model	Rs	0.242Ω
	Ld,Lq	5.5 mH
	Kt	0.11 N-m/amp
No. of Pole pairs		4
Rotor Moment of Inertia		172.9×10^{-6} kgm^2
Viscous Damping		8.5944×10^{-5} kgm^2/s
Static Friction		0.1 N-m

DVT

In the TBA prototype, a three speed DVT is used. A final gear reduction is also used in the simulation to accommodate the real application of the TBA.

The final gear reduction has a ratio of 200, and is installed between the DVT output shaft and the baseline load. The parameters of the DVT are shown in Table 5.2.

Simulation Results

The simulation results presented in this research can be differentiated by trajectory and load. Two trajectories and three loads were simulated in the research, and the values of these two design parameters are given as follows:

1. Trajectory:
    a. Robot arm rotates 150° from horizontal position (0°).
    b. Robot arm move 90° from horizontal position (0°), then pauses for 2 seconds, then continues to rotate for 60° and stops at 150°.
2. Load: full load, 50% load ,and 10% load

Based on the different DVT types, trajectory types, and load types, there are eighteen different simulation parameter combinations. The test combinations are shown in Table 5.3.

Table 5.2, DVT parameters and motor-DVT combinations

DVT Type	Final Gearbox		DVT	
	Ratio	Efficiency	Ratio	Efficiency
three speed	200	0.90	7/3.8/3.16	0.80

Table 5.3, Test combinations

No.	Motor	DVT	Trajectory (deg)	Load
1	KM B204C	three speed	0~150	Full
2	KM B204C	three speed	0~90~150	Full
3	KM B204C	three speed	0~150	Half
4	KM B204C	three speed	0~90~150	Half
5	KM B204C	three speed	0~150	10%
6	KM B204C	three speed	0~90~150	10%

Table 5.4, Simulation results (Full load)

DVT Type	Settling Time (seconds)	Steady State Error (deg)	Tracking Error (deg)
three speed	4.21	0.38	~65

The evaluations of the TBA performance are carried out in terms of settling time, tracking error, and steady state error. The settling time is defined as the time required for the arm to move into a region within ± 2% of the set point. Figure 5.2 shows TBA performance with a full load and a 0°~ 150 ° trajectory. The simulation results of TBA performance with full load are summarized in Table 5.4, where large tracking error is observed.

A closer look at the speed curves in Figure 5.2 reveals that the tracking error is mainly a result of the inappropriate command speed, or an inappropriate trajectory design. The purpose is to evaluate the maximum speed capability of the TBA system. At full load, the steady state error is 0.38 degree, and the speed of the TBA is about 35.6 degree/second, which over match the baseline actuator specification shown in Table 1.1.

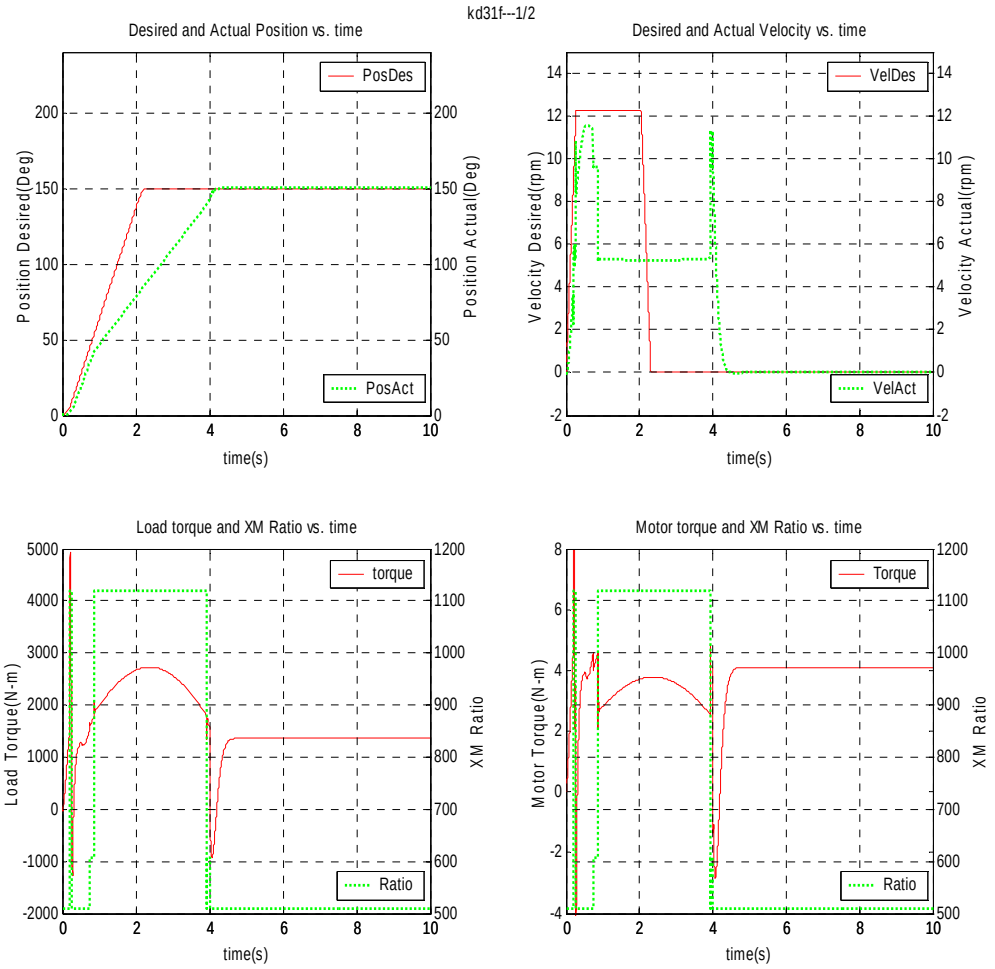


Figure 5.2, Three speed DVT with full load

An immediate solution to the large tracking error is to design a new trajectory with a lower command speed and an appropriate position controller. The result of the new trajectory is shown in Figure 5.3.

By designing a new trajectory, the system has a good tracking ability with a maximum tracking error of ~5 degrees.

The settling time and steady state error are similar as those in the previous trajectory. The new trajectory is used for both half load and 10% load in the rest of the simulations.

The purpose is to examine the control strategy and consistency of the performance with same controller parameters and same trajectory.

The performances for the 50% and 10% load are summarized in Table 5.5.

In summary, the proposed control method meets the TBA control requirement in terms of settling time, steady state error, and tracking error.

The proposed TBA design can match the torque speed requirement of the basleine actuator.

The lacks of the ability to evaluate the shift strategy transition stability under load torque disturbance in this simulation model will be verified by the experiments.

Table 5.5, simulation result (half load)

DVT Type	Settling Time (seconds)	Steady State Error (deg)	Tracking Error (deg)
Half load	2.45	0.38	~5
10% load	2.44	0.36	~5

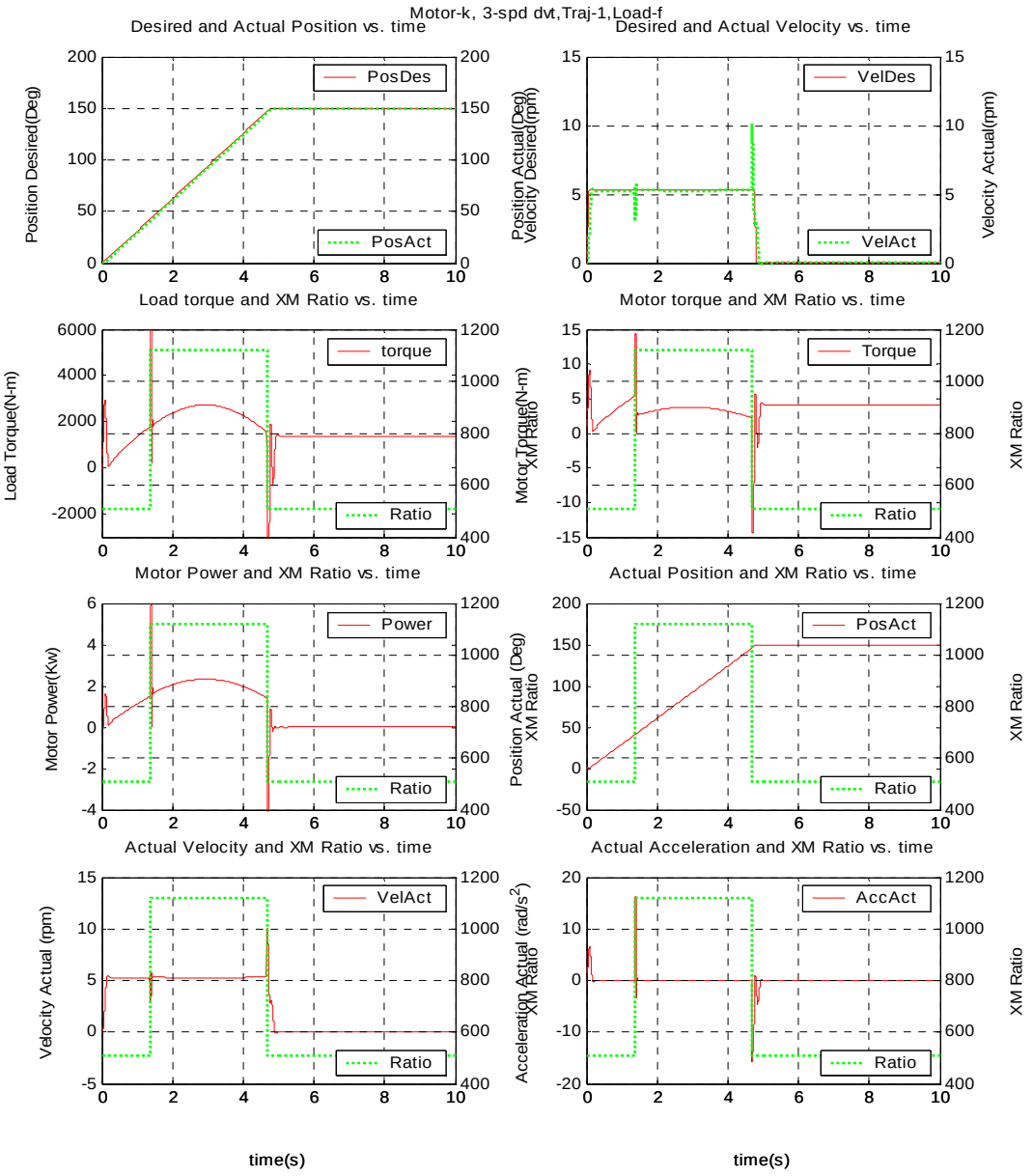


Figure 5.3, Simulation results with new trajectory design

Real Time Control Experimental Results

The experimental results for the TBA prototype presented in this section are for verification of the correctness of the control method presented in Chapter 3 and the effectiveness of the real time software design presented in Chapter 4, thus they are only parts of the complete experimental results, full experimental results are listed in the Appendix. The figures of the experimental results are produced by using Matlab on the actual experimental data generated by the real time control software implementation, and several things need to be clarified in order to fully understand the results:

1. The trajectory is selected such that the robot arm will move from a vertical down to a vertical up position.
2. Position error is evaluated at the DVT output shaft and then converted to the arm joint by dividing the position error by the final gear with a ratio of 200.
3. The three states of the TBA are separated by 5.0 N-m and 8.0 N-m dynamometer torques, which are equivalent to a state based switch signal with a two dimensional shift boundary because the dynamometer torque depends on angular position and acceleration.
4. There are two curves in each figure: the x axis is time in seconds, the left y axis is the TBA gears (1^{st}, 2^{nd}, or 3^{rd}), and the right y axis is the data.
5. The final gear reduction is 200, and errors associated with the final gear box are neglected.

6. The curves in the text are only shown for a full load, for curves of all other loads, please see the appendix.

The results are presented in the following categories:

1. Transition unstable vs. stable switch signals
2. Position errors
3. DVT speed (disturbance suppression)
4. Real time performance measured by difference between required sampling period and the actual period.
5. Load torques

Transition Unstable vs. Stable Switch Signals

The experiment is first carried out to study the transition stability of the switch strategy. As proven in Chapter 3, a shift strategy based on a single shift boundary is not transition stable. The results of transition unstable switch signal based on a single shift boundary are shown in Figure 5.4 and Figure 5.5.

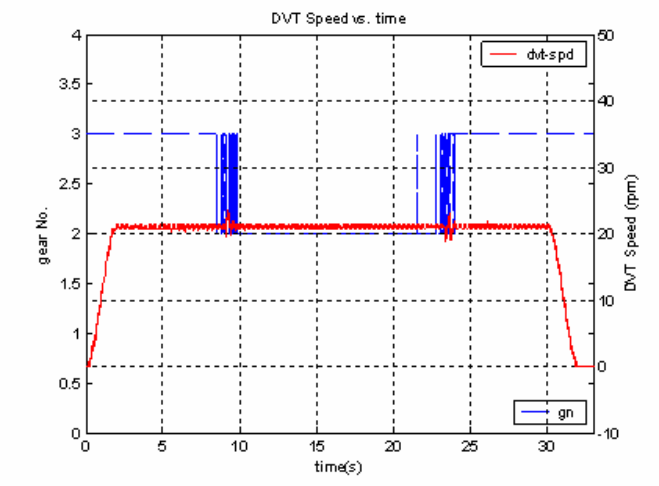


Figure 5.4, Single boundary, state based shift strategy (half load)

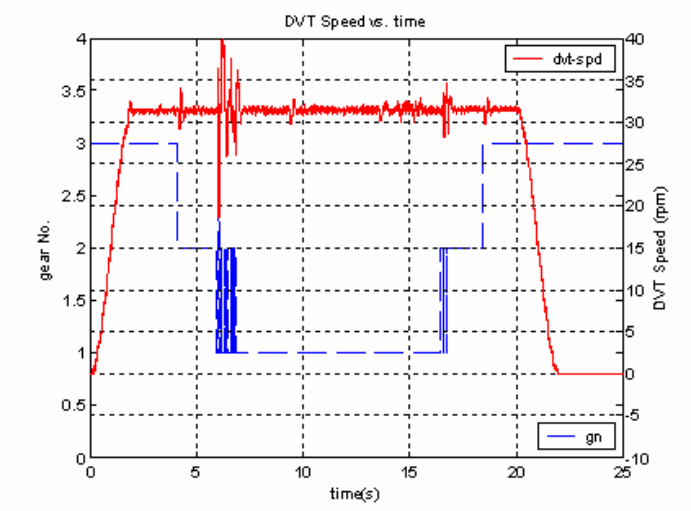


Figure 5.5, Single boundary, state based shift strategy (full load)

From these two figures, we can see that unwanted shifting commands have been generated during the gear shift period for both half load and full load. We can also see that torque disturbances are generated during the shift. As a result, many unwanted shift commands have been generated, which makes the switch signal unstable.

As discussed in the Chapter 3, two measures are adopted in this research to prevent these unwanted shift signals. The first one is to set the grey region width to 1 N-m equivalent of dynamometer torque. The second one is to set the minimal time between adjacent shifts to 0.5 seconds.

After implementing the above measures into the software, new results show that the switch signals are transition stable, which is consistent with the analysis in the Chapter 3. Figure 5.6 and Figure 5.7 show the transition stable results corresponding to the unstable results in Figure 5.4 and Figure 5.5 respectively. Remember that the choice of the width of the grey region and the time constraint in the two measures are not unique.

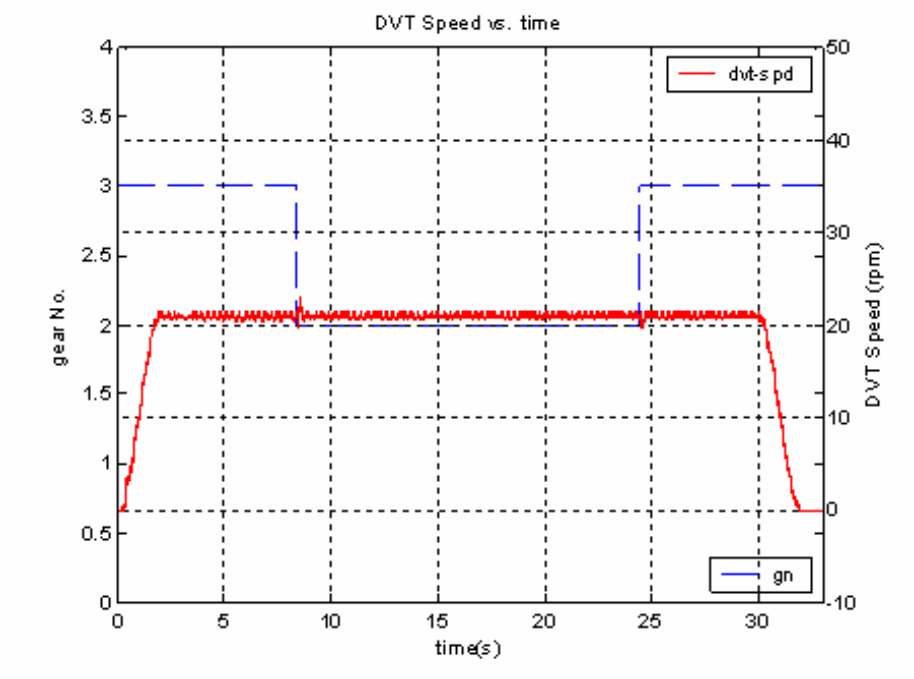


Figure 5.6, Two boundaries, hybrid switch signal (half load)

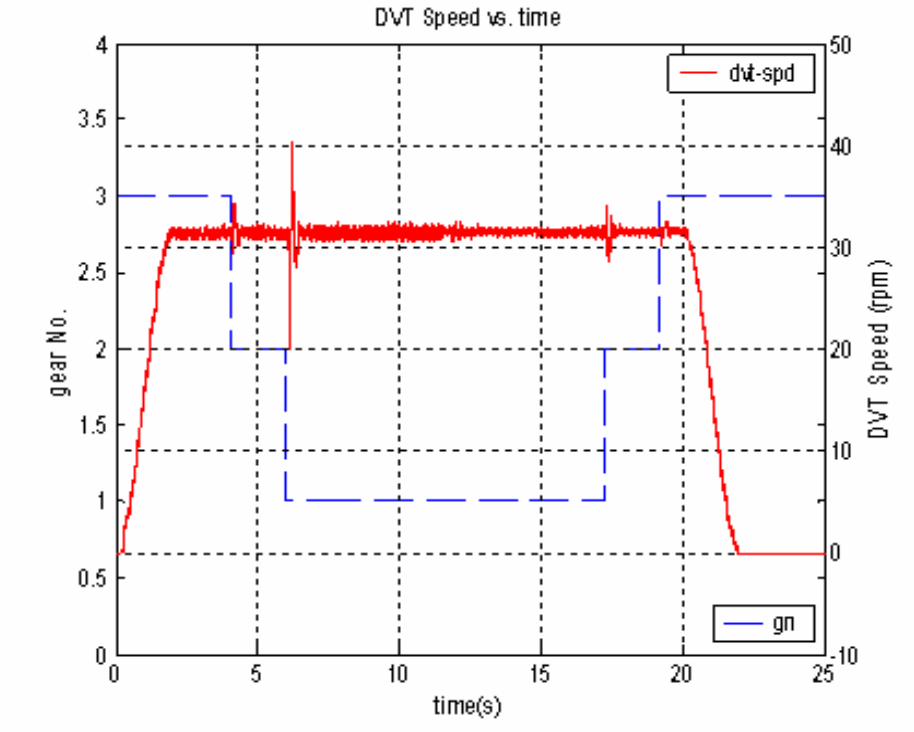


Figure 5.7, Two boundaries, hybrid switch signal (full load)

Position Error

Five different load conditions are tested using the proposed control schemes to evaluate the system performance in terms of steady state position error. The load conditions are zero load, 25% load, half load, 75% load and full load. The load conditions are emulated by the dynamometer. The steady state position error and tracking error curve for full load is given in Figure 5.8. From this figure, the maximum tracking error is about 44 degrees when the TBA is shifting from the 2^{nd} to the 1^{st} gear. And the steady state position error of the TBA is zero degrees. The position is transformed from the encoder reading, and the encoder output resolution is 1024 counts/revolution, so strictly speaking, the steady state position error is less than one count of the encoder reading, which is about 0.35 degrees. The position errors for all load conditions are summarized in Table 5.6, and tracking errors are summarized in Table 5.7.

DVT Speed and Disturbances

Due to the unmatched speed between the BLDC and the load, significant speed disturbances are observed. This phenomenon can be found in the speed curve of the BLDC and the load. When the load has its maximum value, the speed curves for the BLDC and the DVT are shown in Figure 5.9 and Figure 5.10 respectively.

The maximum disturbance of the DVT speed happens when the DVT shifts from the 2^{nd} gear to the 1^{st} gear. This effect is more clearly shown in Figure 5.11, which is a locally amplified version of Figure 5.10.

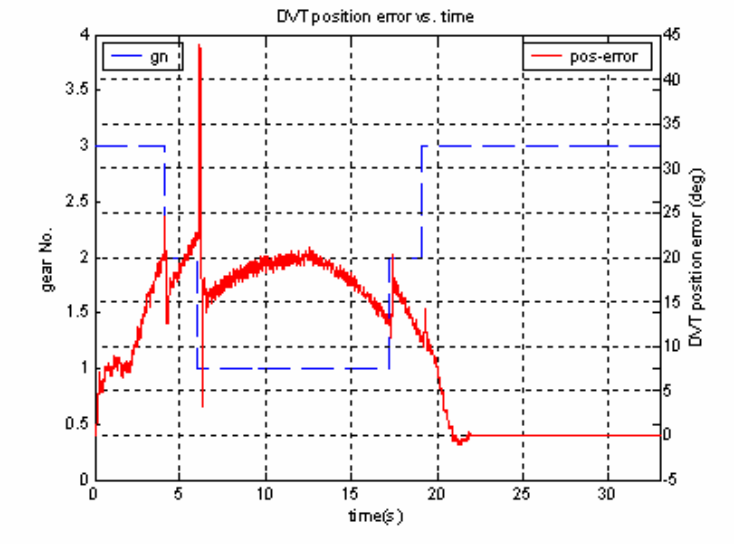


Figure 5.8, Position error with full load

Table 5.6, Position errors for different loads

Load	Angular Position Error at DVT (counts)	Angular Position Error at DVT (deg)	Angular Position Error at arm (deg)	Position Error at arm end (mm)
0	3	<1.06	<0.0053	<0.17
25%	3	<1.06	<0.0053	<0.17
50%	3	<1.06	<0.0053	<0.17
75%	0	<0.35	<0.0018	<0.06
100%	0	<0.35	<0.0018	<0.06

Table 5.7, Maximum tracking errors for different loads

Load	Tracking Error at DVT (counts)	Tracking Error at DVT (deg)	Tracking Error at arm (deg)	Tracking Error at arm end (mm)
0	40	<14.07	<0.0704	<2.22
25%	61	<21.45	<0.1073	<3.38
50%	71	<24.97	<0.1249	<3.93
75%	122	<42.90	<0.2145	<6.75
100%	125	<43.95	<0.2198	<6.92

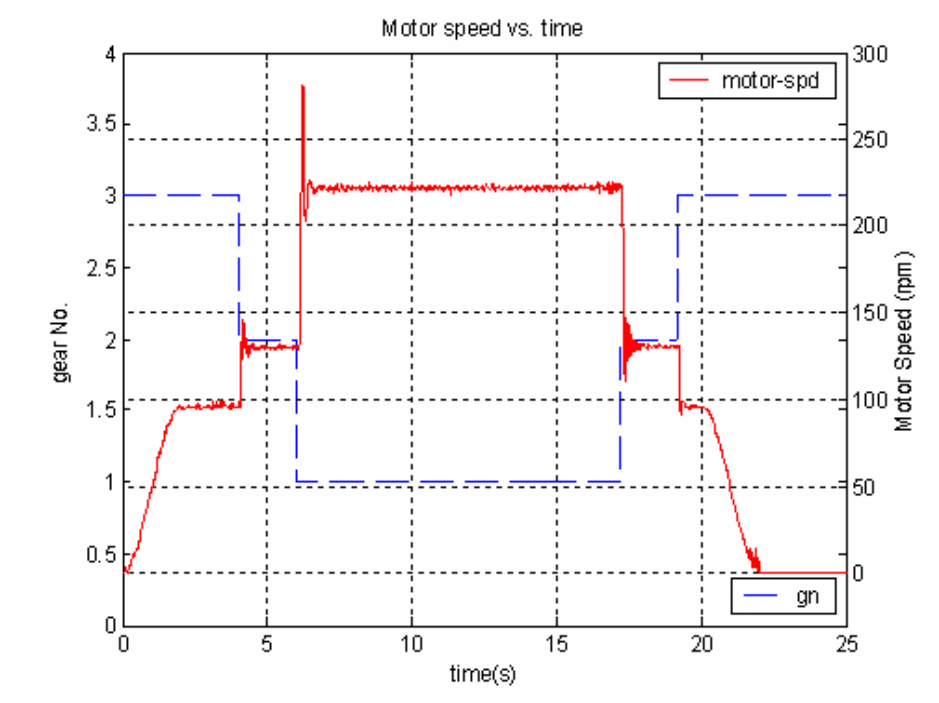


Figure 5.9, BLDC speed with full load

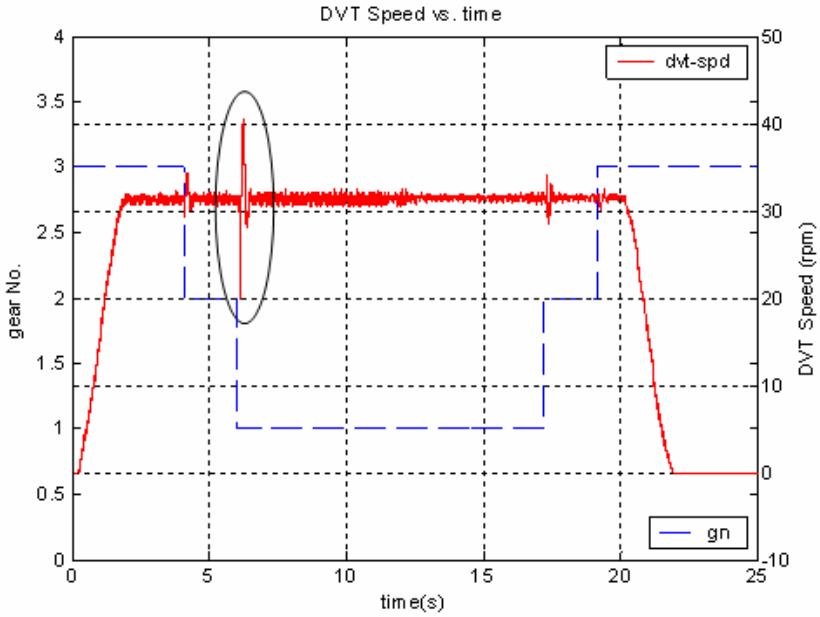


Figure 5.10, DVT speed with full load

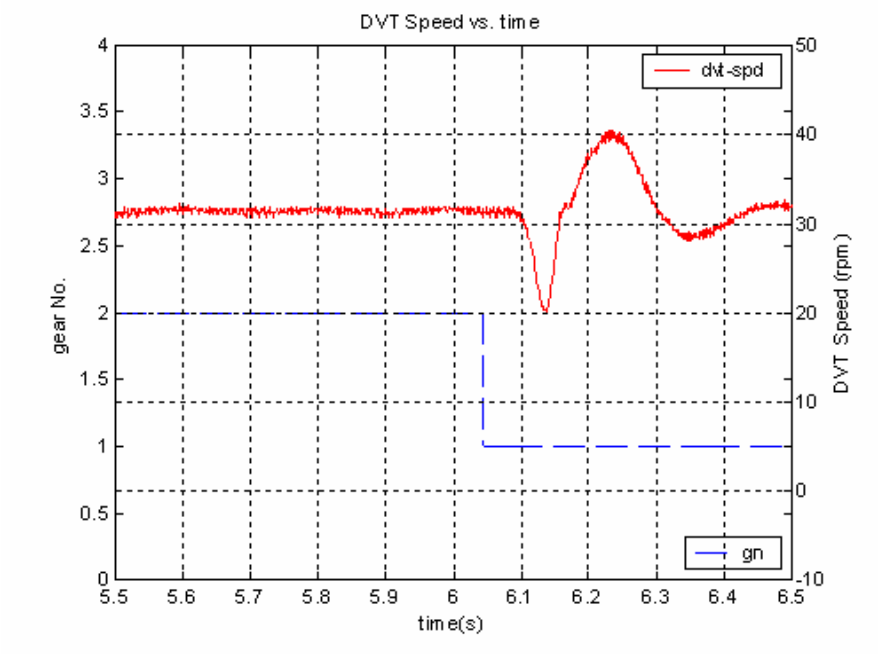


Figure 5.11, DVT speed disturbance near gear change from 2nd gear to 1st

From Figure 5.10 and Figure 5.11, when the TBA shifts from 2nd gear to the 1st gear, a large disturbance is observed; much smaller disturbances are observed for all three other shift actions.

Even though the magnitude of the disturbance is high, the system is still able to recover from the disturbance quick enough and still tracks both the velocity and position after the shift.

The cause of this abnormally large disturbance is probably related to the mechanical design of the DVT gears, since the last stage of the DVT is vastly different from the first two in terms of bearing support. Notice that a much smaller disturbance is observed when the shift is from 1st to 2nd gear. This shift has a speed difference that is comparable to the previous shift, which can serve as support for the above conclusion that the abnormally large disturbance results from the mechanical design of the system.

One can also observe Figure 5.11 from that there is a ~0.05 second lag between the shift command and the shift action. We can conclude that the shift action can be executed in about 0.05 seconds after the command is sent out. This delay adds an additional shift disturbance to the TBA the system.

Real Time Performance

As mentioned in Chapter 3 and Chapter 4, the servo control loop runs at 1000Hz, which is critical for the validity of the digital PID controller.

In this research, the real time performance is evaluated based on the measured jitter values. The jitter is defined as the difference between the commanded sample period and the actual sample period. Table 5.8 shows the jitter values from the results of two consecutive tests. From the table, we can see that the average loop period is almost exactly the same as the commanded loop period (1milliseconds). The maximum absolute value of the jitter is 40 microseconds, which is about 4% of the commanded period. In conclusion, the sample period requirement is met with a negligible time delay.

Load Torques

Since the load torque is used in the shift strategy, a close look at the torque curve gives better understanding of the shift strategy.

Table 5.8, Real time performance and jitter results

Test	Mean (microsecond)	Standard deviation (microseconds)	Maximum (microseconds)	Minimum (microseconds)
No.1	-0.0019	2.5948	13	-33
No.2	-0.0024	2.5730	13	-40

BLDC torque and dynamometer torque curves for full load are shown in Figure 5.12. There are three curves in the figure: the dashed line is the gear number, the upper curve is BLDC torque, and the lower curve is dynamometer torque. As can be observed from the curves, the load torque was tracked very well, and the shift actions can be explained more clearly by the following:

1. The DVT starts from 3rd gear with the lowest ratio, with the arm vertical down;
2. The DVT shifts to 2nd gear when the load torque goes above 5 N-m.
3. The DVT shifts to 1st gear when the load torque goes above 8 N-m.
4. The DVT shifts to 2nd gear when the load torque goes below 6 N-m.
5. The DVT shifts to 3rd gear when the load torque goes below 4 N-m.
6. The TBA tracks the trajectory and stops with arm at a vertical up position.

Experimental Results Summary

Based on the experimental results, the following conclusions can be drawn:

1. The prototype TBA system dynamic performance is stable and suitable for precise servo control such as that used in robotic systems.
2. The shift strategy, or switch signal, is transition stable.
3. The proposed control system design and implementation meets all the TBA control system performance requirements.
4. The experimental results also suggest that a further examination of the effects of DVT mechanical design on the transient disturbances is necessary.

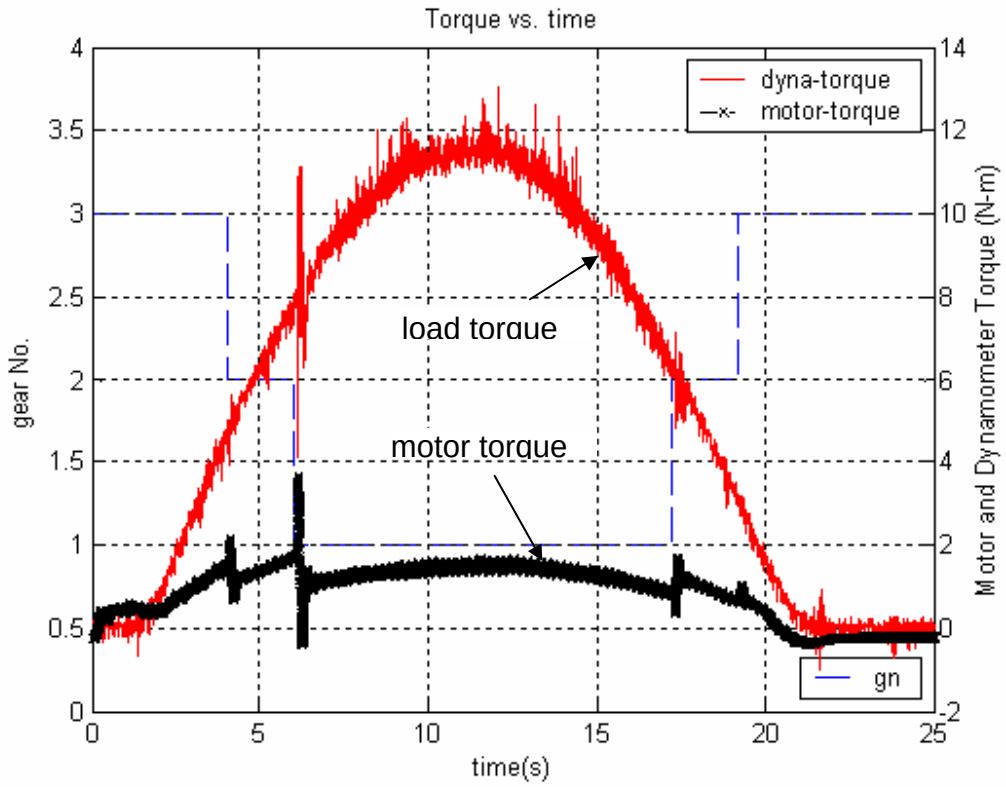


Figure 5.12, BLDC servo motor and dynamometer load torque (full load)

CHAPTER 6

Contributions and Future Work

Contributions

In this dissertation, a number of fundamental problems associated with the TBA prototype and its dynamic operation were studied analytically and experimentally.

1. A hybrid dynamic system model was set up for the TBA system, refer to Chapter 2, which is a set of three 2^{nd} order linear time invariant systems; the model is the first example of the hybrid system theory applied to the TBA system in literature.
2. A supervisory controller was developed for the TBA system. A transition stable switch strategy was identified analytically by finding time invariant and state space invariant switch signals as shown in Chapter 3, and it was demonstrated experimentally as reported in Chapter 5. By doing this, this dissertation addresses the essential hybrid problem which is the design of the associated supervisory controller for continuous systems [36].
3. The control strategy successfully achieved switch signal transition stability. The approach is to find a control strategy whose switch signal is both time invariant and state space invariant. In this research, a heuristic method was adopted to generate a transition stable switch signal. First, a two shift boundary scheme was used such that up and down shift signals are

generated on different conditions; thus, a state space stable switch signal can be found by using properly tuned shift boundaries. Second, a time invariant switch signal was found by limiting the time between two adjacent switch signals. By using a control strategy involving two shift boundaries and a time constraint, a transition stable signal was achieved, which was shown analytically and experimentally.

4. A sufficient condition for TBA stability was also established by solving the second of the three fundamental problems of hybrid control systems formulated in [15]. The method and results can be extended to a wide range of electrical and mechanical systems involving multiple continuous systems and discrete state changes. The sufficient condition for plant stability was established by finding a common feedback controller to stabilize all three subsystems. Under this condition, it was shown that a common Lyapunov function exists for the TBA; thus, stability under asynchronous switching is achieved for the TBA.
5. A PC based real time control software platform was also established for the class of mechanical systems that includes TBA's. The real time control software was set up using RTAI, which provides a real time patch for general Linux and user space real time programming. The software used a multithread design in order to meet the real time performance requirement of different subtasks. The experimental results showed that the proposed software meets all the system design requirements and yet is sufficiently flexible for future controller upgrades. With the proposed real time control software framework, significant performance

improvements were achieved compared with a LabVIEW / Windows XP implementation during the TBA feasibility test [1, 57] .

6. The particular real time control software framework developed here is based on open source software and is the first in literature and is expressly designed to fit the needs for high bandwidth mechanical systems R&D. The multi-thread real time control software framework, by providing deterministic time, task scheduling, inter process communications, Ethernet, and a GUI, can be used as a reference framework for a wide range of PC based control problems where real time performance is required.

In summary, the research successfully addressed the most fundamental control issues associated with the TBA prototype including modeling, control, simulation, and experiment verification. It has expanded the fundamental understanding of the TBA system control and similar hybrid dynamic systems. The methods and results of this dissertation will contribute to the hybrid system control knowledge base and the real time control software design practice in many applications.

Future Work

Although fundamental problems associated with the TBA control were addressed in this dissertation, much research remains pertaining to theoretical and implementation issues. Specially, future works should:

1. Explore other optimal or suboptimal control scheme. One optimal based control scheme for minimal transient response was formulated in this

research. Another possible objective for optimization can be to maximize servo power output control. One of the challenges of these optimal control methods lies in the existence of plant constraints. For example, the servo motor has speed and torque saturations. Another difficulty is to sustain real time loop speeds in the presence of the large calculation overhead of more complex control algorithm.

2. Analyze and evaluate the effects of the parameter uncertainty of the TBA on the system performance. In this research, a conservative approach was used to design the feedback controller. For example, all the closed loop poles not only have negative real parts, but these poles are far away from the origin. This conservative approach can maintain system stability and keep system performance within an acceptable range under parameter uncertainty. An alternative approach is to design a controller using robust control theory, which usually gives a high order controller. In addition to design controller for system with parameter uncertainty, robust control theory can also be used to attack plant disturbance, sensor noise rejection.
3. Asymmetric transient responses were observed in experiments when shifting into and out of the 1^{st} gear. A further examination of the mechanical design of the TBA will be helpful to identify the cause of this problem. The compliance characteristics of the bearing and shaft structures the planetary gears should definitely be studied further.

LIST OF REFERENCES

REFERENCES

[1] W. R. Hamel, D. A. Lumsdaine, D. S. Douglass, D. S. Kim, K. P. Brown, S. Sridharan, R. Zhou, K. Ganti, and A. Srikantaiah, "Transmission-based eletrical servoactuators," 2003.

[2] J. Chiasson, *Modeling and High Performance Control of Electric Machines*: Wiley-IEEE Press, 2005.

[3] R. Qu, M. Aydin, and T. A. Lipo, "Performance Comparison of Dual-rotor Radial-flux and Axial-flux Permanent Magnet BLDC Machines " presented at Electrical Machines and Drives Conference, 2003.

[4] M. Bodson and J. Chiasson, "A systematic approach to selecting flux references for torque maxmization in induction motors," *Control System Magzine, IEEE*, vol. 3, pp. 388-397, 1995.

[5] H. Baruh, *Analytical Dynamics*: McGraw-Hill Science/Engineering/Math, 1998.

[6] M. Bugeja, "Non-Linear Swing-Up and Stabilization Control of and Inverted Pendulum System," presented at EUROCON, Ljubljana,Slovenia, 2003.

[7] K. J. Aström and K. Furuta, "Swing up a pendulum by energy control," presented at 13th IFAC World Congress, San Francisco, CA, 1996.

[8] K. Yoshida, "Swing-up Control of an Inverted Pendulum by Energy Methods " presented at American Control Conference, San Diego, CA, 1999.

[9] R. Alur, C. A. Courcoubetis, T. A. Henzinger, and P.-H. Ho, "Hybrid automata: An algorithmic approach to specification and verification of hybrid systems," in *Lecture Notoes in Computer Science*, vol. 736, 1993, pp. 209-229.

[10] P. J. Antsaklis, "A brief introduction to the theory and applications of hybrid systems," presented at IEEE, Special Issue on Hybrid Systems: Theory and Applications, 2000.

[11] A. Bemporad, A. Giua, and C. Seatzu, "Synthesis of state-feedback optimal controllers for switched linear systems," presented at 41th IEEE Conference on Decision and Control, 2002.

[12] M. S. Branicky, "Stability of Switched and Hybrid Systems," presented at 33rd Conference on Decision and Control, Lake Buena Vista, FL, 1994.

[13] M. S. Branicky, "Multiple Lyapunov Functions and Other Analysis Tools for Switched and Hybrid Systems " *IEEE Transaction Automatic Control*, vol. 43, pp. 475-482, 1998.

[14] W. P. Dayawansa and C. F. Martin, "A converse lyapunov theorem for a class of dynamical sysytems which undergo switching," *IEEE Transaction Automatic Control*, vol. 44, pp. 751-760, 1999.

[15] D. Liberzon and A. S. Morse, "Basic Problems in Stability and Design of Switched systems," *Control System Magazine, IEEE*, vol. 19, pp. 59-70, 1999.

[16] E. Litsyn, Y. V. Nepomnyashchikh, and A. Ponosov, "Hybrid dynamic systems vs. ordinary differential equations: Examples of "pathlogical" behavior," in *Electronic Journal of Qualitative Theory of Differential Equations* 2000, pp. 1-10.
[17] D. Mignone, G. Ferrari-Trecate, and M. Morari, "Stability and Stabilization of Piecewise Affine and Hybrid Systems: an LMI Approach," presented at Decision and Control, IEEE, Sydney, NSW Australia, 2000.
[18] S. Pettersson and B. Lennartson, "Stability and Robustness for hybrid systems," presented at IEEE conference on Decision and Control, Kobe, 1996.
[19] Z. Sun and S. S. Ge, *Switched Linear Systems*. London: Springer-Verlag London Limited, 2005.
[20] K. S. Narendra and J. Balakrishnan, "A common lyapunov function for stable LTI systems with commuting A-matrices," *IEEE Transaction Automatic Control*, vol. 39, pp. 2469-2471, 1994.
[21] T. Ooba and Y. Funahashi, "Two conditions concerning common quadratic lyapunov functions for linear systems," *IEEE Transaction Automatic Control*, vol. 42, pp. 719-721, 1997.
[22] A. V. Savkin and R. J. Evans, *Hybrid Dynamical Systems*. Boston: Birkhaüser Boston, 2002.
[23] R. N. Shorten and K. S. Narendra, "A sufficient condition for the existense of a common lyapunov function for two second order linear systems," presented at IEEE Conference on Decision and Control, San Diago, California, 1997.
[24] R. N. Shorten and K. S. Narendra, "Necessary and sufficient conditions for the existence of a common quadratic lyapunov function for M stable second order linear time invariant systems," presented at American Control Conference, Chicago, Illinois, 2000.
[25] S. Pettersson and B. Lennartson, "Stabiltiy of Hybrid System Using LMIs-A Gear-Box Application," *Lecture Notes in Computer Science*, vol. 1790, pp. 381, 2000.
[26] S.-t. Cho, S. Jeon, H.-S. Jo, J.-M. Lee, and Y.-I. Park, "A development of shift control algorithm for improving the shift characteristics of the automated manual transmission in the hybrid drivetrain," *Int. J. of Vehicle Design*, vol. 26, pp. 469-486, 2001.
[27] A. Bemporad, P. Borodani, and M. Mannelli, "Hybrid control of an automotive robotized gearbox for reduction of consumptions and emmission," *Lecture Notes in Computer Science*, pp. 81-96, 2003.
[28] M. C. Turner and D. J. Walker, "Linear quadratic bumpless transfer," *Automatica*, vol. 36, pp. 1089-1101, 2000.
[29] M. Schinkel, W.-H. Chen, and A. Rantzer, "Optimal control for systems with varying sampling rate," presented at American Control Conference, 2002.
[30] T. Minowa, T. Ochi, H. Kuroshi, and K.-Z. Liu, "Smooth gear shift control technology for clutch-to-clutch shifting," *SAE Paper*, vol. 1999-01-1054, 1999.

[31] A. Bemporad, "Modeling, control, and reachability analysis of discrete time hybrid systems," 2003.
[32] T. A. Henzinger, "The theory of hybrid automata," *11th Annual IEEE Symposium on Logic in Computer Science*, pp. 278-292, 1996.
[33] J. M. Davoren and A. Nerode, "Logics for hybrid automata," *proceedings of the IEEE*, vol. 88, pp. 985-1010, 2000.
[34] D. Liberzon and R. Tempo, "Gradient algorithms for finding common lyaponov functions," presented at IEEE Conference on Decision and Control, Maui,Hawaii, 2003.
[35] X. Xu and P. J. Antsaklis, "Stability of Second-order LTI Switched Systems," ISIS Group at the University of Nortre Dame 1999 1999.
[36] X. D. Koutsoukos, P. J. Antsaklis, J. A. Stiver, and M. D. Lemmon, "Supervisory Control of Hybrid Systems," *Proceedings of the IEEE*, vol. 88, pp. 1026-1049, 2000.
[37] M. C. Turner and D. J. Walker, "Linear Quadratic Bumpless Transfer," *Automatica*, vol. 36, pp. 1089-1101, 2000.
[38] M. C. Turner and D. J. Walker, "Modified Linear Quadratic Bumpless Transfer," *American Control Conference*, vol. 4, pp. 2285-2289, 1999.
[39] J. V. d. Vegte, *Feedback Control Systems*, 3rd ed: Prentice Hall, Inc., 1994.
[40] K. J. Åström and T. Hägglund, "PID control," in *The Control Handbook*, W. S. levine, Ed.: CRC press, IEEE press, 1996, pp. 198-208.
[41] K. J. Aström and T. Hägglund, "PID control," in *The Control Handbook*, W. S. levine, Ed.: CRC press, IEEE press, 1996, pp. 198-208.
[42] T. Hu and Z. Lin, *Control Systems with Actuator Saturation*: Birkhaüser Boston, 2001.
[43] M. Masatoshi and Y. Tomoshige, "Research of Engine Optimal Control," *IHI Engineering Review*, vol. 38, pp. 70-73, 2005.
[44] F. L. Lewis and V. L. Syrmos, *Optimal control*, 2 ed. New York: Wiley-Interscience, 1995.
[45] K. Zhou and J. C. Doyle, *Essentials of Robust Control*. Upper Saddle River, NJ: Prentice-Hall, 1998.
[46] M. J. Bach, *The design of the unix operating system* 1ed: Prentice Hall PTR, 1986.
[47] D. P. Bovert and M. Cesati, *Understanding the Linux Kernel*, 2nd ed: O'Reilly, 2002.
[48] P. N. Leroux, "RTOS versus GPOS," *Embedde Computing Design*, 2005.
[49] I. Sun Microsystems, "Multithreaded Programming Guide," 2002.
[50] B. Ip, "Performance analysis of VxWorks and RTLinux," presented at the First Embedded Software Contest, Korea, 2003.
[51] RTAI, "www.rtai.org."
[52] P. Cloutier, "DIAPM-RTAI position paper," RTSS 2000-Real Time Operating Systems Workshop, 2000.
[53] P. Mantegazza, "DIAPM RTAI for Linux: WHYs, WHATs, and HOWs," presented at Real Time Linux Workshop, Vienna University of Technology, 1999.

[54] Comedi, "www.comedi.org."
[55] L. Dozio and P. Mantegazza, "Real Time Distributed Control Systems using RTAI," presented at IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2003.
[56] M. S. Santina, A. R. Stubberud, and G. H. Hostetter, "Sample-Rate Selection," in *The Control Handbook*, W. S. Levine, Ed.: CRC Press, IEEE Press, 1996, pp. 313-321.
[57] W. R. Hamel, S. Kim, R. Zhou, and A. Lumsdaine, "Design and testing of a prototype transmission-based robot servoactuator," presented at 2004 IEEE Conference on Robotics and Automation, 2004.

APPENDIX

APPENDIX

DVT model

3rd gear derivation: (Brake #1, lowest ratio)

$$J_s\ddot{\theta}_s = T_{in} + r_{s1}\lambda_1 - r_{s1}\lambda_2 + r_{s2}\lambda_3 - r_{s2}\lambda_4 + r_{s3}\lambda_5 - r_{s3}\lambda_6$$

$$3J_{p1}\ddot{\theta}_{p1} = r_{p1}\lambda_1$$

$$\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} = -T_{b2} - \left(r_{c1} - r_{p1}\right)\lambda_1 + \left(r_{s1} + r_{r1}\right)\lambda_2 - r_{r2}\lambda_4$$

$$J_{r1}\ddot{\theta}_{r1} = -T_{b1} - r_{r1}\lambda_2$$

$$3J_{p2}\ddot{\theta}_{p2} = r_{p2}\lambda_3$$

$$\left(3m_{p2}r_{c2}^2 + J_{c2} + J_{r3}\right)\ddot{\theta}_{r3} = -T_{b3} - \left(r_{c2} - r_{p2}\right)\lambda_3 + \left(r_{s2} + r_{r2}\right)\lambda_4 - r_{r3}\lambda_6$$

$$3J_{p3}\ddot{\theta}_{p3} = r_{p3}\lambda_5$$

$$\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} = -T_{out} - \left(r_{c3} - r_{p3}\right)\lambda_5 + \left(r_{s3} + r_{r3}\right)\lambda_6$$

$$J_s\ddot{\theta}_s - r_{s1}\frac{3J_{p1}}{r_{p1}}\ddot{\theta}_{p1} - r_{s2}\frac{3J_{p2}}{r_{p2}}\ddot{\theta}_{p2} - r_{s3}\frac{3J_{p3}}{r_{p3}}\ddot{\theta}_{p3} = T_{in} + r_{s1}\frac{T_{b1}}{r_{r1}} - r_{s2}\lambda_4 - r_{s3}\lambda_6 \quad (1)$$

$$\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} + \left(r_{c1} - r_{p1}\right)\frac{3J_{p1}}{r_{p1}}\ddot{\theta}_{p1} = -\left(r_{s1} + r_{r1}\right)\frac{T_{b1}}{r_{r1}} - r_{r2}\lambda_4 \quad (2)$$

$$\left(3m_{p2}r_{c2}^2 + J_{c2} + J_{r3}\right)\ddot{\theta}_{r3} + \left(r_{c2} - r_{p2}\right)\frac{3J_{p2}}{r_{p2}}\ddot{\theta}_{p2} = \left(r_{s2} + r_{r2}\right)\lambda_4 - r_{r3}\lambda_6 \quad (3)$$

$$\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} + \left(r_{c3} - r_{p3}\right)\frac{3J_{p3}}{r_{p3}}\ddot{\theta}_{p3} = -T_{out} + \left(r_{s3} + r_{r3}\right)\lambda_6 \quad (4)$$

(1) + (2) + (3) + (4)

$$J_s\ddot{\theta}_s - r_{s1}\frac{3J_{p1}}{r_{p1}}\ddot{\theta}_{p1} - r_{s2}\frac{3J_{p2}}{r_{p2}}\ddot{\theta}_{p2} - r_{s3}\frac{3J_{p3}}{r_{p3}}\ddot{\theta}_{p3} + \left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} + \left(r_{c1} - r_{p1}\right)\frac{3J_{p1}}{r_{p1}}\ddot{\theta}_{p1} +$$

$$\left(3m_{p2}r_{c2}^2 + J_{c2} + J_{r3}\right)\ddot{\theta}_{r3} + \left(r_{c2} - r_{p2}\right)\frac{3J_{p2}}{r_{p2}}\ddot{\theta}_{p2} + \left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} + \left(r_{c3} - r_{p3}\right)\frac{3J_{p3}}{r_{p3}}\ddot{\theta}_{p3} =$$

$$T_{in} + r_{s1}\frac{T_{b1}}{r_{r1}} - \left(r_{s1} + r_{r1}\right)\frac{T_{b1}}{r_{r1}} - T_{out} - r_{r2}\lambda_4 - r_{s2}\lambda_4 + \left(r_{s2} + r_{r2}\right)\lambda_4 + \left(r_{s3} + r_{r3}\right)\lambda_6 - r_{s3}\lambda_6 - r_{r3}\lambda_6$$

$$J_s\ddot{\theta}_s + \left(3m_{p2}r_{c2}^2 + J_{c2} + J_{r3}\right)\ddot{\theta}_{r3} + \left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} + \left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} = T_{in} - T_{b1} - T_{out}$$

$$T_{b1} = T_{in} - T_{out} - J_s\ddot{\theta}_s - \left(3m_{p2}r_{c2}^2 + J_{c2} + J_{r3}\right)\ddot{\theta}_{r3} - \left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} - \left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} \quad (*)$$

from (4), $\lambda_6 = \frac{1}{(r_{s3}+r_{r3})}\left(\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} + \left(r_{c3}-r_{p3}\right)\frac{3J_{p3}}{r_{p3}}\ddot{\theta}_{p3} + T_{out}\right)$

from (2), $\lambda_4 = -\frac{1}{r_{r2}}\left(\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} + \left(r_{c1}-r_{p1}\right)\frac{3J_{p1}}{r_{p1}}\ddot{\theta}_{p1} + \left(r_{s1}+r_{r1}\right)\frac{T_{b1}}{r_{r1}}\right)$

substitute λ_4 and λ_6 into (3)

$$\left(3m_{p2}r_{c2}^2 + J_{c2} + J_{r3}\right)\ddot{\theta}_{r3} + \left(r_{c2}-r_{p2}\right)\frac{3J_{p2}}{r_{p2}}\ddot{\theta}_{p2} = \left(r_{s2}+r_{r2}\right)\lambda_4 - r_{r3}\lambda_6 \quad (3)$$

$$\left(3m_{p2}r_{c2}^2 + J_{c2} + J_{r3}\right)\ddot{\theta}_{r3} + \frac{\left(r_{c2}-r_{p2}\right)}{r_{p2}}3J_{p2}\ddot{\theta}_{p2} + \frac{r_{r3}}{\left(r_{s3}+r_{r3}\right)}\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} +$$

$$\frac{r_{r3}}{\left(r_{s3}+r_{r3}\right)}\frac{\left(r_{c3}-r_{p3}\right)}{r_{p3}}3J_{p3}\ddot{\theta}_{p3} + \frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2}$$

$$+\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{c1}-r_{p1}\right)}{r_{p1}}3J_{p1}\ddot{\theta}_{p1} = -\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}T_{b1} - \frac{r_{r3}}{\left(r_{s3}+r_{r3}\right)}T_{out} \quad (6)$$

$$T_{b1} = T_{in} - T_{out} - J_s\ddot{\theta}_s - \left(3m_{p2}r_{c2}^2 + J_{c2} + J_{r3}\right)\ddot{\theta}_{r3} - \left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} - \left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} \quad (*)$$

substitute (*) into (6)

$$-\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}T_{b1} = -\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}T_{in} + \frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}T_{out} +$$

$$\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}J_s\ddot{\theta}_s + \frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}\left(3m_{p2}r_{c2}^2 + J_{c2} + J_{r3}\right)\ddot{\theta}_{r3} +$$

$$\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} + \frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3}$$

$$\left(3m_{p2}r_{c2}^2 + J_{c2} + J_{r3}\right)\ddot{\theta}_{r3} + \frac{\left(r_{c2}-r_{p2}\right)}{r_{p2}}3J_{p2}\ddot{\theta}_{p2} + \frac{r_{r3}}{\left(r_{s3}+r_{r3}\right)}\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} +$$

$$\frac{r_{r3}}{\left(r_{s3}+r_{r3}\right)}\frac{\left(r_{c3}-r_{p3}\right)}{r_{p3}}3J_{p3}\ddot{\theta}_{p3} + \frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} + \frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{c1}-r_{p1}\right)}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}$$

$$= -\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}T_{in} + \frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}T_{out} + \frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}J_s\ddot{\theta}_s +$$

$$\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}\left(3m_{p2}r_{c2}^2 + J_{c2} + J_{r3}\right)\ddot{\theta}_{r3} + \frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} +$$

$$\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} - \frac{r_{r3}}{\left(r_{s3}+r_{r3}\right)}T_{out}$$

$$
\begin{aligned}
&-\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}J_s\ddot{\theta}_s \\
&+\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3}-\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3} \\
&+\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{c1}-r_{p1}\right)}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}+\frac{\left(r_{c2}-r_{p2}\right)}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}+\frac{r_{r3}}{\left(r_{s3}+r_{r3}\right)}\frac{\left(r_{c3}-r_{p3}\right)}{r_{p3}}3J_{p3}\ddot{\theta}_{p3} \\
&+\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\left(3m_{p1}r_{c1}^2+J_{c1}+J_{r2}\right)\ddot{\theta}_{r2}-\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}\left(3m_{p1}r_{c1}^2+J_{c1}+J_{r2}\right)\ddot{\theta}_{r2} \\
&+\frac{r_{r3}}{\left(r_{s3}+r_{r3}\right)}\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}-\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3} \\
&=-\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}T_{in}+\left(\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}-\frac{r_{r3}}{\left(r_{s3}+r_{r3}\right)}\right)T_{out}
\end{aligned}
$$

$$
\begin{aligned}
&-\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}J_s\ddot{\theta}_s \\
&+\left(1-\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}\right)\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3} \\
&+\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}+\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}+\frac{r_{r3}}{\left(r_{s3}+r_{r3}\right)}\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3} \\
&+\left(\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}-\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}\right)\left(3m_{p1}r_{c1}^2+J_{c1}+J_{r2}\right)\ddot{\theta}_{r2} \\
&+\left(\frac{r_{r3}}{\left(r_{s3}+r_{r3}\right)}-\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}\right)\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3} \\
&=-\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}T_{in}+\left(\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}-\frac{r_{r3}}{\left(r_{s3}+r_{r3}\right)}\right)T_{out}
\end{aligned}
$$

$$-p_1 p_2 J_s \ddot{\theta}_s + (1 - p_1 p_2) J_{eq2} \ddot{\theta}_{r3} + 3 p_1 m_1 J_{p1} \ddot{\theta}_{p1} + 3 m_2 J_{p2} \ddot{\theta}_{p2} + 3 p_3 m_3 J_{p3} \ddot{\theta}_{p3} +$$
$$p_2 (1 - p_1) J_{eq1} \ddot{\theta}_{r2} + (p_3 - p_1 p_2) J_{eq3} \ddot{\theta}_{c3} = -p_1 p_2 T_{in} + (p_1 p_2 - p_3) T_{out}$$

$$\frac{(r_{s1} + r_{r1})}{r_{r1}} \equiv p_1, \frac{(r_{s2} + r_{r2})}{r_{r2}} \equiv p_2, \frac{r_{r3}}{(r_{s3} + r_{r3})} \equiv p_3$$
$$\frac{r_{s1}}{r_{p1}} \equiv m_1, \frac{r_{s2}}{r_{p2}} \equiv m_2, \frac{r_{s3}}{r_{p3}} \equiv m_3$$
$$(3 m_{p1} r_{c1}^2 + J_{c1} + J_{r2}) \equiv J_{eq1}, (3 m_{p2} r_{c2}^2 + J_{c2} + J_{r3}) \equiv J_{eq2}, (3 m_{p3} r_{c3}^2 + J_{c3}) \equiv J_{eq3}$$

$$\dot{\theta}_s r_{s1} - \dot{\theta}_{r2} r_{s1} + \dot{\theta}_{p1} r_{p1} = 0 \Rightarrow \dot{\theta}_{p1} = (\beta_1 - 1) \frac{r_{s1}}{r_{p1}} \dot{\theta}_s$$

$$-\dot{\theta}_s r_{s1} + \dot{\theta}_{r2} (r_{r1} + r_{s1}) = 0 \Rightarrow \dot{\theta}_{r2} = \frac{r_{s1}}{(r_{r1} + r_{s1})} \dot{\theta}_s \equiv \beta_1 \dot{\theta}_s$$

$$\dot{\theta}_s r_{s2} - \dot{\theta}_{r3} r_{s2} + \dot{\theta}_{p2} r_{p2} = 0 \Rightarrow \dot{\theta}_{p2} = (\alpha_1 - 1) \frac{r_{s2}}{r_{p2}} \dot{\theta}_s$$

$$-\dot{\theta}_s r_{s2} + \dot{\theta}_{r3} (r_{r2} + r_{s2}) - \dot{\theta}_{r2} r_{r2} = 0 \Rightarrow \dot{\theta}_{r3} = \frac{r_{r2} \frac{r_{s1}}{(r_{r1} + r_{s1})} + r_{s2}}{(r_{r2} + r_{s2})} \dot{\theta}_s \Rightarrow$$

$$\dot{\theta}_{r3} = \left(\frac{r_{r2} \frac{r_{s1}}{(r_{r1} + r_{s1})}}{(r_{r2} + r_{s2})} + \frac{r_{s2}}{(r_{r2} + r_{s2})} \right) \dot{\theta}_s \equiv \alpha_1 \dot{\theta}_s$$

$$\dot{\theta}_s r_{s3} - \dot{\theta}_{c3} r_{s3} + \dot{\theta}_{p3} r_{p3} = 0 \Rightarrow \dot{\theta}_{p3} = (\gamma_1 - 1) \frac{r_{s3}}{r_{p3}} \dot{\theta}_s$$

$$-\dot{\theta}_s r_{s3} + \dot{\theta}_{c3} (r_{r3} + r_{s3}) - \dot{\theta}_{r3} r_{r3} = 0 \Rightarrow \dot{\theta}_{c3} = \frac{\left(r_{s3} + \left(\frac{r_{r2} \frac{r_{s1}}{(r_{r1} + r_{s1})}}{(r_{r2} + r_{s2})} + \frac{r_{s2}}{(r_{r2} + r_{s2})} \right) r_{r3} \right)}{(r_{r3} + r_{s3})} \dot{\theta}_s \equiv \gamma_1 \dot{\theta}_s$$

$$\begin{pmatrix} -p_1 p_2 J_s + (1 - p_1 p_2) \alpha_1 J_{eq2} + 3 p_1 m_1 (\beta_1 - 1) m_1 J_{p1} + 3 m_2 (\alpha_1 - 1) m_2 J_{p2} + \\ 3 p_3 m_3 (\gamma_1 - 1) m_3 J_{p3} + p_2 (1 - p_1) \beta_1 J_{eq1} + (p_3 - p_1 p_2) \gamma_1 J_{eq3} \end{pmatrix} \ddot{\theta}_s$$
$$= -p_1 p_2 T_{in} + (p_1 p_2 - p_3) T_{out}$$

2nd gear derivation (brake #2, middle ratio)

$$J_s\ddot{\theta}_s = T_{in} + r_{s1}\lambda_1 - r_{s1}\lambda_2 + r_{s2}\lambda_3 - r_{s2}\lambda_4 + r_{s3}\lambda_5 - r_{s3}\lambda_6$$

$$\frac{3J_{p1}}{r_{p1}}\ddot{\theta}_{p1} = \lambda_1$$

$$0 = -T_{b2} - \left(r_{c1} - r_{p1}\right)\lambda_1 + \left(r_{s1} + r_{r1}\right)\lambda_2 - r_{r2}\lambda_4$$

$$-\frac{J_{r1}}{r_{r1}}\ddot{\theta}_{r1} = \lambda_2$$

$$\frac{3J_{p2}}{r_{p2}}\ddot{\theta}_{p2} = \lambda_3$$

$$\left(3m_{p2}r_{c2}^2 + J_{c2} + J_{r3}\right)\ddot{\theta}_{r3} = -\left(r_{c2} - r_{p2}\right)\lambda_3 + \left(r_{s2} + r_{r2}\right)\lambda_4 - r_{r3}\lambda_6$$

$$\frac{3J_{p3}}{r_{p3}}\ddot{\theta}_{p3} = \lambda_5$$

$$\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} = -T_{out} - \left(r_{c3} - r_{p3}\right)\lambda_5 + \left(r_{s3} + r_{r3}\right)\lambda_6$$

$$J_s\ddot{\theta}_s = T_{in} + \frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1} + \frac{r_{s1}}{r_{r1}}J_{r1}\ddot{\theta}_{r1} + \frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2} - r_{s2}\lambda_4 + \frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3} - r_{s3}\lambda_6$$

$$\frac{1}{r_{p1}}3J_{p1}\ddot{\theta}_{p1} = \lambda_1$$

$$0 = -T_{b2} - \frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1} - \frac{\left(r_{s1} + r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1} - r_{r2}\lambda_4$$

$$-\frac{1}{r_{r1}}J_{r1}\ddot{\theta}_{r1} = \lambda_2$$

$$\frac{1}{r_{p2}}3J_{p2}\ddot{\theta}_{p2} = \lambda_3$$

$$\left(3m_{p2}r_{c2}^2 + J_{c2} + J_{r3}\right)\ddot{\theta}_{r3} = -\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2} + \left(r_{s2} + r_{r2}\right)\lambda_4 - r_{r3}\lambda_6$$

$$\frac{1}{r_{p3}}3J_{p3}\ddot{\theta}_{p3} = \lambda_5$$

$$\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} = -T_{out} - \frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3} + \left(r_{s3} + r_{r3}\right)\lambda_6$$

$$J_s\ddot{\theta}_s-\frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}-\frac{r_{s1}}{r_{r1}}J_{r1}\ddot{\theta}_{r1}-\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}-\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}=T_{in}-r_{s2}\lambda_4-r_{s3}\lambda_6(1)$$

$$\frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}+\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1}=-T_{b2}-r_{r2}\lambda_4(2)$$

$$\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3}+\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}=\left(r_{s2}+r_{r2}\right)\lambda_4-r_{r3}\lambda_6(3)$$

$$\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}+\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}=-T_{out}+\left(r_{s3}+r_{r3}\right)\lambda_6(4)$$

add all the above equations

$$J_s\ddot{\theta}_s-\frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}-\frac{r_{s1}}{r_{r1}}J_{r1}\ddot{\theta}_{r1}-\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}-\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}+\frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}+\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1}+$$

$$\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3}+\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}+\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}+\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}=T_{in}-T_{b2}-T_{out}$$

$$J_s\ddot{\theta}_s+J_{r1}\ddot{\theta}_{r1}+\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3}+\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}=T_{in}-T_{b2}-T_{out}(5)$$

$$from(4)\quad \lambda_6=\frac{\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}+\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}+T_{out}}{\left(r_{s3}+r_{r3}\right)}$$

$$\text{from (2)}\quad \lambda_4=-\frac{\frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}+\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1}+T_{b2}}{r_{r2}}$$

λ_4,λ_6 into (3)

$$\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3}+\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}=-\left(r_{s2}+r_{r2}\right)\frac{\frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}+\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1}+T_{b2}}{r_{r2}}-$$

$$r_{r3}\frac{\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}+\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}+T_{out}}{\left(r_{s3}+r_{r3}\right)}$$

$\Rightarrow$

$$J_{eq2}\ddot{\theta}_{r3}+3m_2J_{p2}\ddot{\theta}_{p2}+p_3J_{eq3}\ddot{\theta}_{c3}+3m_3p_3J_{p3}\ddot{\theta}_{p3}+p_3T_{out}+3p_2m_1J_{p1}\ddot{\theta}_{p1}+p_1p_2J_{r1}\ddot{\theta}_{r1}=-p_2T_{tb2}$$

$$\frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}+\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1}+\frac{r_{r2}}{\left(r_{s2}+r_{r2}\right)}\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3}+\frac{r_{r2}}{\left(r_{s2}+r_{r2}\right)}\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}+$$

$$\frac{r_{r2}r_{r3}}{\left(r_{s2}+r_{r2}\right)\left(r_{s3}+r_{r3}\right)}\left(\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}+\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}+T_{out}\right)=-T_{b2}$$

$$J_s\ddot{\theta}_s-\frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}-\frac{r_{s1}}{r_{r1}}J_{r1}\ddot{\theta}_{r1}-\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}-\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}=T_{in}-r_{s2}\lambda_4-r_{s3}\lambda_6 (1)$$

$$\frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}+\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1}=-T_{b2}-r_{r2}\lambda_4 (2)$$

$$\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3}+\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}=\left(r_{s2}+r_{r2}\right)\lambda_4-r_{r3}\lambda_6 (3)$$

$$\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}+\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}=-T_{out}+\left(r_{s3}+r_{r3}\right)\lambda_6 (4)$$

add all the above equations

$$J_s\ddot{\theta}_s-\frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}-\frac{r_{s1}}{r_{r1}}J_{r1}\ddot{\theta}_{r1}-\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}-\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}+\frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}+\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1}+$$

$$\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3}+\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}+\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}+\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}=T_{in}-T_{b2}-T_{out}$$

$$J_s\ddot{\theta}_s+J_{r1}\ddot{\theta}_{r1}+\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3}+\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}=T_{in}-T_{b2}-T_{out} (5)$$

from(4) $\dfrac{\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}+\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}+T_{out}}{\left(r_{s3}+r_{r3}\right)}=\lambda_6$ into (3)

$$\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3}+\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}+r_{r3}\frac{\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}+\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}+T_{out}}{\left(r_{s3}+r_{r3}\right)}}{\left(r_{s2}+r_{r2}\right)}=\lambda_4$$

into (2)

$$\frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}+\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1}=-T_{b2}-\frac{r_{r2}}{\left(r_{s2}+r_{r2}\right)}\left(\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3}+\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}+\right.$$

$$\left.\frac{r_{r3}}{\left(r_{s3}+r_{r3}\right)}\left(\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}+\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}+T_{out}\right)\right)$$

$\Rightarrow$

$$\frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}+\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1}+\frac{r_{r2}}{\left(r_{s2}+r_{r2}\right)}\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3}+\frac{r_{r2}}{\left(r_{s2}+r_{r2}\right)}\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}+$$

$$\frac{r_{r2}r_{r3}}{\left(r_{s2}+r_{r2}\right)\left(r_{s3}+r_{r3}\right)}\left(\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}+\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}+T_{out}\right)=-T_{b2}$$

$$\frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}+\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1}+\frac{r_{r2}}{\left(r_{s2}+r_{r2}\right)}\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3}+$$

$$\frac{r_{r2}}{\left(r_{s2}+r_{r2}\right)}\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}+\frac{r_{r2}}{\left(r_{s2}+r_{r2}\right)}\frac{r_{r3}}{\left(r_{s3}+r_{r3}\right)}$$

$$\left(\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}+\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}+T_{out}\right)=-T_{b2}\text{(*) into (5)}$$

$$J_s\ddot{\theta}_s+J_{r1}\ddot{\theta}_{r1}+\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3}+\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}=$$

$$T_{in}+\frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1}+\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1}+\frac{r_{r2}}{\left(r_{s2}+r_{r2}\right)}\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\ddot{\theta}_{r3}+\frac{r_{r2}}{\left(r_{s2}+r_{r2}\right)}$$

$$\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2}+\frac{r_{r2}}{\left(r_{s2}+r_{r2}\right)}\frac{r_{r3}}{\left(r_{s3}+r_{r3}\right)}\left(\left(3m_{p3}r_{c3}^2+J_{c3}\right)\ddot{\theta}_{c3}+\frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3}+T_{out}\right)-T_{out}$$

$$J_s\ddot{\theta}_s+\left(1-p_1\right)J_{r1}\ddot{\theta}_{r1}+\left(1-\frac{1}{p_2}\right)J_{eq2}\ddot{\theta}_{r3}+\left(1-\frac{p_3}{p_2}\right)J_{eq3}\ddot{\theta}_{c3}-m_1 3J_{p1}\ddot{\theta}_{p1}-$$

$$\frac{1}{p_2}m_2 3J_{p2}\ddot{\theta}_{p2}-\frac{p_3}{p_2}3m_3J_{p3}\ddot{\theta}_{p3}=T_{in}-\left(1-\frac{p_3}{p_2}\right)T_{out}$$

$$\dot{\theta}_{p1}=-\frac{r_{s1}}{r_{p1}}\dot{\theta}_s\equiv-m_1\dot{\theta}_s$$

$$-\dot{\theta}_s r_{s1}-\dot{\theta}_{r1}r_{r1}=0\Rightarrow\dot{\theta}_{r1}=-\frac{r_{s1}}{r_{r1}}\dot{\theta}_s\equiv\beta_2\dot{\theta}_s$$

$$\dot{\theta}_s r_{s2}-\dot{\theta}_{r3}r_{s2}+\dot{\theta}_{p2}r_{p2}=0=>\dot{\theta}_{p2}=m_2\left(\alpha_2-1\right)\dot{\theta}_s$$

$$-\dot{\theta}_s r_{s2}+\dot{\theta}_{r3}\left(r_{r2}+r_{s2}\right)=0\Rightarrow\dot{\theta}_{r3}=\frac{r_{s2}}{\left(r_{r2}+r_{s2}\right)}\dot{\theta}_s\equiv\alpha_2\dot{\theta}_s$$

$$-\dot{\theta}_s r_{s2}+\dot{\theta}_{r3}\left(r_{r2}+r_{s2}\right)=0\Rightarrow\dot{\theta}_{r3}=\frac{r_{s2}}{\left(r_{r2}+r_{s2}\right)}\dot{\theta}_s\equiv\alpha_2\dot{\theta}_s$$

$$\dot{\theta}_s r_{s3}-\dot{\theta}_{c3}r_{s3}+\dot{\theta}_{p3}r_{p3}=0\Rightarrow\dot{\theta}_{p3}=m_3\left(\gamma_2-1\right)\dot{\theta}_s$$

$$-\dot{\theta}_s r_{s3}+\dot{\theta}_{c3}\left(r_{r3}+r_{s3}\right)-\dot{\theta}_{r3}r_{r3}=0\Rightarrow\dot{\theta}_{c3}=\left(\frac{r_{s2}}{\left(r_{r2}+r_{s2}\right)}\frac{r_{r3}}{\left(r_{r3}+r_{s3}\right)}+\frac{r_{s3}}{\left(r_{r3}+r_{s3}\right)}\right)\dot{\theta}_s\equiv\gamma_2\dot{\theta}_s$$

$$\left(J_s+\left(1-p_1\right)J_{r1}\beta_2+\left(1-\frac{1}{p_2}\right)J_{eq2}\alpha_2+\left(1-\frac{p_3}{p_2}\right)J_{eq3}\gamma_2+{m_1}^2 3J_{p1}-\frac{1}{p_2}m_2 3J_{p2}m_2\left(\alpha_2-1\right)-\right.$$

$$\left.\frac{p_3}{p_2}3m_3J_{p3}m_3\left(\gamma_2-1\right)\right)\ddot{\theta}_s=T_{in}-\left(1-\frac{p_3}{p_2}\right)T_{out}$$

1st gear derivation (brake #3, highest ratio)

$$J_s\ddot{\theta}_s = T_{in} + r_{s1}\lambda_1 - r_{s1}\lambda_2 + r_{s2}\lambda_3 - r_{s2}\lambda_4 + r_{s3}\lambda_5 - r_{s3}\lambda_6$$
$$3J_{p1}\ddot{\theta}_{p1} = r_{p1}\lambda_1$$
$$\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} = -T_{b2} - \left(r_{c1} - r_{p1}\right)\lambda_1 + \left(r_{s1} + r_{r1}\right)\lambda_2 - r_{r2}\lambda_4$$
$$J_{r1}\ddot{\theta}_{r1} = -T_{b1} - r_{r1}\lambda_2$$
$$3J_{p2}\ddot{\theta}_{p2} = r_{p2}\lambda_3$$
$$\left(3m_{p2}r_{c2}^2 + J_{c2} + J_{r3}\right)\ddot{\theta}_{r3} = -T_{b3} - \left(r_{c2} - r_{p2}\right)\lambda_3 + \left(r_{s2} + r_{r2}\right)\lambda_4 - r_{r3}\lambda_6$$
$$3J_{p3}\ddot{\theta}_{p3} = r_{p3}\lambda_5$$
$$\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} = -T_{out} - \left(r_{c3} - r_{p3}\right)\lambda_5 + \left(r_{s3} + r_{r3}\right)\lambda_6$$
$$\Rightarrow$$
$$J_s\ddot{\theta}_s = T_{in} + r_{s1}\lambda_1 - r_{s1}\lambda_2 + r_{s2}\lambda_3 - r_{s2}\lambda_4 + r_{s3}\lambda_5 - r_{s3}\lambda_6$$
$$3J_{p1}\ddot{\theta}_{p1} = r_{p1}\lambda_1$$
$$\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} = -\left(r_{c1} - r_{p1}\right)\lambda_1 + \left(r_{s1} + r_{r1}\right)\lambda_2 - r_{r2}\lambda_4$$
$$J_{r1}\ddot{\theta}_{r1} = -r_{r1}\lambda_2$$
$$3J_{p2}\ddot{\theta}_{p2} = r_{p2}\lambda_3$$
$$0 = -T_{b3} - \left(r_{c2} - r_{p2}\right)\lambda_3 + \left(r_{s2} + r_{r2}\right)\lambda_4 - r_{r3}\lambda_6$$
$$3J_{p3}\ddot{\theta}_{p3} = r_{p3}\lambda_5$$
$$\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} = -T_{out} - \left(r_{c3} - r_{p3}\right)\lambda_5 + \left(r_{s3} + r_{r3}\right)\lambda_6$$

$$J_s\ddot{\theta}_s = T_{in} + \frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1} + \frac{r_{s1}}{r_{r1}}J_{r1}\ddot{\theta}_{r1} + \frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2} + \frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3} - r_{s2}\lambda_4 - r_{s3}\lambda_6$$

$$\frac{1}{r_{p1}}3J_{p1}\ddot{\theta}_{p1} = \lambda_1$$

$$\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} = -\frac{\left(r_{c1} - r_{p1}\right)}{r_{p1}}3J_{p1}\ddot{\theta}_{p1} - \frac{\left(r_{s1} + r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1} - r_{r2}\lambda_4$$

$$-\frac{1}{r_{r1}}J_{r1}\ddot{\theta}_{r1} = \lambda_2$$

$$\frac{1}{r_{p2}}3J_{p2}\ddot{\theta}_{p2} = \lambda_3$$

$$0 = -T_{b3} - \frac{\left(r_{c2} - r_{p2}\right)}{r_{p2}}3J_{p2}\ddot{\theta}_{p2} + \left(r_{s2} + r_{r2}\right)\lambda_4 - r_{r3}\lambda_6$$

$$\frac{1}{r_{p3}}3J_{p3}\ddot{\theta}_{p3} = \lambda_5$$

$$\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} = -T_{out} - \frac{\left(r_{c3} - r_{p3}\right)}{r_{p3}}3J_{p3}\ddot{\theta}_{p3} + \left(r_{s3} + r_{r3}\right)\lambda_6$$

$$J_s\ddot{\theta}_s - \frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1} - \frac{r_{s1}}{r_{r1}}J_{r1}\ddot{\theta}_{r1} - \frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2} - \frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3} = T_{in} - r_{s2}\lambda_4 - r_{s3}\lambda_6 (1)$$

$$\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} + \frac{\left(r_{c1} - r_{p1}\right)}{r_{p1}}3J_{p1}\ddot{\theta}_{p1} + \frac{\left(r_{s1} + r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1} = -r_{r2}\lambda_4 (2)$$

$$\frac{\left(r_{c2} - r_{p2}\right)}{r_{p2}}3J_{p2}\ddot{\theta}_{p2} = -T_{b3} + \left(r_{s2} + r_{r2}\right)\lambda_4 - r_{r3}\lambda_6 (3)$$

$$\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} + \frac{\left(r_{c3} - r_{p3}\right)}{r_{p3}}3J_{p3}\ddot{\theta}_{p3} = -T_{out} + \left(r_{s3} + r_{r3}\right)\lambda_6 (4)$$

add all four equations

$$J_s\ddot{\theta}_s - \frac{r_{s1}}{r_{r1}}J_{r1}\ddot{\theta}_{r1} + \left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} + \frac{\left(r_{s1} + r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1} +$$

$$\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} = T_{in} - T_{b3} - T_{out} (5)$$

$$J_s\ddot{\theta}_s - \frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1} - \frac{r_{s1}}{r_{r1}}J_{r1}\ddot{\theta}_{r1} - \frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2} - \frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3} = T_{in} - r_{s2}\lambda_4 - r_{s3}\lambda_6 (1)$$

$$\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} + \frac{\left(r_{c1} - r_{p1}\right)}{r_{p1}}3J_{p1}\ddot{\theta}_{p1} + \frac{\left(r_{s1} + r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1} = -r_{r2}\lambda_4 (2)$$

$$\frac{\left(r_{c2} - r_{p2}\right)}{r_{p2}}3J_{p2}\ddot{\theta}_{p2} = -T_{b3} + \left(r_{s2} + r_{r2}\right)\lambda_4 - r_{r3}\lambda_6 (3)$$

$$\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} + \frac{\left(r_{c3} - r_{p3}\right)}{r_{p3}}3J_{p3}\ddot{\theta}_{p3} = -T_{out} + \left(r_{s3} + r_{r3}\right)\lambda_6 (4)$$

from (2)

$$\lambda_4 = -\frac{1}{r_{r2}}\left(\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} + \frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1} + \frac{\left(r_{s1} + r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1}\right)$$

from (4)

$$\lambda_6 = \frac{1}{\left(r_{s3} + r_{r3}\right)}\left(\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} + \frac{\left(r_{c3} - r_{p3}\right)}{r_{p3}}3J_{p3}\ddot{\theta}_{p3} + T_{out}\right)$$

into (3)

$$T_{b3} = -\frac{r_{s2}}{r_{p2}}3J_{p2}\ddot{\theta}_{p2} - \frac{\left(r_{s2} + r_{r2}\right)}{r_{r2}}\left(\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} + \frac{r_{s1}}{r_{p1}}3J_{p1}\ddot{\theta}_{p1} + \frac{\left(r_{s1} + r_{r1}\right)}{r_{r1}}J_{r1}\ddot{\theta}_{r1}\right) -$$

$$\frac{r_{r3}}{\left(r_{s3} + r_{r3}\right)}\left(\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} + \frac{r_{s3}}{r_{p3}}3J_{p3}\ddot{\theta}_{p3} + T_{out}\right) (*)$$

$$T_{b3} = -\frac{r_{s2}}{r_{p2}} 3J_{p2}\ddot{\theta}_{p2} - \frac{(r_{s2}+r_{r2})}{r_{r2}}\left(\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} + \frac{r_{s1}}{r_{p1}} 3J_{p1}\ddot{\theta}_{p1} + \frac{(r_{s1}+r_{r1})}{r_{r1}} J_{r1}\ddot{\theta}_{r1} \right) -$$

$$\frac{r_{r3}}{(r_{s3}+r_{r3})}\left(\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} + \frac{r_{s3}}{r_{p3}} 3J_{p3}\ddot{\theta}_{p3} + T_{out} \right)$$

$$J_s\ddot{\theta}_s - \frac{r_{s1}}{r_{r1}} J_{r1}\ddot{\theta}_{r1} + \left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} + \frac{(r_{s1}+r_{r1})}{r_{r1}} J_{r1}\ddot{\theta}_{r1} +$$

$$\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} = T_{in} + \frac{r_{s2}}{r_{p2}} 3J_{p2}\ddot{\theta}_{p2} +$$

$$\frac{(r_{s2}+r_{r2})}{r_{r2}}\left(\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} + \frac{r_{s1}}{r_{p1}} 3J_{p1}\ddot{\theta}_{p1} + \frac{(r_{s1}+r_{r1})}{r_{r1}} J_{r1}\ddot{\theta}_{r1} \right) +$$

$$\frac{r_{r3}}{(r_{s3}+r_{r3})}\left(\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} + \frac{r_{s3}}{r_{p3}} 3J_{p3}\ddot{\theta}_{p3} + T_{out} \right) - T_{out}$$

$\Rightarrow$

$$J_s\ddot{\theta}_s + J_{r1}\ddot{\theta}_{r1} + \left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} + \left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} - \frac{r_{s2}}{r_{p2}} 3J_{p2}\ddot{\theta}_{p2} -$$

$$\frac{(r_{s2}+r_{r2})}{r_{r2}}\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} - \frac{(r_{s2}+r_{r2})}{r_{r2}}\frac{r_{s1}}{r_{p1}} 3J_{p1}\ddot{\theta}_{p1} - \frac{(r_{s2}+r_{r2})}{r_{r2}}\frac{(r_{s1}+r_{r1})}{r_{r1}} J_{r1}\ddot{\theta}_{r1} -$$

$$\frac{r_{r3}}{(r_{s3}+r_{r3})}\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} - \frac{r_{r3}}{(r_{s3}+r_{r3})}\frac{r_{s3}}{r_{p3}} 3J_{p3}\ddot{\theta}_{p3}$$

$$= T_{in} + \frac{r_{r3}}{(r_{s3}+r_{r3})} T_{out} - T_{out}$$

$\Rightarrow$

$$J_s\ddot{\theta}_s + \left(1 - \frac{(r_{s2}+r_{r2})}{r_{r2}}\frac{(r_{s1}+r_{r1})}{r_{r1}}\right) J_{r1}\ddot{\theta}_{r1} + \left(1 - \frac{(r_{s2}+r_{r2})}{r_{r2}}\right)\left(3m_{p1}r_{c1}^2 + J_{c1} + J_{r2}\right)\ddot{\theta}_{r2} +$$

$$\left(1 - \frac{r_{r3}}{(r_{s3}+r_{r3})}\right)\left(3m_{p3}r_{c3}^2 + J_{c3}\right)\ddot{\theta}_{c3} - \frac{(r_{s2}+r_{r2})}{r_{r2}}\frac{r_{s1}}{r_{p1}} 3J_{p1}\ddot{\theta}_{p1} - \frac{r_{s2}}{r_{p2}} 3J_{p2}\ddot{\theta}_{p2} - \frac{r_{r3}}{(r_{s3}+r_{r3})}\frac{r_{s3}}{r_{p3}} 3J_{p3}\ddot{\theta}_{p3}$$

$$= T_{in} - \frac{r_{s3}}{(r_{s3}+r_{r3})} T_{out}$$

$$J_s\ddot{\theta}_s+\left(1-p_1p_2\right)J_{r1}\ddot{\theta}_{r1}+\left(1-p_2\right)J_{eq1}\ddot{\theta}_{r2}+\left(1-p_3\right)J_{eq3}\ddot{\theta}_{c3}$$
$$-3p_2m_1J_{p1}\ddot{\theta}_{p1}-3m_2J_{p2}\ddot{\theta}_{p2}-3p_3m_3J_{p3}\ddot{\theta}_{p3}=T_{in}-\left(1-p_3\right)T_{out}$$

$$\frac{\left(r_{s1}+r_{r1}\right)}{r_{r1}}\equiv p_1,\frac{\left(r_{s2}+r_{r2}\right)}{r_{r2}}\equiv p_2,\frac{r_{r3}}{\left(r_{s3}+r_{r3}\right)}\equiv p_3$$
$$\frac{r_{s1}}{r_{p1}}\equiv m_1,\frac{r_{s2}}{r_{p2}}\equiv m_2,\frac{r_{s3}}{r_{p3}}\equiv m_3$$
$$\left(3m_{p1}r_{c1}^2+J_{c1}+J_{r2}\right)\equiv J_{eq1},\left(3m_{p2}r_{c2}^2+J_{c2}+J_{r3}\right)\equiv J_{eq2},\left(3m_{p3}r_{c3}^2+J_{c3}\right)\equiv J_{eq3}$$

$$\left(J_s+\left(1-p_1p_2\right)\left(\alpha_3p_1+\beta_2\right)J_{r1}+\left(1-p_2\right)\alpha_3J_{eq1}+\right.$$
$$\left.\left(1-p_3\right)\gamma_3J_{eq3}-3p_1m_1^{\,2}\left(\alpha_3-1\right)J_{p1}+3m_2^{\,2}J_{p2}-3p_3m_3^{\,2}\left(\gamma_3-1\right)J_{p3}\right)\ddot{\theta}_s$$
$$=T_{in}-\gamma_3T_{out}$$

$$\ddot{\theta}_sr_{s1}-\ddot{\theta}_{r2}r_{s1}+\ddot{\theta}_{p1}r_{p1}=0\Rightarrow\ddot{\theta}_{p1}=\frac{r_{s1}}{r_{p1}}\left(\alpha_3-1\right)\ddot{\theta}_s=\left(\alpha_3-1\right)\frac{r_{s1}}{r_{p1}}\ddot{\theta}_s$$
$$-\ddot{\theta}_sr_{s1}+\ddot{\theta}_{r2}\left(r_{r1}+r_{s1}\right)-\ddot{\theta}_{r1}r_{r1}=0\Rightarrow\ddot{\theta}_{r1}=\frac{-r_{s1}+\alpha_3\left(r_{r1}+r_{s1}\right)}{r_{r1}}\ddot{\theta}_s$$
$$\ddot{\theta}_sr_{s2}+\ddot{\theta}_{p2}r_{p2}=0\Rightarrow\ddot{\theta}_{p2}=-\frac{r_{s2}}{r_{p2}}\ddot{\theta}_s\equiv-m_2\ddot{\theta}_s$$
$$-\ddot{\theta}_sr_{s2}-\ddot{\theta}_{r2}r_{r2}=0\Rightarrow\ddot{\theta}_{r2}=-\frac{r_{s2}}{r_{r2}}\ddot{\theta}_s\equiv\alpha_3\ddot{\theta}_s$$
$$\ddot{\theta}_sr_{s3}-\ddot{\theta}_{c3}r_{s3}+\ddot{\theta}_{p3}r_{p3}=0\Rightarrow\ddot{\theta}_{p3}=\frac{r_{s3}}{r_{p3}}\left(\gamma_3-1\right)\ddot{\theta}_s$$
$$-\ddot{\theta}_sr_{s3}+\ddot{\theta}_{c3}\left(r_{r3}+r_{s3}\right)=0\Rightarrow\ddot{\theta}_{c3}=\frac{r_{s3}}{\left(r_{r3}+r_{s3}\right)}\ddot{\theta}_s\equiv\gamma_3\ddot{\theta}_s$$

A planetary gear with ring gear locked:

$$J_s\ddot{\theta}_s + c_s\dot{\theta}_s = T_{in} + (\lambda_1 - \lambda_2) r_s \quad (1)$$

$$\left(3m_p r_c^2 + J_c\right)\ddot{\theta}_c + c_c\dot{\theta}_c = -T_{out} - \lambda_1 r_s + \lambda_2 (r_s + r_r) \quad (2)$$

$$3J_p\ddot{\theta}_p + c_p\dot{\theta}_p = \lambda_1 r_p \quad (3)$$

$$J_r\ddot{\theta}_r + c_r\dot{\theta}_r = -T_b - \lambda_2 r_r \quad (4)$$

when the ring gear is blocked, $\ddot{\theta}_r = 0, \dot{\theta}_r = 0$

(1) + (2) + (4)

$$J_s\ddot{\theta}_s + c_s\dot{\theta}_s + \left(3m_p r_c^2 + J_c\right)\ddot{\theta}_c + c_c\dot{\theta}_c = T_{in} + (\lambda_1 - \lambda_2) r_s - T_{out} - \lambda_1 r_s + \lambda_2 (r_s + r_r) - T_b - \lambda_2 r_r$$

$\Rightarrow$

$$J_s\ddot{\theta}_s + \left(3m_p r_c^2 + J_c\right)\ddot{\theta}_c + c_s\dot{\theta}_s + c_c\dot{\theta}_c = T_{in} - T_{out} - T_b$$

$$\Rightarrow\ T_b = T_{in} - T_{out} - J_s\ddot{\theta}_s - \left(3m_p r_c^2 + J_c\right)\ddot{\theta}_c - c_s\dot{\theta}_s - c_c\dot{\theta}_c \quad (*)$$

$$from(3)\quad \lambda_1 = \frac{3J_p\ddot{\theta}_p + c_p\dot{\theta}_p}{r_p}, \qquad from(4)\ \ \lambda_2 = -\frac{T_b}{r_r}$$

substitute λ_1, λ_2 into (1)

$$J_s\ddot{\theta}_s + c_s\dot{\theta}_s = T_{in} + \frac{r_s}{r_p}\left(3J_p\ddot{\theta}_p + c_p\dot{\theta}_p\right) + \frac{r_s}{r_r}T_b \quad (5)$$

(*) into (5)

$$J_s\ddot{\theta}_s + c_s\dot{\theta}_s = T_{in} + \frac{r_s}{r_p}\left(3J_p\ddot{\theta}_p + c_p\dot{\theta}_p\right) - \frac{r_s}{r_r}\left(T_{in} - T_{out} - J_s\ddot{\theta}_s - \left(3m_p r_c^2 + J_c\right)\ddot{\theta}_c - c_s\dot{\theta}_s - c_c\dot{\theta}_c\right)$$

$\Rightarrow$

$$J_s\ddot{\theta}_s + c_s\dot{\theta}_s = T_{in} + \frac{r_s}{r_p}\left(3J_p\ddot{\theta}_p + c_p\dot{\theta}_p\right) + \frac{r_s}{r_r}\left(T_{in} - T_{out} - J_s\ddot{\theta}_s - \left(3m_p r_c^2 + J_c\right)\ddot{\theta}_c - c_s\dot{\theta}_s - c_c\dot{\theta}_c\right)$$

$\Rightarrow$

$$\frac{(r_s + r_r)}{r_s}J_s\ddot{\theta}_s + \frac{(r_s + r_r)}{r_s}c_s\dot{\theta}_s + \left(3m_p r_c^2 + J_c\right)\ddot{\theta}_c + c_c\dot{\theta}_c - \frac{r_r}{r_p}\left(3J_p\ddot{\theta}_p + c_p\dot{\theta}_p\right) = T_{in}\frac{r_s + r_r}{r_s} - T_{out} \quad (6)$$

$$J_s\ddot{\theta}_s + c_s\dot{\theta}_s + \frac{r_s}{r_r + r_s}\left(3m_p r_c^2 + J_c\right)\ddot{\theta}_c + \frac{r_s}{r_r + r_s}c_c\dot{\theta}_c - \frac{r_s}{r_r + r_s}\frac{r_r}{r_p}\left(3J_p\ddot{\theta}_p + c_p\dot{\theta}_p\right) = T_{in} - \frac{r_s}{r_r + r_s}T_{out} \quad (6)$$

$$\dot{\theta}_s r_s - \dot{\theta}_c r_s + \dot{\theta}_p r_p = 0 \Rightarrow \dot{\theta}_p = \frac{r_s}{r_p}\left(\gamma - 1\right)\dot{\theta}_s \equiv m\left(\gamma - 1\right)\dot{\theta}_s$$

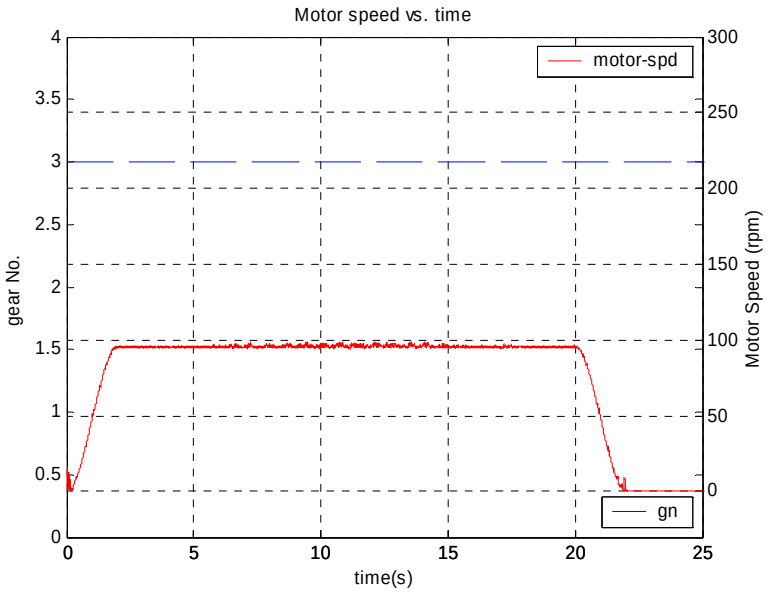
$$\dot{\theta}_r r_r - \dot{\theta}_c\left(r_r + r_s\right) + \dot{\theta}_s r_s = 0 \Rightarrow \dot{\theta}_c = \frac{r_s}{r_r + r_s}\dot{\theta}_s \equiv \gamma\dot{\theta}_s$$

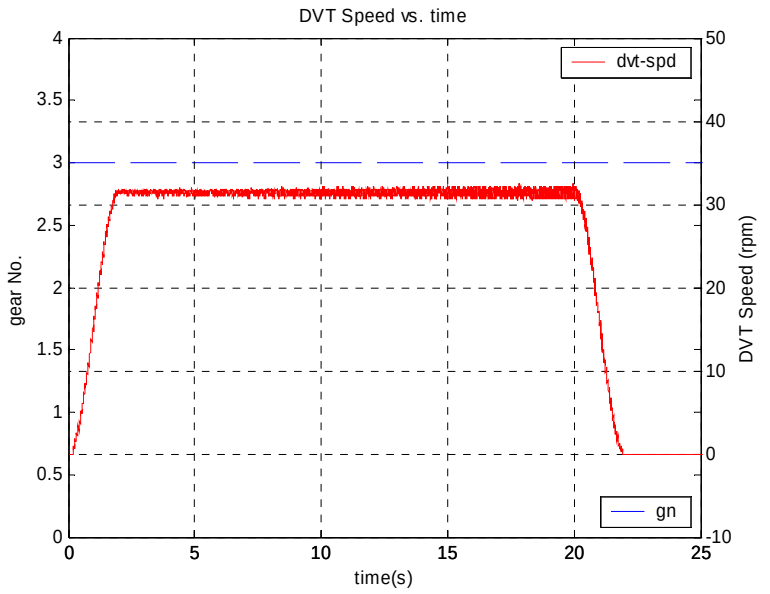
$$\left(3m_p r_c^2 + J_c\right) \equiv J_{eq}$$

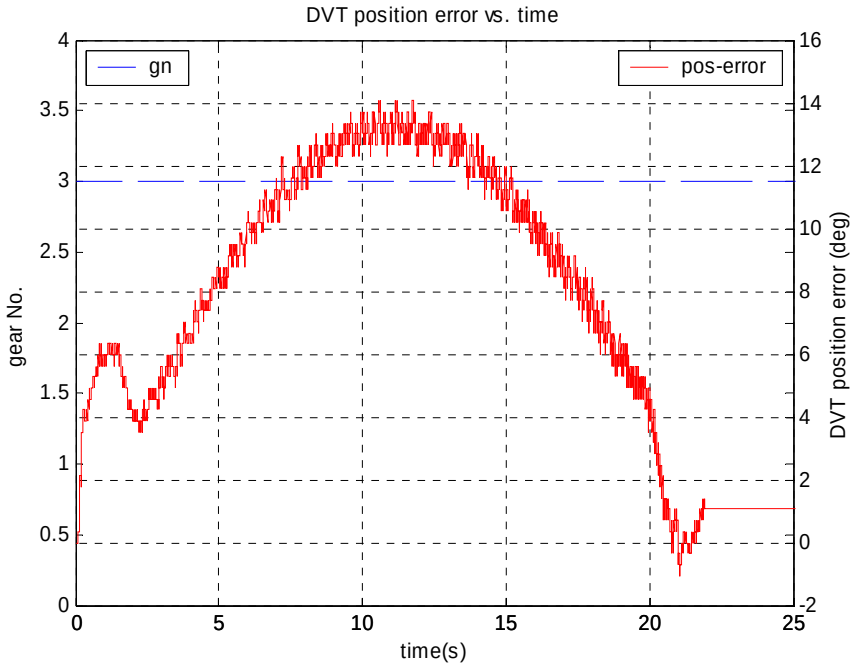
$$\left(J_s - 3m\left(m + 2\right)\gamma\left(\gamma - 1\right)J_p + \gamma J_{eq}\right)\ddot{\theta}_s + \left(c_s + c_c\gamma - c_p m\left(m + 2\right)\gamma\left(\gamma - 1\right)\right)\dot{\theta}_s = T_{in} - \gamma T_{out}$$

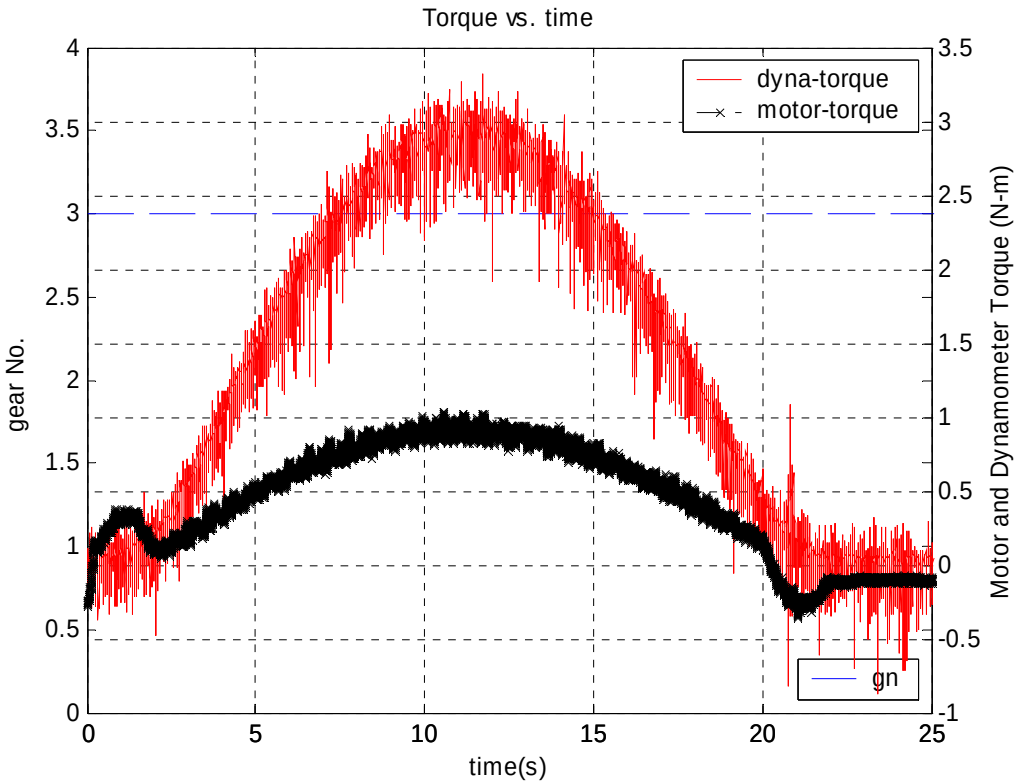
Experimental results

Zero load

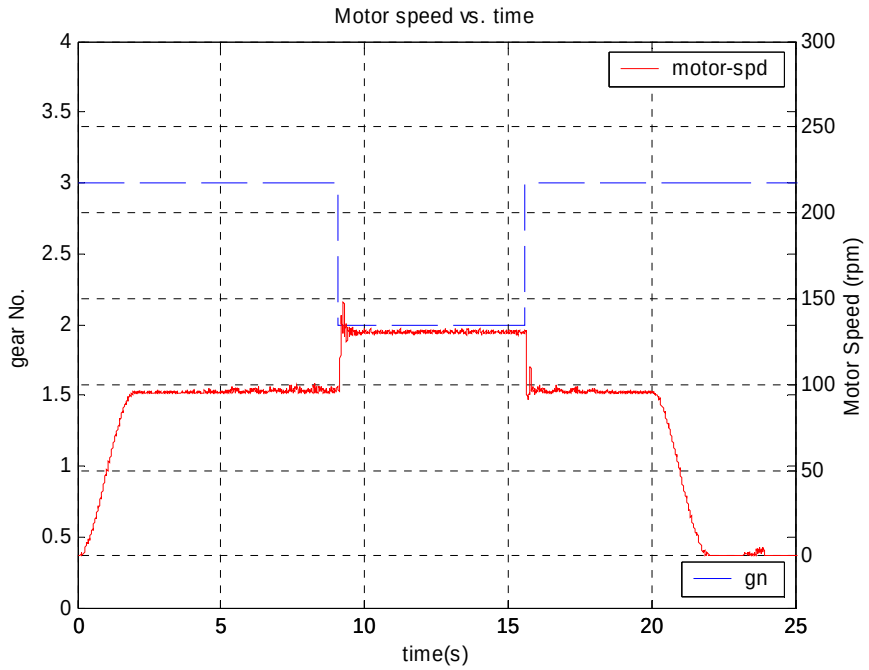


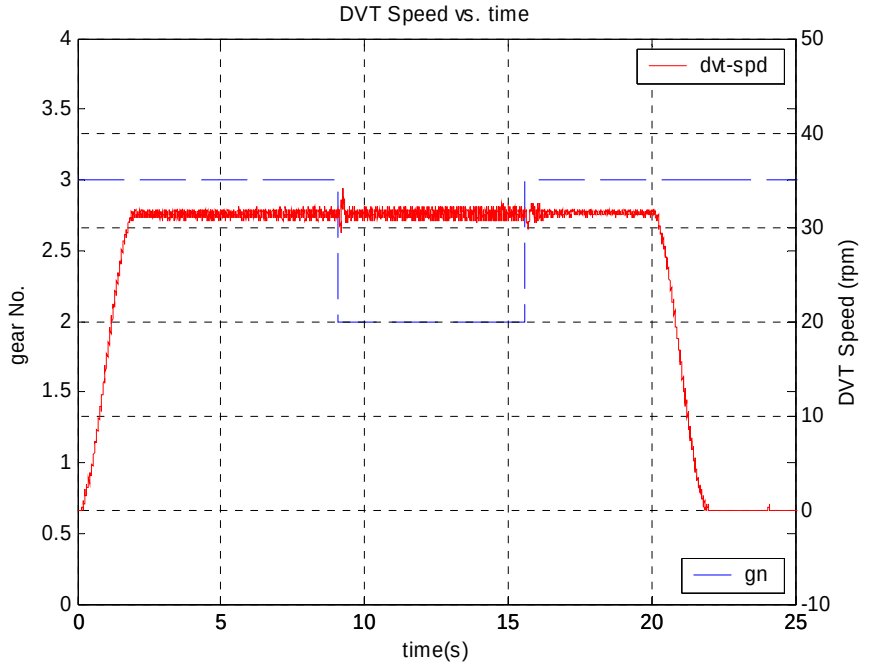


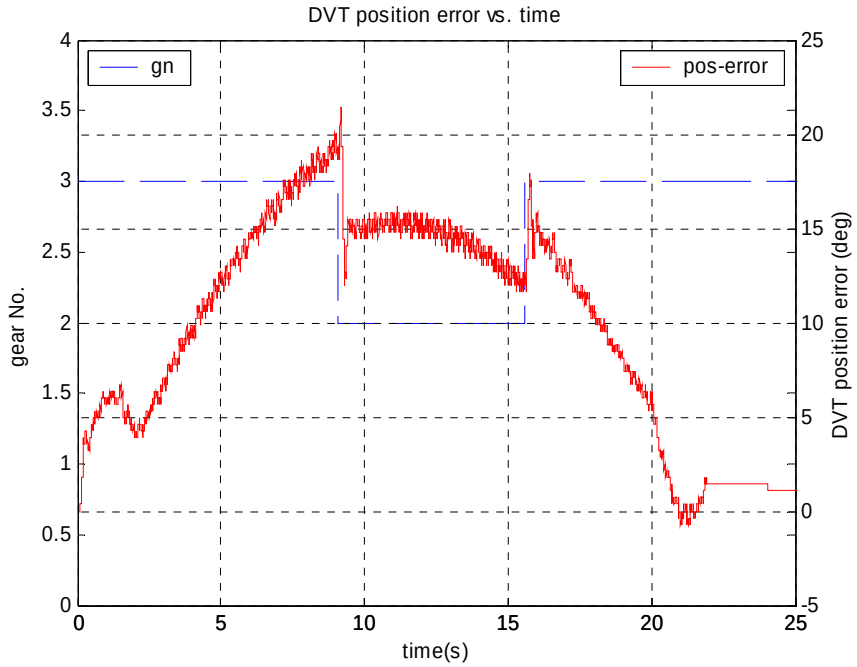

DVT position error vs. time
gn
pos-error
gear No.
DVT position error (deg)
time(s)
0
0.5
1
1.5
2
2.5
3
3.5
4
-2
0
2
4
6
8
10
12
14
16
0
5
10
15
20
25

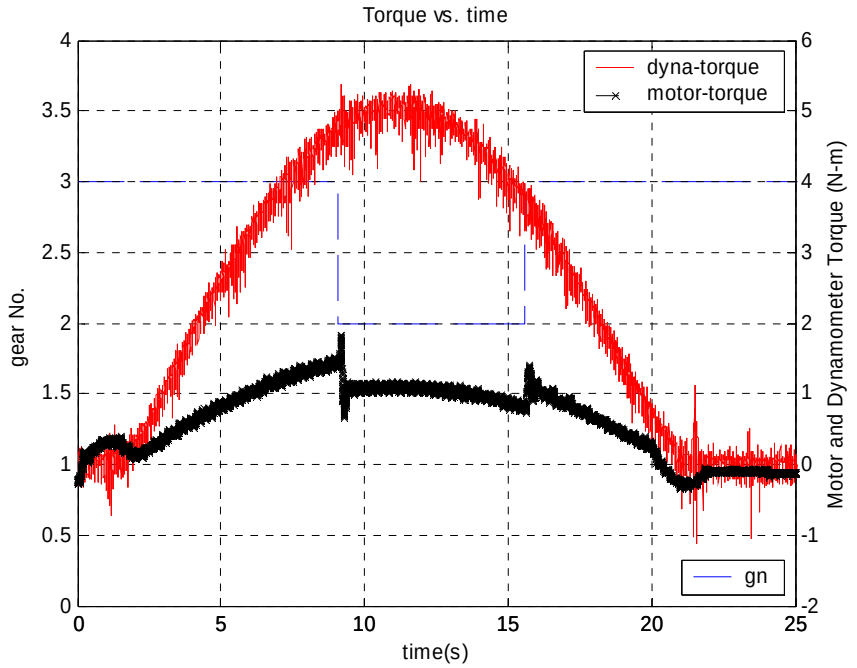

Torque vs. time
dyna-torque
motor-torque
gn
gear No.
Motor and Dynamometer Torque (N-m)
time(s)
0
0.5
1
1.5
2
2.5
3
3.5
4
-1
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
0
5
10
15
20
25

25% load

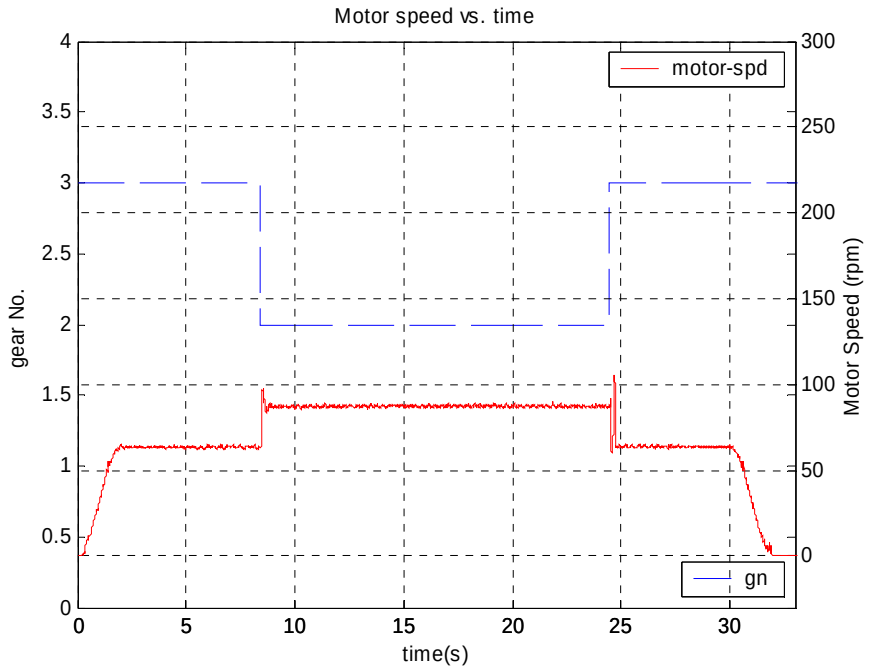

Motor speed vs. time
motor-spd
gn
gear No.
Motor Speed (rpm)
time(s)
0
0.5
1
1.5
2
2.5
3
3.5
4
0
50
100
150
200
250
300
0
5
10
15
20
25

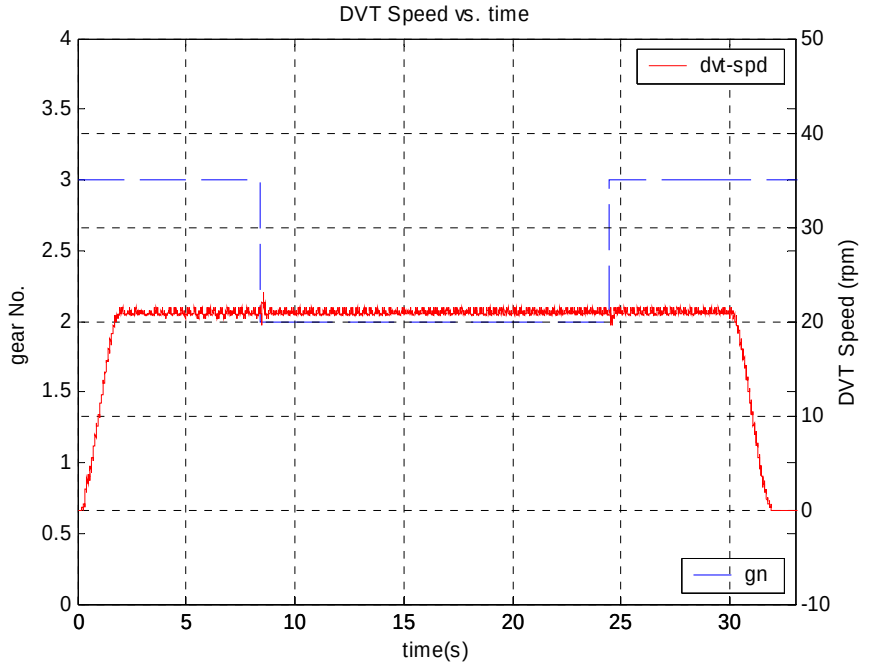

DVT Speed vs. time
dvt-spd
gn
gear No.
DVT Speed (rpm)
time(s)
0
0.5
1
1.5
2
2.5
3
3.5
4
-10
0
10
20
30
40
50
0
5
10
15
20
25

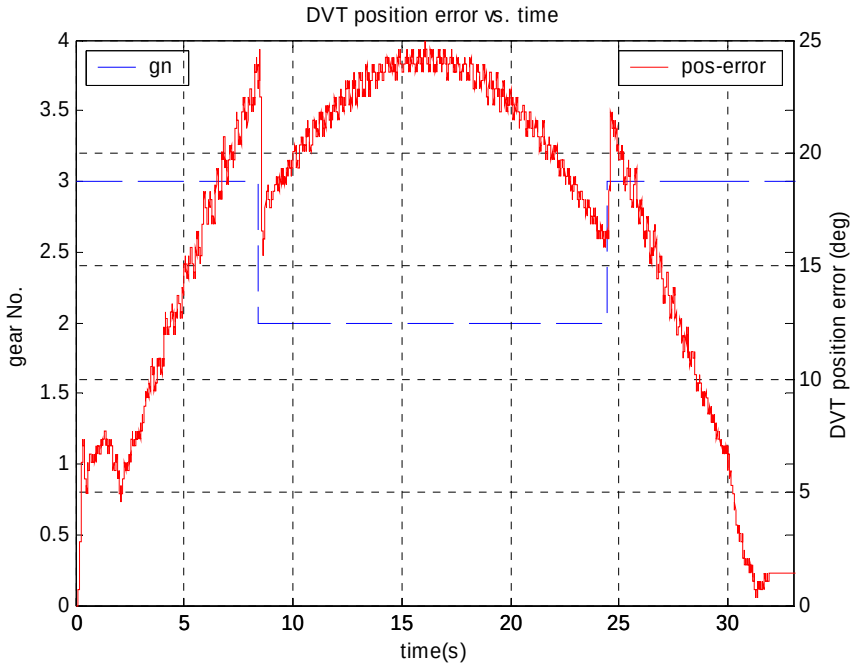

DVT position error vs. time
gn
pos-error
gear No.
DVT position error (deg)
time(s)
0
0.5
1
1.5
2
2.5
3
3.5
4
-5
0
5
10
15
20
25

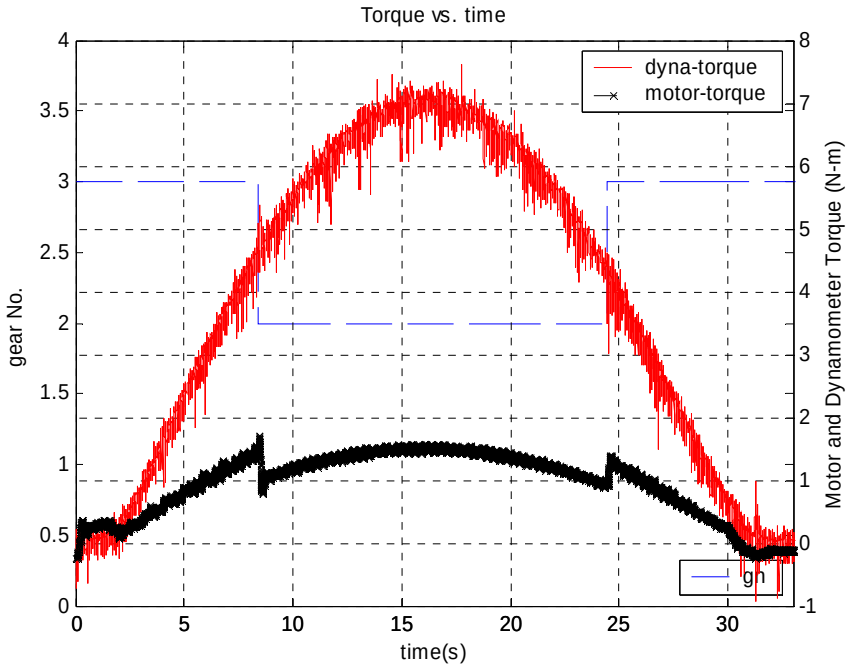

Torque vs. time
dyna-torque
motor-torque
gn
gear No.
Motor and Dynamometer Torque (N-m)
time(s)
0
0.5
1
1.5
2
2.5
3
3.5
4
-2
-1
0
1
2
3
4
5
6
0
5
10
15
20
25

50% load

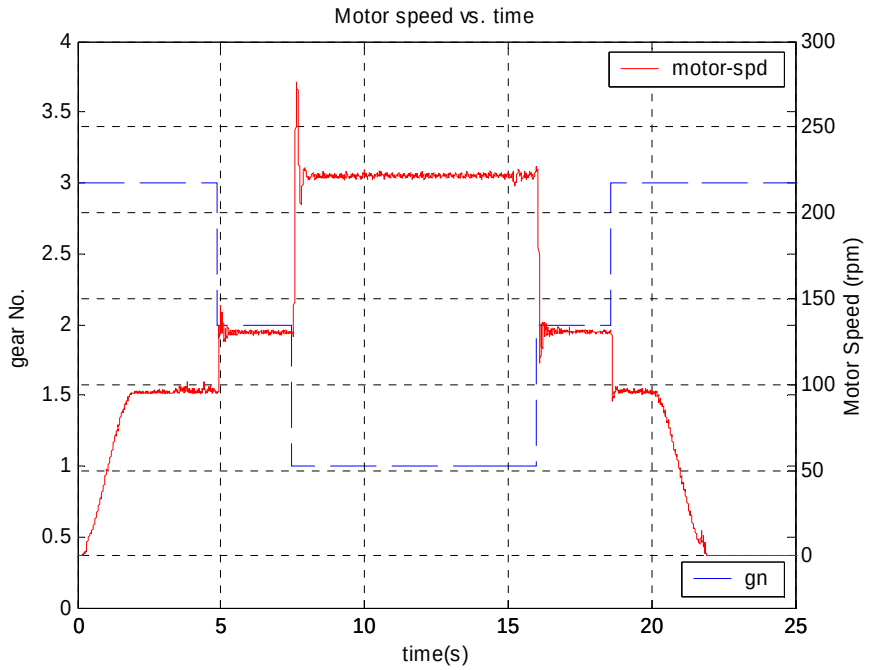

Motor speed vs. time
motor-spd
gn
gear No.
Motor Speed (rpm)
time(s)
0
0.5
1
1.5
2
2.5
3
3.5
4
0
50
100
150
200
250
300
0
5
10
15
20
25
30

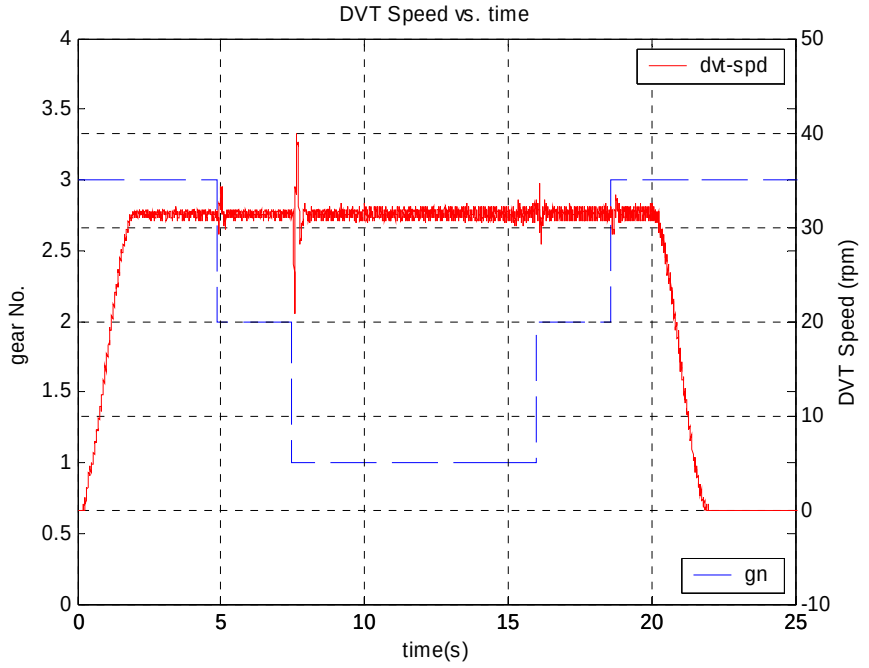

DVT Speed vs. time
dvt-spd
gn
gear No.
DVT Speed (rpm)
time(s)
0
0.5
1
1.5
2
2.5
3
3.5
4
-10
0
10
20
30
40
50
0
5
10
15
20
25
30

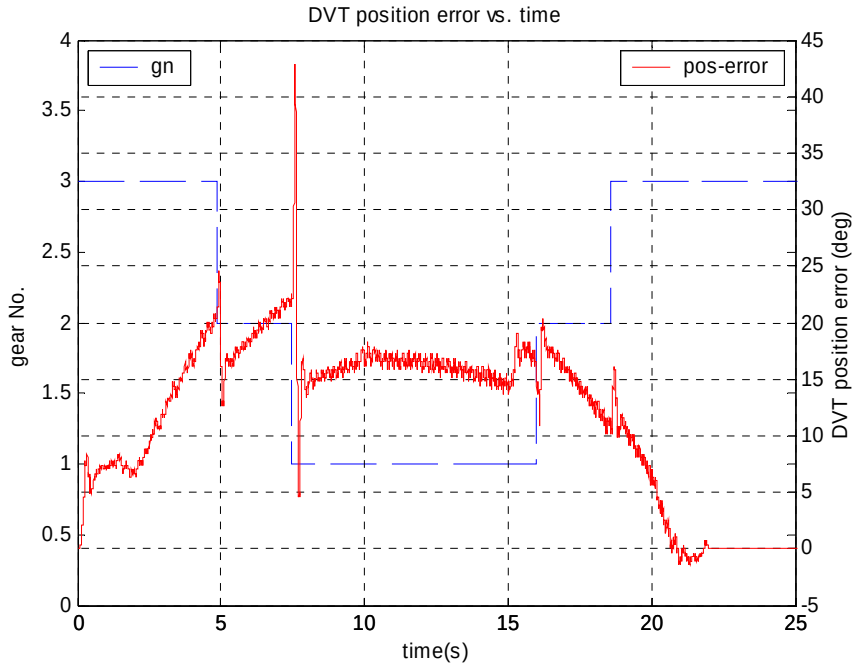

DVT position error vs. time
gn
pos-error
gear No.
DVT position error (deg)
time(s)
0
0.5
1
1.5
2
2.5
3
3.5
4
0
5
10
15
20
25
30

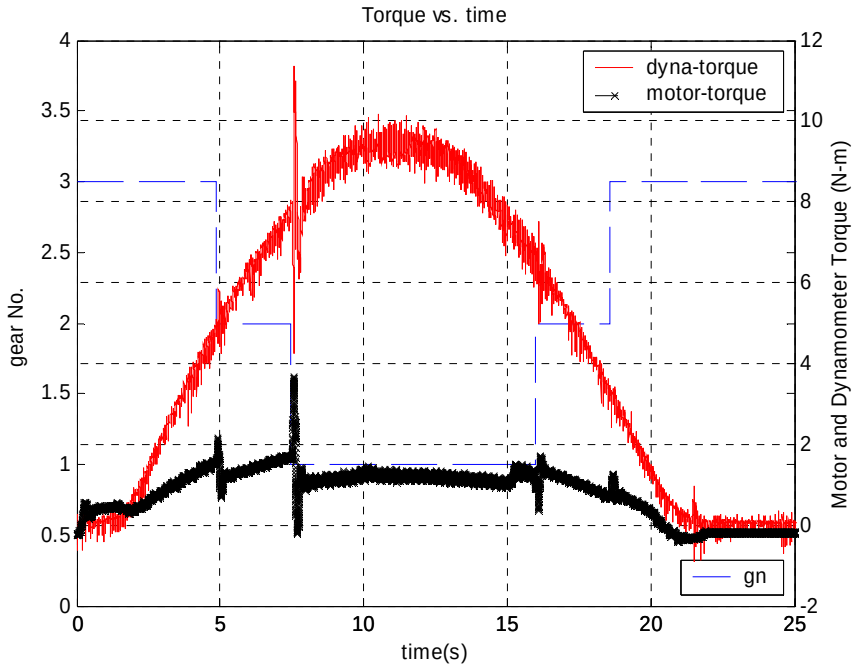

Torque vs. time
dyna-torque
motor-torque
gn
gear No.
Motor and Dynamometer Torque (N-m)
time(s)
0
0.5
1
1.5
2
2.5
3
3.5
4
-1
0
1
2
3
4
5
6
7
8
0
5
10
15
20
25
30

75% load


Motor speed vs. time
motor-spd
gn
gear No.
Motor Speed (rpm)
time(s)
0
0.5
1
1.5
2
2.5
3
3.5
4
0
50
100
150
200
250
300
0
5
10
15
20
25


DVT Speed vs. time
dvt-spd
gn
gear No.
DVT Speed (rpm)
time(s)
0
0.5
1
1.5
2
2.5
3
3.5
4
-10
0
10
20
30
40
50
0
5
10
15
20
25


DVT position error vs. time
gn
pos-error
gear No.
DVT position error (deg)
time(s)


Torque vs. time
dyna-torque
motor-torque
gn
gear No.
Motor and Dynamometer Torque (N-m)
time(s)

TBA low level control software source code

tbacontrol.c

```
/* Transmission based servo actuator system control
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file tbacontrol.c
        This is the main function of the TBA control software
        It does the following tasks by calling specific function calls:

-# Initialize servo
-# Clibrate NI-6023e
-# Initialize the braking motors
-# Startup TBA GUI interface
-# Manage all the task threads

*/

#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <getopt.h>
#include <ctype.h>
#include <signal.h>
#include <comedilib.h>
#include <pthread.h>

#define KEEP_STATIC_INLINE
#include <rtai_lxrt_user.h>
#include <rtai_lxrt.h>
#include <rtai_fifos.h>
#include <rtai_msg.h>

#include "ni6711.h"
#include "ni6023e.h"
#include "common.h"
#include "setmode.h"
#include "control.h"

int quit=0;

int main()
{
```

```
int drive_mode,controller_id,to_gear;

pid_t pid;
int status;


drive_mode=2;
controller_id=2;
to_gear=1;


if(init_hrt()==ERROR)
{
printf("hard real timer initilization failed!\n");
exit(0);
}


//initialize NI-6711

if(!(ni6711.ni_daq=init_6711()))
        {
                printf("NI6711 initializtion failed!\n");
                exit(ERROR);
                }


//initilize NI-6023e


if(!(ni6023e.ni_daq=init_6023e()))
        {
                printf("NI6023e initializtion failed!\n");
                exit(ERROR);
                }
printf("\n\n************************************************************\n");
printf("\nwait for analog input calibaration finshed!\n");
printf("\n************************************************************\n");
sleep(5);


//initialize rtai fifo

if (!(rtfifo = rtf_open_sized("/dev/rtf0", O_RDWR, 2000))) {
                printf("ERROR OPENING FIFO0\n");
                exit(ERROR);
          }

//start gui

pid=fork();

if (pid==-1)
{
printf("fork failed!\n");
exit(0);
```

```
}

if(pid==0)
{
execv("/home/robin/Dissertation/src/qtgui/realtime","");
_exit (EXIT_FAILURE);
}

if (pid==1)
{
printf("I am parent, waiting\n");
waitpid(pid,&status,0);
}

printf("\n\n*************************************************************\n");
printf("\nWait for gui to start,click 'START' button on the gui!\n");
printf("\n*************************************************************\n");
sleep(3);


//set drive mode

setMode(drive_mode);
printf("\n\n*************************************************************\n");
printf("\nwait for u3k initialization ...\n");
printf("\n*************************************************************\n");


printf("Done!\n");



pthread_create(&shift_thrd,NULL,setGear,(void*)mbxgear);
pthread_create(&stdout_thrd,NULL,stdOut,(void*)mbxstdout);


motorControl(controller_id,drive_mode);
quit=1;


pthread_join(shift_thrd,NULL);
pthread_join(stdout_thrd,NULL);


//clean up before termination

close(rtfifo);

return;
}
```

common.h

```
/* Transmission based servo actuator system control
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file common.h


*/
#ifndef _COMMONMETRIC_
 #define _COMMONMETRIC_

 #include "ni6023e.h"
 #include "ni6711.h"
 #include "control.h"

#include <rtai_lxrt_user.h>
#include <rtai_lxrt.h>
#include <rtai_fifos.h>
#include <rtai_mbx.h>
#include <pthread.h>


 #define PI 3.1415926
 #define rev2cnt(x) ((x)*1024) //counter resolution 1024 counts per revolution
 #define cnt2rev(x) ((x)/1024)
 #define rev2deg(x) ((x)*360)
 #define deg2rev(x) ((x)/360)
 #define cnt2rad(x) ((x)/1024*2*PI)
 #define deg2rad(x) ((x)*PI/180)
 #define rad2cnt(x) ((x)/2/PI*1024)

 #define rpm2rps(x) ((x)*2*PI/60) //rev/min to rad/sec

 #define sec2nsec(x) ((x)*1000000000)
 #define sec2usec(x) ((x)*1000000)
 #define sec2msec(x) ((x)*1000)
 #define msec2nsec(x) ((x)*1000000)
 #define msec2usec(x) ((x)*1000)
 #define usec2nsec(x) ((x)*1000)
 #define ENGAGE 0
 #define DISENGAGE 1

 #define MAFPNT 20

 //control loop

 #define PERIOD      0.001 //control loop time (second)
```

```
//return status
#define ERROR 0
#define OK    1

//verbose output mode
#define VERBOSE


//ultra 3000-030x

#define SYSAMPS 30 //drive peak current
#define AOSCALE 0x7FFF //analog current output scale 0x7FFF
#define AOCONST (8191*128) //a constant as in AOSCALE/AOCONST*SYSAMPS (amps/volt)

//loadcell calibration
#define  N-M_PER_VOLT 3.53698

//motor parametrs
#define MOTOR_TORQUE_CONSTANT 0.414

//load and arm
#define ARMWEIGHT 90.71847 //Kg=200lb
#define ARMLENGTH 1.8288   //m=72inch



//rtai fifo handle
int rtfifo;

unsigned int ticks_per_second;

//int quit;

typedef struct
{
comedi_t *ni_daq; //handle for DAQ
int n_subdev; //number of subdevices
int n_ranges;//number of channel for specific subdevice
int n_channels;
comedi_range *rng;//range information of channel
lsampl_t maxdata;//maximum data value for specific channel
lsampl_t offset; //sample value corresponding to physical zero volt
}analogDev;

analogDev ni6711;
analogDev ni6023e;
/*
typedef struct
{
double time;
int dvt_position;
int motor_position;
int gear_current;
int command_position;
double motor_speed;
```

```
 double dvt_speed;
 double load;
 //double motor_torque;
  }stdoutMsg;
*/
 typedef struct
 {
 float time;
 int dvt_position;
 int motor_position;
 int gear_current;
 int command_position;
 float motor_speed;
 float dvt_speed;
 float load;
 float motor_torque;
  }stdoutMsg;

MBX *mbxstdout;
MBX *mbxgear;

pthread_t shift_thrd;
pthread_t stdout_thrd;

 void endme(int);
 void motorControl(int,int);
 void*setGear(void *);
 int init_hrt(void);
 double readLoadcell(void);
 void dynaControl(tbaPID *,float);
 void *stdOut(void *);
 //char* openDataFile(int, int);

#endif
```

common.c

```
/* Transmission based servo actuator system control
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file common.c
        This file defines functions to initialize a real time task and RTAI mailboxes:



*/

 #include "common.h"
 #include <stdlib.h>
 #include <rtai_shm.h>


 extern int quit;

/** \function
        initilize a real time task and two mailboxes


*/

 int init_hrt(void){

  struct sched_param mysched;
  RT_TASK *hrttsk;
  unsigned long hrttsk_name;

  rt_allow_nonroot_hrt();

  mysched.sched_priority = sched_get_priority_max(SCHED_FIFO)-4;

  if( sched_setscheduler( 0, SCHED_FIFO, &mysched ) == -1 ) {
    puts("ERROR IN SETTING THE SCHEDULER");
    perror("errno");
    return 0;
  }

  hrttsk_name = nam2num("HRTTSK");

  if(!(hrttsk = rt_task_init(hrttsk_name, 1, 0 ,0 ))) {
  puts("Can't Init Hard Real Time TASK\n");
  return 0;
  }
```

```
  rt_set_oneshot_mode();

  ticks_per_second = (unsigned int)nano2count((RTIME)(1000000000));

  printf("Ticks per second: %u\n",ticks_per_second);

  rt_task_use_fpu(hrttsk,1);

  rt_linux_use_fpu(1);


  // make a mailbox for shift controller  use
  if((mbxgear=rt_mbx_init(nam2num("SHFTCTR"),2048))==0)

  {
    puts("Couldn't create shiftcontrol mailbox.\n");
    exit(4);
  }

  // make a mailbox for shift controller  use
  if((mbxstdout=rt_mbx_init(nam2num("STDOUT"),2048))==0)

  {
    puts("Couldn't create stdout mailbox.\n");
    exit(4);
  }


  start_rt_timer(0);

  rt_task_delete(hrttsk);

  return 1;
 }


/** \function
          receive messages from a mailbox and send them out to a real time FIFO


*/

 void* stdOut(void*pmbx)
{
  RT_TASK *stdtsk;
  struct sched_param mysched;
  mysched.sched_priority = 98;
  int nbyte;


  stdoutMsg *msg_rcv;

  msg_rcv=(stdoutMsg *)malloc(sizeof(stdoutMsg));
```

```
  if( sched_setscheduler( 0, SCHED_FIFO, &mysched ) == -1 )

  {
    puts(" ERROR IN SETTING THE SCHEDULER UP");
    perror( "errno" );
    exit( 0 );

  }

  if(!(stdtsk = rt_task_init(nam2num("STDTSK"), 1 , 0 ,0)))

  {
    puts("CANNOT INIT STDOUT TASK\n");
    exit(3);
  }

  rt_task_use_fpu(stdtsk,1); // floating point for real time task "stdtsk"
  rt_linux_use_fpu(1); // floating point for foreground Linux processes

  while(!quit) {
    nbyte=rt_mbx_receive_timed(pmbx,msg_rcv,sizeof(stdoutMsg),(RTIME)100000000);

    write(rtfifo,(void *)msg_rcv,sizeof(stdoutMsg));

  }
  free((void *)msg_rcv);
  rt_mbx_delete(mbxstdout);


}
```

control.h

```
/* Transmission based servor actuator system control
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file control.h

*/

#ifndef _CONTROLLER_H
#define _CONTROLLER_H


//data structure for PID control
typedef struct
{
double Kp,Ki,Kd;//gains

double Ts;//sample time
char flag;//with or without windup,("w" or "n")
double e[3],u[2];//error and control signal
double ra[4]; //reference and actual (position and velocity)
}tbaPID;

typedef struct
{

double t[3];
double c[2];
double omegaMax;
double thetaCommandDVT;
double loadTorque;
double alphaRef;
double omegaRef;
double thetaRef;
}tbaTrj;

void positionalPID2D(tbaPID *);//2-D positional PID
void velocityPID2D(tbaPID *);//2-D velocity PID

void positionalPID1D(tbaPID *);//1-D positional PID
void velocityPID1D(tbaPID *);//1-D velocity PID

void trajectoryParams(tbaTrj *);

void tbaFeedbackControl(tbaPID *);

void initPID(tbaPID *); //initialization of PID stucture
```

```
void trajectoryRef(tbaTrj *,double);

#endif
/* Transmission based servor actuator system control
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file control.c
        This file defines the all the controller functions


*/

#include "control.h"
#include "common.h"
#include <math.h>

/** \function
        *Position PID control algorithm, 2-D design\n
   *
        *u[k]=Kp*e[k] + Ki*Ts* sum(e[i]) + Kd/Ts*(e[k]-e[k-1]) + u[0]\n
   *
*/

void positionalPID2D(tbaPID *pPID)
{
double Pk,Ik,Dk;

pPID->e[0]=pPID->ra[0]-pPID->ra[1];//initial error

pPID->e[2]=pPID->e[2]+pPID->e[0];//sum of errors

Pk=pPID->Kp*pPID->e[0];
Ik=pPID->Ki*pPID->Ts*pPID->e[2];
Dk=pPID->Kd/pPID->Ts*(pPID->e[0]-pPID->e[1]);


pPID->u[0]=pPID->u[1]+Pk+Ik+Dk;//u[1] is the initial control command

pPID->e[1]=pPID->e[0];

//printf("%f %f %f\n",Pk,Ik,Dk);

}


/** \function
Velocity PID control algorithm, 2-D design


u[k]=u[k-1] + Kp*(e[k]-e[k-i]) + Ki*Ts*e[k] + Kd/Ts*(e[k]-2*e[k-1]+e[k-2])
```

```
*/

void velocityPID2D(tbaPID *pPID)
{
double Pk,Ik,Dk;

pPID->e[0]=pPID->ra[0]-pPID->ra[1];//initial error

Pk=pPID->Kp*(pPID->e[0]-pPID->e[1]);
Ik=pPID->Ki*pPID->Ts*pPID->e[0];
Dk=pPID->Kd/pPID->Ts*(pPID->e[0]-2*pPID->e[1]+pPID->e[2]);

pPID->u[0]=pPID->u[1]+Pk+Ik+Dk;//u[1] is the previous control command

pPID->u[1]=pPID->u[0];

pPID->e[2]=pPID->e[1];

pPID->e[1]=pPID->e[0];

}


/** \function
Position PID control algorithm, 1-D design

u[k]=-Kp*(x[k]-x[]) + Ki*Ts* sum (e[i]) + Kd/Ts*(e[k]-e[k-1]) + u[0]

*/



void positionalPID1D(tbaPID *pPID)
{
double Pk,Ik,Dk;

pPID->e[0]=pPID->ra[0]-pPID->ra[1];//initial error

pPID->e[2]=pPID->e[2]+pPID->e[0];//sum of errors

Pk=pPID->Kp*pPID->e[0];
Ik=pPID->Ki*pPID->Ts*pPID->e[2];
Dk=pPID->Kd/pPID->Ts*(pPID->e[0]-pPID->e[1]);


pPID->u[0]=pPID->u[1]+Pk+Ik+Dk;//u[1] is the initial control command

pPID->e[1]=pPID->e[0];

//printf("%f %f %f\n",Pk,Ik,Dk);

}
```

```
/** \function
Velocity PID control algorithm, 1-D design


u[k]=u[k-1] + Kp*(e[k]-e[k-i]) + Ki*Ts*e[k] + Kd/Ts*(e[k]-2*e[k-1]+e[k-2])


*/

void velocityPID1D(tbaPID *pPID)
{
double Pk,Ik,Dk;

pPID->e[0]=pPID->ra[0]-pPID->ra[1];//initial error

Pk=pPID->Kp*(pPID->e[0]-pPID->e[1]);
Ik=pPID->Ki*pPID->Ts*pPID->e[0];
Dk=pPID->Kd/pPID->Ts*(pPID->e[0]-2*pPID->e[1]+pPID->e[2]);

pPID->u[0]=pPID->u[1]+Pk+Ik+Dk;//u[1] is the previous control command

pPID->u[1]=pPID->u[0];

pPID->e[2]=pPID->e[1];

pPID->e[1]=pPID->e[0];

}


/**
set up PID default initial value
*/

void initPID(tbaPID *pPID)
{
pPID->Kp=0;
pPID->Ki=0;
pPID->Kd=0;

pPID->Ts=PERIOD;

pPID->u[1]=0;

pPID->e[1]=0;

pPID->e[2]=0;

}

/** \function
symmetric trajectory generation parameters:(static)
*/
/*
Final position (thetaCommandDVT), time (t[3]) required, and maximum speed (omegaMax) are prescribed.
```

```
t[1]=2*thetaCommandDVT/omegaMax

t[0]=t[3]-t[2]

c[0]=3*omegaMax/t[0]^2

c[1]=-2*omegaMax/t[0]^3

*/

void trajectoryParams(tbaTrj *ptbaTrj)
{

ptbaTrj->t[1]=ptbaTrj->thetaCommandDVT/ptbaTrj->omegaMax;

ptbaTrj->t[0]=ptbaTrj->t[2]-ptbaTrj->t[1];

ptbaTrj->c[0]=3*ptbaTrj->omegaMax/(ptbaTrj->t[0]*ptbaTrj->t[0]);

ptbaTrj->c[1]=-2*ptbaTrj->omegaMax/pow(ptbaTrj->t[0],3);

//printf("c[0]=%f c[1]=%f t[0]=%f t[1]=%f t[2]=%f \n",ptbaTrj->c[0],ptbaTrj->c[1],ptbaTrj->t[0],ptbaTrj->t[1],ptbaTrj->t[2]);

}

/** \function
symmetric trajectory generation reference with time (time varying)
*/
/*
                /  c[0]*t^3/3+c[1]*t^4/4                                  t=[0,t[0])
                |
thetaRef(t) =|  omegaMax*t[0]/2+ omegaMax*(t-t[0])                  t=[t[0],t[1])
                |
                |  omegaMax*t[1]-c[0]*(t[2]-t)^3/3-c[1]*(t[2]-t)^4/4      t=[t[1],t[2])
                |
                \  thetaCommandDVT                                        else

                  /  c[0]*t^2+c[1]*t^3              t=[0,t[0])
                  |
omegaRef(t) =|  omegaMax                        t=[t[0],t[1])
                  |
                  |  c[0]*(t[2]-t)^2+c[1]*(t[2]-t)^3   t=[t[1],t[2])
                  |
                  \  0                                    else


                  /  2*c[0]*t+3*c[1]*t^2              t=[0,t[0])
                  |
alphaRef(t) =|  0                                  t=[t[0],t[1])
                  |
                  |  -2*c[0]*(t[2]-t)-3*c[1]*(t[2]-t)^2   t=[t[1],t[2])
                  |
                  \  0                                    else
```

```
iRef(t) = (J*alpharef(t)+f*omegaRef(t))/KT
*/


void trajectoryRef(tbaTrj *ptbaTrj,double rt_time)
{

   if(rt_time<=ptbaTrj->t[0])
    {
    ptbaTrj->thetaRef=ptbaTrj->c[0]*pow(rt_time,3.0)/3.0+ptbaTrj->c[1]*pow(rt_time,4.0)/4.0;

    ptbaTrj->omegaRef=ptbaTrj->c[0]*pow(rt_time,2.0)+ptbaTrj->c[1]*pow(rt_time,3.0);

    ptbaTrj->alphaRef=2.0*ptbaTrj->c[0]*rt_time+3.0*ptbaTrj->c[1]*pow(rt_time,2.0);

    }

    if(rt_time>ptbaTrj->t[0]&&rt_time<=ptbaTrj->t[1])
    {

    ptbaTrj->thetaRef=ptbaTrj->omegaMax*(-ptbaTrj->t[0]/2.0+rt_time);

    ptbaTrj->omegaRef=ptbaTrj->omegaMax;

    ptbaTrj->alphaRef=0.0;

    }

    if(rt_time>ptbaTrj->t[1]&&rt_time<=ptbaTrj->t[2])
    {

    ptbaTrj->thetaRef=ptbaTrj->omegaMax*ptbaTrj->t[1]-ptbaTrj->c[0]*pow(ptbaTrj->t[2]-
rt_time,3.0)/3.0-ptbaTrj->c[1]*pow(ptbaTrj->t[2]-rt_time,4.0)/4.0;

    ptbaTrj->omegaRef=ptbaTrj->c[0]*pow(ptbaTrj->t[2]-rt_time,2)+ptbaTrj->c[1]*pow(ptbaTrj-
>t[2]-rt_time,3.0);

    ptbaTrj->alphaRef=-2.0*ptbaTrj->c[0]*(ptbaTrj->t[2]-rt_time)-3.0*ptbaTrj->c[1]*pow(ptbaTrj-
>t[2]-rt_time,2.0);


    }

    if(rt_time>ptbaTrj->t[2])
    {


    ptbaTrj->omegaRef=0.0;

    ptbaTrj->alphaRef=0.0;

    }
}
```

```
/**
feedback control

*/
void tbaFeedbackControl(tbaPID *ptbaPID)
{

ptbaPID->e[0]=ptbaPID->ra[0]-ptbaPID->ra[1];//position error


ptbaPID->e[1]=ptbaPID->ra[2]-ptbaPID->ra[3];//velocity error

ptbaPID->e[2]=ptbaPID->e[2]+ptbaPID->e[0]*ptbaPID->Ts;//sum of position errors, Forward Eular

ptbaPID->u[0]=ptbaPID->Ki*ptbaPID->e[2]+ptbaPID->Kp*ptbaPID->e[0]+ptbaPID->Kd*ptbaPID-
>e[1];

}
```

loadcell.c

```
/* Transmission based servo actuator system control
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file loadcell.c
        This file defines the functions associated with the dynamomater and control


*/

#include <stdio.h>
#include <comedilib.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <getopt.h>
#include <ctype.h>
#include "common.h"
#include "ni6023e.h"
#include "ni6711.h"

/** \function
        read the loadcell voltage

*/
double readLoadcell()
{
        lsampl_t data;
        double voltage,torque;

        comedi_data_read(ni6023e.ni_daq,AI_DEVICE,0,0,AI_REF,&data);
        voltage=comedi_to_phys(data,ni6023e.rng,ni6023e.maxdata);
        torque=voltage*N-M_PER_VOLT;

        return torque;
}

/** \function
        dynamomater close loop control


*/

void dynaControl(tbaPID *pPID,float dvt_speed)
{
        int ret;
```

```
	int voltage_command;


	pPID->Kp=1000.0;
	pPID->Ki=00.0;
	pPID->Kd=00.0;
	//set torque output limit

	if(pPID->ra[0]>20.0)
	{
	pPID->ra[0]=20.0;
	printf("Warning :voltage command exceeds upper limit\n");
	}


	//read dynamometer current toruqe
	pPID->ra[1]=readLoadcell();


	//PID procedure
	velocityPID2D(pPID);


	//command to ni6711 AO_3
	voltage_command=-(int)pPID->u[0]+AO_ZERO;


	if(dvt_speed==0.0)
	{
	voltage_command=AO_ZERO;

	}


	//set voltage output limit
	if(voltage_command<0)
	{
	voltage_command=0;
	//printf("Warning :voltage command exceeds lower limit\n");
	}
	if(voltage_command>2024)
	{
	voltage_command=2024;
	//printf("Warning :voltage command exceeds upper limit\n");
	}

	ret=comedi_data_write(ni6711.ni_daq,AO_DEVICE,AO_2,AO_RANGE,AO_REF,voltage
_command);
	if(ret<0){
			comedi_perror(NI6711);
			}

}
```

motorcontrol.c

```
/* Transmission based servor actuator system control
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file motorcontrol.c
        This file defines the main motor control routines
        */
 #include <stdio.h>
 #include <string.h>
 #include <unistd.h>
 #include <fcntl.h>
 #include <sys/types.h>
 #include <sys/mman.h>
 #include <sys/stat.h>
 #include <stdlib.h>
 #include <pthread.h>
 #include <signal.h>
 #include <math.h>

 #define KEEP_STATIC_INLINE
 #include <rtai_lxrt_user.h>
 #include <rtai_lxrt.h>
 #include <rtai_fifos.h>
 #include <rtai_sem.h>
 #include <rtai_msg.h>
 #include <rtai_shm.h>

 #include "ni6023e.h"
 #include "ni6711.h"
 #include "common.h"
 #include "control.h"

 extern int quit;

 void motorControl(int c_id, int drv_mod)
  {

    unsigned long testcnttsk_name = nam2num("PIDVEL");

    RT_TASK *testcnttsk;

    int i,j,ret;

    int doPrintPID=1,isFirstPosition=1;

    double ior=0.0002618,cof=0.0;//inertia of rotor and coefficient of friction, should be put in
motor.h
```

```
char drive_mode[256];
char control_method[256];
char data_file_name[256];
char time_buffer[256];

double Mp;

//Mp=240.;
//Mp=0.0;
//Mp=120.0;
//Mp=60.0;
Mp=180.0;

double Ma=200.;
double L=71.0;
double g;
double Meq;

Mp=Mp*0.454;
Ma=Ma*0.454;
L=L*0.0254;
g=9.81;

Meq=(Mp+Ma/2.0)*L*g;

FILE *fp;

// double motor_speed,dvt_speed,ratio;
double pos_com_deg;

int pos_com_count,pos_act_count,vel_com_rpm,u=0,pos[3];

int to_gear,nbyte,gear_cur=3,gear_prev=3;

double shift_bound[2]={5.0,8.0};

double grey_region=1.0;//grey region width, N-m

double shift_time[2]={0.0,0.0};



struct sched_param mysched;
stdoutMsg *msg_send;

tbaPID *pDynaPID,*pMotorPID;

tbaTrj *ptbaTrj;


quit=0;
j=0;

double dvt_disp,dvt_disp_abs,motor_disp;
double motor_speed[MAFPNT],dvt_speed[MAFPNT],sum_speed,dvt_acc;
```

```
lsampl_t data;

double motor_current,voltage,load;

double rt_time,rt_time_current,rt_time_init,rt_time_temp;



time_t curtime;

struct tm *loctime;

//Get the current time.
curtime = time (NULL);

// Convert it to local time representation.
loctime = localtime (&curtime);

// Print it out in a nice format.
strftime (time_buffer, 256, "%B_%d_%Y_%kh%Mm", loctime);

switch(c_id)
{
 case 1: //P
 strcpy(control_method,"PControl_");
 break;

 case 2: //PID
 strcpy(control_method,"PIDControl_");

 break;

}

switch(drv_mod)
{
 case 1: //velocity mode
 strcpy(drive_mode,"VMode_");
 break;

 case 2: //current mode
 strcpy(drive_mode,"CMode_");
 break;
}

strcpy(data_file_name,"../../result/");
strcat(data_file_name,control_method);
strcat(data_file_name,drive_mode);
strcat(data_file_name,time_buffer);
strcat(data_file_name,".txt");

fp = fopen(data_file_name, "w+");
if(fp==NULL)
printf("tba data file open failed!\n");
```

```
fprintf(fp,"##############################################################################
#################################################\n\n");
   fprintf(fp,"#                Transmission based Servo Actuator system control\n\n");
   fprintf(fp,"#                ");
   fprintf(fp,data_file_name);



   pDynaPID=(tbaPID*)malloc(sizeof(tbaPID));
   if(pDynaPID==NULL)
   printf("PID malloc failed\n");

   pMotorPID=(tbaPID*)malloc(sizeof(tbaPID));
   if(pMotorPID==NULL)
   printf("PID malloc failed\n");

   //initialization of PID algorithm
   initPID(pDynaPID);

   initPID(pMotorPID);


   ptbaTrj=(tbaTrj*)malloc(sizeof(tbaTrj));
   if(ptbaTrj==NULL)
   printf("tbatrj malloc failed\n");


   //initialization of TBA trajectory

   //position command
   //TODO : need user interaction and/or automatic updated feature

   pos_com_deg=36000.;

   vel_com_rpm=300.0;

   pos_com_count=(int)rev2cnt(deg2rev(pos_com_deg));


   //unit conversion
   ptbaTrj->thetaCommandDVT=(double)deg2rad(pos_com_deg);  //radian
   ptbaTrj->omegaMax=rpm2rps(vel_com_rpm);         //rad/sec
   ptbaTrj->t[2]=22.00;                            //seconds



   //calculate trajectory parameters
   trajectoryParams(ptbaTrj);



   //initialize motor and dvt speed vector
   for(i=0;i<MAFPNT;i++)
```

```
    {
    dvt_speed[i]=0.0;
    motor_speed[i]=0.0;
    }


    // memory allocation
    msg_send=(stdoutMsg *)rtai_malloc(nam2num("stdMSG"),sizeof(stdoutMsg));


    //set priority and real time task
    mysched.sched_priority = 99;


    if (!(testcnttsk = rt_task_init_schmod(testcnttsk_name, 1, 0, 0,SCHED_FIFO,1))) {
                        printf("CANNOT INIT MASTER TASK\n");
                        exit(1);
              }

   if( sched_setscheduler( 0, SCHED_FIFO, &mysched ) == -1 )
            {
            puts(" ERROR IN SETTING THE SCHEDULER UP");
             perror( "errno" );
             exit( 0 );
             }

    // make task periodic excution
    rt_task_make_periodic(testcnttsk,
rt_get_time()+(RTIME)(PERIOD*ticks_per_second+1.0),(RTIME)(PERIOD*ticks_per_second));

    //read initial motor and DVT encoder count
    lsampl_t motor_pos_first= counterRead(ni6023e.ni_daq,CNT_DEVICE,1);
    lsampl_t dvt_pos_first= counterRead(ni6023e.ni_daq,CNT_DEVICE,0);

    lsampl_t motor_pos_last= motor_pos_first;
    lsampl_t dvt_pos_last= dvt_pos_first;


    //initialize timers
    rt_time_init=1.0*rt_get_time();
    rt_time_temp=rt_time_init;

    //make hard real time
    rt_make_hard_real_time();


    // main control loop
    while(1)

      {

      // get time, absolute and relative
      rt_time_current=1.0*rt_get_time();

      rt_time=rt_time_current-rt_time_init;
```

```
//get position, motor and DVT

lsampl_t motor_pos_current= counterRead(ni6023e.ni_daq,CNT_DEVICE,1);
lsampl_t dvt_pos_current= counterRead(ni6023e.ni_daq,CNT_DEVICE,0);


//initialize position array to the same first encoder reading
if(isFirstPosition)
{
for(j=0;j<3;j++)
pos[j]=dvt_pos_current;
isFirstPosition=0;
}

//shift array element to the left by 1 position
if(!isFirstPosition)
{
for(i=0;i<2;i++)
pos[i]=pos[i+1];
pos[2]=dvt_pos_current;
}


//printf("motor position: %d (%d, abs: %d)\n", motor_pos_current,motor_pos_current-motor_pos_last,motor_pos_current-motor_pos_first);
//printf("%d (%d, abs: %d)\n", dvt_pos_current,dvt_pos_current-dvt_pos_last,dvt_pos_current-dvt_pos_first);
//printf("command motor speed= %.2f, DVT output speed=%.2frpm, gear ratio=%f\n\n",motor_speed, dvt_speed,ratio);

pos_act_count=dvt_pos_current-dvt_pos_first;


//read loadcell voltage


comedi_data_read(ni6023e.ni_daq,AI_DEVICE,AI_1,0,AI_REF,&data);

voltage=comedi_to_phys(data,ni6023e.rng,ni6023e.maxdata);

load=voltage*N-M_PER_VOLT;



//read motor current from pin 22 and 23 (Analog Current out)


comedi_data_read(ni6023e.ni_daq,AI_DEVICE,AI_2,0,AI_REF,&data);

voltage=comedi_to_phys(data,ni6023e.rng,ni6023e.maxdata);

motor_current=voltage;//zero reading when motor is at rest, will vary, this is not a solution.
```

```
//calculate dvt angular position
dvt_disp_abs=cnt2rad((double)dvt_pos_current-(double)dvt_pos_first);

gear_cur=0;

pDynaPID->ra[0]=Meq*sin(dvt_disp_abs/200.)/200.; //reference torque input

//printf("torque=%f\n", pDynaPID->ra[0]);

/*
//send out gear command to mailbox (time based switch signal)

if(msg_send->time>=0.0)
to_gear=1;

if(msg_send->time>=5.0)
to_gear=2;

if(msg_send->time>=10.0)
to_gear=3;

if(msg_send->time>=20.0)
to_gear=0;
*/

 //send out gear command to mailbox (state-based switch signal)
/*
if(pDynaPID->ra[0]>=0.0&&pDynaPID->ra[0]<=5.0)
{
to_gear=3;
if(gear_cur==to_gear)
to_gear=0;
}

if(pDynaPID->ra[0]>5.0&&pDynaPID->ra[0]<=8.0)
{
to_gear=2;
if(gear_cur==to_gear)
to_gear=0;
}

if(pDynaPID->ra[0]>8.0)

{
to_gear=1;
if(gear_cur==to_gear)
to_gear=0;
}
*/

//state based switch signal- one boundary
/*
if(load<=5.0)
{
to_gear=3;
if(gear_cur==to_gear)
```

```
to_gear=0;
}

if(load>5.0&&load<=8.0)
{
to_gear=2;
if(gear_cur==to_gear)
to_gear=0;
}

if(load>8.0)

{
to_gear=1;
if(gear_cur==to_gear)
to_gear=0;
}
*/

//hybrid switch signal- two boundaries

//3rd gear

if((load<=shift_bound[0]))
{
if(gear_prev==3)
to_gear=3;//stay put

if(load<=(shift_bound[0]-grey_region)&&(gear_prev==2))
{
shift_time[0]=shift_time[0]+0.001;
if(shift_time[0]>=0.5)
{
to_gear=3;//new shift signal
shift_time[0]=0.0;
}
}

gear_prev=to_gear;//save previous gear

if(gear_cur==to_gear)
{
to_gear=0;
}
}
//2nd gear
else if((load>shift_bound[0])&&(load<=shift_bound[1]))
{
if(gear_prev==2||(gear_prev==3))
to_gear=2;//new signal if previous gear is 1


if(load<=(shift_bound[1]-grey_region)&&(gear_prev==1))
{
shift_time[1]=shift_time[1]+0.001;
if(shift_time[1]>=0.5)
```

```
{
to_gear=2;//new shift signal
shift_time[1]=0.0;
}
}
gear_prev=to_gear;//save previous gear

if(gear_cur==to_gear)
{
to_gear=0;
}

}

//third gear
else if(load>shift_bound[1])
{

to_gear=1;
gear_prev=to_gear;//save previous gear
if(gear_cur==to_gear)
{
to_gear=0;
}
}



//send out gear shift command
nbyte=rt_mbx_send_if(mbxgear,(void*)&to_gear,sizeof(int));
if(nbyte)
{
      printf("%d unsent bytes\n",nbyte);

}

//save current gear
gear_cur=gear_prev;

//calculate dvt speed using MAFPNT point moving average
dvt_disp=cnt2rad((double)dvt_pos_current-(double)dvt_pos_last);


sum_speed=0.0;
for(i=0;i<MAFPNT-1;i++)
{
dvt_speed[i]=dvt_speed[i+1];
sum_speed=sum_speed+dvt_speed[i];
}
dvt_speed[MAFPNT-1]=dvt_disp/((rt_time_current-rt_time_temp)/ticks_per_second);
sum_speed=sum_speed+dvt_speed[4];
msg_send->dvt_speed=sum_speed/MAFPNT;


dvt_acc=msg_send->dvt_speed/((rt_time_current-rt_time_temp)/ticks_per_second);
```

```
 //calculate motor speed using MAFPNT point moving average
 motor_disp=cnt2rad((double)motor_pos_current-(double)motor_pos_last);

 sum_speed=0.0;
 for(i=0;i<MAFPNT-1;i++)
 {
 motor_speed[i]=motor_speed[i+1];
 sum_speed=sum_speed+motor_speed[i];
 }
 motor_speed[MAFPNT-1]=motor_disp/((rt_time_current-rt_time_temp)/ticks_per_second);
 sum_speed=sum_speed+motor_speed[MAFPNT-1];
 msg_send->motor_speed=sum_speed/MAFPNT;


//controller setup

//new controller can be added here;

//pMotorPID->ra[0]=(double)pos_com_count;
//pMotorPID->ra[1]=(double)pos_act_count;


//trajectory reference position, velocity, and acceleration
trajectoryRef(ptbaTrj,rt_time/ticks_per_second);



// initialize PID entry value
pMotorPID->ra[0]=ptbaTrj->thetaRef;
pMotorPID->ra[2]=ptbaTrj->omegaRef;

//pMotorPID->ra[1]=cnt2rad(pos_act_count);

pMotorPID->ra[1]=cnt2rad((double)pos_act_count);

pMotorPID->ra[3]=(double)msg_send->dvt_speed;

switch(c_id)
{
case 1: //P

 pMotorPID->Kp=0.1;
 pMotorPID->Kd=0.0;
 pMotorPID->Ki=0.0;


 tbaFeedbackControl(pMotorPID);

 //positionalPID2D(pMotorPID);

//u=Kp*(-pos_err+AO_ZERO);//zero offset

 u=-(int)pMotorPID->u[0]+AO_ZERO;

 //positive satuation, 12 bit D/A
```

```
    if (u>2*AO_ZERO)
    {
      u=4096;
      }

    //negative satuation, 12 bit D/A
    if (u<0)
    {
      u=0;
      }

    break;


    case 2: //PID

    pMotorPID->Kp=1800.5;


    if((ptbaTrj->thetaCommandDVT-ptbaTrj->thetaRef)<=0.0005)
    {
    //pMotorPID->Kp=950.5;
    //pMotorPID->Ki=500.0;
    //pMotorPID->Kd=1000.0;

    //printf("%f %f %f %f %d \n",pMotorPID->e[0],pMotorPID->u[0],pMotorPID->ra[0],pMotorPID-
>ra[1],pos_act_count);
    //printf("%f %f \n",ptbaTrj->thetaCommandDVT, ptbaTrj->thetaRef);
    }

    pMotorPID->Ki=0.0;
    pMotorPID->Kd=45.0;


    tbaFeedbackControl(pMotorPID);
    //positionalPID2D(pMotorPID);

    u=-(int)((ior*ptbaTrj->alphaRef+cof*ptbaTrj-
>omegaRef)/MOTOR_TORQUE_CONSTANT+pMotorPID->u[0])+AO_ZERO;


    //u=-(int)pMotorPID->u[0]+AO_ZERO;

    //printf("%d\n",u);

    //positive satuation, 12 bit D/A
    if (u>2*AO_ZERO)
    {
      u=4096;
      }

    //negative satuation, 12 bit D/A
    if (u<0)
    {
      u=0;
      }
```

```
	break;


	}

	// send analog velocity referent voltage

	ret=comedi_data_write(ni6711.ni_daq,AO_DEVICE,AO_4,AO_RANGE,AO_REF,u);

	if(ret<0){
			comedi_perror(NI6711);
			//exit(0);
	}




	//frequency=ticks_per_second/(rt_time-rt_time_temp);


	//send out standard output data to mailbox


	msg_send->time=rt_time/ticks_per_second;

	msg_send->dvt_position=dvt_pos_current-dvt_pos_first;

	msg_send->command_position=(int)rad2cnt(ptbaTrj->thetaRef);

	msg_send->motor_position=motor_pos_current-motor_pos_first;

	msg_send->gear_current=to_gear;

	msg_send->load=load;

	msg_send->motor_torque=motor_current*MOTOR_TORQUE_CONSTANT;



	//nbyte=rt_mbx_send_timed(mbxstdout,(void
*)msg_send,sizeof(stdoutMsg),(RTIME)10000000);
	//nbyte=rt_mbx_send_if(mbxstdout,(void*)msg_send,sizeof(stdoutMsg));
	nbyte=rt_mbx_send_if(mbxstdout,(void*)msg_send,sizeof(stdoutMsg));
		if(nbyte)
		{
		//printf("%d unsent bytes\n",nbyte);

		}


	if(doPrintPID)
	{
```

```
    fprintf(fp, "\n\n#       Kp=%f  Ki=%f   Kd=%f   #\n",pMotorPID->Kp,pMotorPID->Ki,pMotorPID-
>Kd);
    doPrintPID=0;

    fprintf(fp,"\n#       time        motor_spd      dvt_spd       load_tor       motor_tor  com_pos
dvt_pos  motor_pos  gear\n");
    fprintf(fp,"#     (seconds)     (rad/sec)     (rad/sec)       (N-m)          (N-m)     (count)  (count)
(count)\n\n");

fprintf(fp,"#########################################################################
##############################################\n\n");
    }

    if(msg_send->time<=35.)

    fprintf(fp,"%14f %14f %14f %14f %14f %8d %8d %8d %8d \n",msg_send->time,msg_send-
>motor_speed,msg_send->dvt_speed,msg_send->load,    msg_send->motor_torque,msg_send-
>command_position,msg_send->dvt_position,          msg_send->motor_position,msg_send-
>gear_current);


    //dynamometer toruqe command

    dynaControl(pDynaPID,msg_send->dvt_speed);


    rt_time_temp=rt_time_current;

    motor_pos_last= motor_pos_current;
    dvt_pos_last= dvt_pos_current;

    rt_task_wait_period();

    }

  counterDisarm(ni6023e.ni_daq,CNT_DEVICE,0);
  counterDisarm(ni6023e.ni_daq,CNT_DEVICE,1);

  rt_make_soft_real_time();

  //comedi_close(ni6023e.ni_daq);

  rtai_free(nam2num("stdMSG"),(void *)msg_send);


  rt_task_delete(testcnttsk);

  free(pMotorPID);
  free(pDynaPID);
  fclose(fp);
  return;

}
```

ni6023e.h

```
/* Transmission based servor actuator system control
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file ni6023e.h


*/
#ifndef _NI6023E_H
#define _NI6023E_H

#include <comedilib.h>


//device name
#define NI6023E "/dev/comedi0"

/*Analog input*/

//subdevice number
#define AI_DEVICE 0

//channel list (16 SE/8 DI)
#define AI_1 0
#define AI_2 1
#define AI_3 2
#define AI_4 3
#define AI_5 4
#define AI_6 5
#define AI_7 6
#define AI_8 7

#define AI_OFFSET 0.610


//reference type, range, requency, etc.
#define AI_REF AREF_DIFF

#define AI_RANGE 0
#define AI_CHANNEL 8;
#define AI_FREQ 1000.0
#define AI_SCAN 1000


/*Counter*/

//subdevice number
#define CNT_DEVICE 4
```

```
//channel list
#define CNT_1 0
#define CNT_2 1

/**
 * Definitions of some of the common code.
 */

comedi_t *init_6023e();

int counterReset(comedi_t *dev, int subdev, int channel);

int counterSetSource(comedi_t *dev, int subdev, int channel, int SrcType);

int counterSetGate(comedi_t *dev, int subdev, int channel, int GateType);

int counterSetDirection(comedi_t *dev, int subdev, int channel, int Direction);

int counterSetOperation(comedi_t *dev, int subdev, int channel, int Operation, int OptParam);

int counterArm(comedi_t *dev, int subdev, int channel);

int counterDisarm(comedi_t *dev, int subdev, int channel);

lsampl_t counterRead(comedi_t *dev, int subdev, int channel);

#endif
```

ni6023e.c

```
/* Transmission based servor actuator system control
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file ni6023e.c
        This file defines the function acssociated with the NI-PCI6023E data acquisition board



*/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h> //fork()
#include <sys/types.h>//pid_t *
#include <sys/wait.h>
#include "ni6023e.h"
#include "common.h"



/** \function
        initilize ni-pci6023e


*/

comedi_t *init_6023e()
{
        int stype;
      int i;
        pid_t pid;
        int status;
        //char *cmd[]={"/usr/local/comedilib/comedi_calibrate/comedi_calibrate","",(char *)0};

        comedi_t *device;


        //open NI-6023E
        device=comedi_open(NI6023E);
        if(!device){
                comedi_perror(NI6023E);
                printf("%s open fail\n",NI6023E);
                exit(0);
        }
```

```
	//check analog out availability
	stype = comedi_get_subdevice_type(device,AI_DEVICE);
	if(stype!=COMEDI_SUBD_AI){
		printf("%d is not an analog input subdevice\n",AI_DEVICE);
		exit(0);
	}

	//calibrate AI device

	pid=fork();

	if (pid==-1)
	{
	printf("fork failed!\n");
	exit(0);
	}

	if(pid==0)
	{
	execv("/usr/local/comedilib/comedi_calibrate/comedi_calibrate","");
	_exit (EXIT_FAILURE);
	}

	if (pid==1)
	{
	printf("I am parent, waiting\n");
	waitpid(pid,&status,0);
	}

	//wait for calibration finish
	//sleep(20);

	//get some parameters
    ni6023e.n_ranges=comedi_get_n_ranges(device,AI_DEVICE,AI_1);
	ni6023e.maxdata=comedi_get_maxdata(device,AI_DEVICE,AI_1);
    ni6023e.rng = comedi_get_range(device,AI_DEVICE,AI_1,0);
	ni6023e.n_channels=comedi_get_n_channels(device,AI_DEVICE);
	ni6023e.offset=comedi_from_phys(0.0,ni6023e.rng,ni6023e.maxdata);


    printf("n_ranges=%d,                                                    maxdata=%d
offset=%d\n",ni6023e.n_ranges,ni6023e.maxdata,ni6023e.offset);



	//check counter availability

	stype = comedi_get_subdevice_type(device,CNT_DEVICE);
	if(stype!=COMEDI_SUBD_COUNTER){
		printf("%d is not a counter subdevice\n",CNT_DEVICE);
		exit(0);
	}


	//configure counter device, ready to read count
```

```
        for(i=0;i<=1;i++)
           {
              counterReset(device,CNT_DEVICE,i);
                 counterSetSource(device,CNT_DEVICE,i,GPCT_EXT_PIN);
                 counterSetGate(device,CNT_DEVICE,i,GPCT_NO_GATE);
                 counterSetDirection(device,CNT_DEVICE,i,GPCT_HWUD);
                 counterSetOperation(device,CNT_DEVICE,i,GPCT_SIMPLE_EVENT,-1);
                 counterArm(device,CNT_DEVICE,i);
                 }

        /*
        if(verbose)
        {
      printf("writing %d to device=%s subdevice=%d channel=%d range=%d analog
reference=%d\n",
        maxdata,NI6711,AO_DEVICE,i,AO_RANGE,AO_REF);
        }
        */


        return device;

}


/** \function
        reset counter for new task


*/

int counterReset(comedi_t *dev, int subdev, int channel)
  {
    comedi_insn insn;
    lsampl_t params[]= { GPCT_RESET };  // the config subcommand: reset counter

    insn.insn= INSN_CONFIG;   // it is a configuration sub-command
    insn.n= 1;                // the parameter-array contains 1 element
    insn.data= params;        // pass parameters
    insn.subdev= subdev;      // which subdevice controls the counter?
    insn.chanspec= channel; // tell which counter to use

    return comedi_do_insn(dev, &insn);
  }


/** \function
        set a counter source


*/

int counterSetSource(comedi_t *dev, int subdev, int channel, int SrcType)
  {
```

```
   comedi_insn insn;
   lsampl_t params[]= { GPCT_SET_SOURCE, SrcType };   // the config subcommand: set source, and the add. param.: type of the source

   insn.insn= INSN_CONFIG;   // it is a configuration sub-command
   insn.n= 2;                // the parameter-array contains 2 elements
   insn.data= params;        // pass parameters
   insn.subdev= subdev;      // which subdevice controls the counter?
   insn.chanspec= channel;   // tell which counter to use

   return comedi_do_insn(dev, &insn);
  }


/** \function
            set counter gate type

*/

int counterSetGate(comedi_t *dev, int subdev, int channel, int GateType)
  {
   comedi_insn insn;
   lsampl_t params[]= { GPCT_SET_GATE, GateType };  // the config subcommand: set source, and the add. param.: type of the gate

   insn.insn= INSN_CONFIG;   // it is a configuration sub-command
   insn.n= 2;                // the parameter-array contains 2 elements
   insn.data= params;        // pass parameters
   insn.subdev= subdev;      // which subdevice controls the counter?
   insn.chanspec= channel;   // tell which counter to use

   return comedi_do_insn(dev, &insn);
  }

/** \function
            set counter direction-following edge, rising edge or both

*/

  int counterSetDirection(comedi_t *dev, int subdev, int channel, int Direction)
  {
   comedi_insn insn;
   lsampl_t params[]= { GPCT_SET_DIRECTION, Direction };   // the config subcommand: set direction, and the add. param.: direction

   insn.insn= INSN_CONFIG;   // it is a configuration sub-command
   insn.n= 2;                // the parameter-array contains 2 elements
   insn.data= params;        // pass parameters
   insn.subdev= subdev;      // which subdevice controls the counter?
   insn.chanspec= channel;   // tell which counter to use

   return comedi_do_insn(dev, &insn);
  }


/** \function
```

```
        set counter operations


*/

  int counterSetOperation(comedi_t *dev, int subdev, int channel, int Operation, int OptParam)
  {
    comedi_insn insn;
    lsampl_t params[]= { GPCT_SET_OPERATION, Operation, OptParam };   // the config
subcommand: set source, and the add. param.: operation

    insn.insn= INSN_CONFIG;   // it is a configuration sub-command
    insn.n= OptParam == -1 ? 2 : 3;  // the parameter-array contains 2 or 3 elements
    insn.data= params;        // pass parameters
    insn.subdev= subdev;      // which subdevice controls the counter?
    insn.chanspec= channel;   // tell which counter to use

    return comedi_do_insn(dev, &insn);
  }


/** \function
        arm a counter


*/

int counterArm(comedi_t *dev, int subdev, int channel)
  {
    comedi_insn insn;
    lsampl_t params[]= { GPCT_ARM };  // the config subcommand: arm

    insn.insn= INSN_CONFIG;   // it is a configuration sub-command
    insn.n= 1;                // the parameter-array contains 1 element
    insn.data= params;        // pass parameters
    insn.subdev= subdev;      // which subdevice controls the counter?
    insn.chanspec= channel;   // tell which counter to use

    return comedi_do_insn(dev, &insn);
  }

/** \function
        disarm a counter


*/

int counterDisarm(comedi_t *dev, int subdev, int channel)
  {
    comedi_insn insn;
    lsampl_t params[]= { GPCT_DISARM };  // the config subcommand: disarm

    insn.insn= INSN_CONFIG;   // it is a configuration sub-command
    insn.n= 1;                // the parameter-array contains 1 element
    insn.data= params;        // pass parameters
    insn.subdev= subdev;      // which subdevice controls the counter?
```

```
   insn.chanspec= channel;   // tell which counter to use

   return comedi_do_insn(dev, &insn);
  }


/** \function
            read a counter value


*/

lsampl_t counterRead(comedi_t *dev, int subdev, int channel)
  {
   comedi_insn insn;
   lsampl_t value= 0;

   insn.insn= INSN_READ;     // it is a read command
   insn.n= 1;                // the size of the parameter array is 1 (1 value is passed back, more is not
supported by this instruction)
   insn.data= &value;        // pass destination "array" (or simple pointer to a single tsampl_t)
   insn.subdev= subdev;      // which subdevice controls the counter?
   insn.chanspec= channel;   // tell which counter to use

   if (comedi_do_insn(dev, &insn) < 0)
     printf("[Warning] comedi_do_insn failed.\n            ComediCounterRead: returned value is not
valid.\n");

   return value;
  }
```

ni6711.h

```
/* Transmission based servor actuator system control
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file ni6711.h


*/
#ifndef _NI6711_H
#define _NI6711_H

#include <comedilib.h>

//device name
#define NI6711 "/dev/comedi1"

/*Analog out*/

//subdevice number
#define AO_DEVICE 1

//channel list
#define AO_1 0
#define AO_2 1
#define AO_3 2
#define AO_4 3

//reference type, range, requency, etc.
#define AO_REF AREF_GROUND

#define AO_RANGE 0
#define AO_CHANNEL 4;
#define AO_FREQ 1000.0
#define AO_SCAN 1000


/*Digital Input/Out*/

//subdevice number
#define DIO_DEVICE 2

//channel list
#define DIO_1 0
#define DIO_2 1
#define DIO_3 2
#define DIO_4 3
#define DIO_5 4
```

```
#define DIO_6 5
#define DIO_7 6
#define DIO_8 7

/*
//brake engage/disengae
#define ENGAGE 1
#define DISENGAGE 0
*/
//input/output
#define DO COMEDI_OUTPUT
#define DI COMEDI_INPUT

//number of DIO channels
#define DIO_CHANNEL 8;

//zero analog output

#define AO_ZERO 2024


comedi_t *init_6711();

#endif
```

ni6711.c

```
/* Transmission based servor actuator system control
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file ni6711.c
        This file defines the function acssociated with the NI-PCI6023E data acquisition board

*/


#include <stdlib.h>
#include "ni6711.h"
#include "common.h"

/** \function
        initilize ni-pci6711

*/
comedi_t *init_6711()
{

        int ret;
        int stype;
        int i;

        comedi_t *device;

        int amplitude;


        //open NI-6711
        device=comedi_open(NI6711);
        if(!device){
                comedi_perror(NI6711);
                printf("%s open fail\n",NI6711);
                exit(ERROR);
        }

      //check digital io availability
        stype = comedi_get_subdevice_type(device,DIO_DEVICE);
        if(stype!=COMEDI_SUBD_DIO){
                printf("%d is not a digital I/O subdevice\n",DIO_DEVICE);
                exit(ERROR);
        }

        //configure digital i/o for output
        for (i=0;i<7;i++)
```

```
        {
        ret=comedi_dio_config(device,DIO_DEVICE,i,DO);
        if(ret<0)
        {
        printf("channel %d of %d configure fail!\n ",i,DIO_DEVICE);
        exit(ERROR);
        }
        }

        //check Analog out availability and initialize to zero

        stype = comedi_get_subdevice_type(device,AO_DEVICE);
        if(stype!=COMEDI_SUBD_AO){
                printf("%d is not a analog output subdevice\n",AO_DEVICE);
                exit(ERROR);
        }

        //configure analog device
        for (i=0;i<=3;i++)
        {

        ni6711.maxdata= comedi_get_maxdata(device,AO_DEVICE,i);

        ni6711.rng = comedi_get_range(device,AO_DEVICE,i,AO_RANGE);

        ni6711.offset=comedi_from_phys(0.0,ni6711.rng,ni6711.maxdata);
        amplitude = comedi_from_phys(1.0,ni6711.rng,ni6711.maxdata) - ni6711.offset;


        }
        //rt_sleep(nano2count(10000000000));
    //printf("writing %d to device=%s subdevice=%d offset=%d analog
reference=%d\n",ni6711.maxdata,NI6711,AO_DEVICE,ni6711.offset,amplitude);
        //printf("device=%s subdevice=%d offset=%d analog
reference=%d\n",NI6711,AO_DEVICE,ni6711.offset,2048);

        //set brake motor current limit reference voltage (10 V is about 3Amps)
        ret=comedi_data_write(device,AO_DEVICE,0,AO_RANGE,AO_REF,1300);
    if(ret<0){
                comedi_perror(NI6711);
                exit(0);
        }

        //set BLDC motor speed/current refernce to zero
        ret=comedi_data_write(device,AO_DEVICE,3,AO_RANGE,AO_REF,2024);
    if(ret<0){
                comedi_perror(NI6711);
                exit(0);
        }

        //configure digital I/O for output only
        comedi_dio_config(device,DIO_DEVICE,DIO_1,COMEDI_OUTPUT);
        comedi_dio_config(device,DIO_DEVICE,DIO_2,COMEDI_OUTPUT);
        comedi_dio_config(device,DIO_DEVICE,DIO_3,COMEDI_OUTPUT);
        return device;
}
```

setmode.h

```
/* Transmission based servor actuator system control
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file setmode.h


*/

#ifndef _SETMODE
#define _SETMODE
#define OPERAND_cc 256
#define WAITDONE  rt_sleep(nano2count(50000000));
#define GIGA 1000000000

void checkSum(char *,int);
void analog_velocity_mode(void);
void *setMode(int mode);
#endif
```

setmode.c

```
/* Transmission based servor actuator system control
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file setmode.c
        This file defines the function acssociated with servo BLDC configuration through a serial
port



*/
   #include <stdio.h>
   #include <string.h>
   #include <unistd.h>
   #include <fcntl.h>

   #include <sys/types.h>
   #include <sys/mman.h>
   #include <sys/stat.h>

   #include <rtai_lxrt_user.h>
   #include <rtai_lxrt.h>
   #include <rtai_serial.h>
   #include <rtai_shm.h>

   #include "setmode.h"

extern int quit;

/** \function
        caculate the command string


*/

void u3kSerCom(char *com_str,int str_len){

         int i;
         int a_sum;
         int c_sum;
         char *ctemp;
         char *com_init;


        //printf("\n\n%s \n",com_str);

        if( (ctemp=(char*)rtai_malloc(nam2num("tempString"),str_len+4))==0)
        {
```

```
            printf("malloc failed!\n");
            exit(0);
            }

             if( (com_init=(char *)rtai_malloc(nam2num("commandString"),str_len+4))==0)
            {
            printf("malloc failed!\n");
            exit(0);
            }


            strcpy(com_init,":");//insert ":" as the first element of the array

            //printf("%d %s\n",str_len,com_init);

             //calculate checksum
             a_sum=0;

             for(i=0;i<str_len;i++)

            {
                        a_sum+=com_str[i];
            }
            c_sum=OPERAND_cc-a_sum;

            sprintf(ctemp,"%X",c_sum);

    //printf("ctemp=%s\n",ctemp);

    //append command, HEX checksum and a carriage return

    ctemp[0]=ctemp[strlen(ctemp)-2];
    ctemp[1]=ctemp[strlen(ctemp)-1];
    ctemp[2]='\r';

    //printf(" %s\n",ctemp);

    strcat(com_init,com_str);

    strncat(com_init,ctemp,3);

    //printf("sizes %d  %d\n",strlen(com_str),strlen(ctemp));
    //printf("%s\n",com_init);


    rt_spwrite(COM1, com_init,strlen(com_str)+4);
    WAITDONE
    rtai_free((void *)ctemp,str_len+4);
    rtai_free((void *)com_init,str_len+4);

}

/*!

*/
```

```
/** \function
        set up BLDC drive mode to analog velocity mode

u3000 serial command format:(see u3k host command-2098-RM003A-EN-P-October 2001.pdf)\n
-----------------------------------------------------------------------------------------\n
start   address         parameter       function data           checksum        end\n
-----   -------         ---------  --------  ------  ---------  ----\n
  :       aa              ppp             f           d...d           cc              <CR>\n
-----------------------------------------------------------------------------------------\n
*/

void analog_velocity_mode(void)
{

  char disableDrive[]="0006B100"; //disable drive
  char analogVelMode[]="0005A100"; //set analog velocity mode
  char analogVelScale[]="0011810020";//set analog velocity scale, in units of percentage of motor
maximum speed per 10 volts.
  char analogVelOffset[]="000471FFFB";//set analog velocity offset in units of millivolts
  char enableDrive[]="0006B101"; //enable drive
  char velocityLoopPgain[]="0001F1020";//velocity loop proportional gain=32d
  char velocityLoopIgain[]="000201020";//velocity loop integral gain=32d
  char velocityLoopDgain[]="0002110000";//velocity loop differentilal gain=0d
  char forwardCurrentLimit[]="0002F160"; //forward current limit, in unit of percentage of motor
interrim current or drive interim current
  char reverseCurrentLimit[]="00030160"; //reverse current limit, in unit of percentage of motor
interrim current or drive interim current

  //char read_buff[256];


  /*
  if( (commandString=malloc(50))==NULL)
          {
          printf("malloc failed!\n");
          exit(0);
          }
  */

  u3kSerCom(disableDrive,strlen(disableDrive));
  //printf("Disable drive...");
  //rt_spwrite(COM1, commandString,strlen(disableDrive)+4);
  //WAITDONE
  //printf("Done!\n");
  //rt_spread(COM1,read_buff,18);
  //WAITDONE
  //printf("Response is %s\n",read_buff);
  //printf("%s\n",disableDrive);


  u3kSerCom(forwardCurrentLimit,strlen(forwardCurrentLimit));
  /*
  //printf("Setting forward current limit...");
  rt_spwrite(COM1, commandString,strlen(forwardCurrentLimit)+4);
  WAITDONE
  rt_spread(COM1,read_buff,18);
```

```
WAITDONE
//printf("Response is %s\n",read_buff);
printf("      %s\n",forwardCurrentLimit);
//printf("Done!\n");
*/


u3kSerCom(reverseCurrentLimit,strlen(reverseCurrentLimit));
/*
//printf("Setting reverse current limit...");
rt_spwrite(COM1, commandString,strlen(reverseCurrentLimit)+4);
WAITDONE
rt_spread(COM1,read_buff,18);
WAITDONE
//printf("Response is %s\n",read_buff);
//printf("%s\n",commandString);
//printf("Done!\n");
*/

//printf("Setting analog velocity mode...");
u3kSerCom(analogVelMode,strlen(analogVelMode));
/*
rt_spwrite(COM1, commandString,strlen(analogVelMode)+4);
WAITDONE
rt_spread(COM1,read_buff,18);
WAITDONE
//printf("Response is %s\n",read_buff);
//printf("%s\n",commandString);
//printf("Done!\n");
*/

//printf("Setting analog velocity scale...");
u3kSerCom(analogVelScale,strlen(analogVelScale));
/*/
rt_spwrite(COM1, commandString,strlen(analogVelScale)+4);
WAITDONE
rt_spread(COM1,read_buff,18);
WAITDONE
//printf("Response is %s\n",read_buff);
//printf("%s\n",commandString);
//printf("Done!\n");
*/

//printf("Setting analog velocity offset...");
u3kSerCom(analogVelOffset,strlen(analogVelOffset));
/*
rt_spwrite(COM1, commandString,strlen(analogVelOffset)+4);
WAITDONE
rt_spread(COM1,read_buff,18);
WAITDONE
//printf("Response is %s\n",read_buff);
//printf("%s\n",commandString);
//printf("Done!\n");
*/

//printf("Setting velocity loop PID gains...");
```

```
  u3kSerCom(velocityLoopPgain,strlen(velocityLoopPgain));
  /*
  rt_spwrite(COM1, commandString,strlen(velocityLoopPgain)+4);
  WAITDONE
  rt_spread(COM1,read_buff,18);
  WAITDONE
  //printf("Response is %s\n",read_buff);
  //printf("%s\n",commandString);
  */

  u3kSerCom(velocityLoopIgain,strlen(velocityLoopIgain));
  /*
  rt_spwrite(COM1, commandString,strlen(velocityLoopIgain)+4);
  WAITDONE
  rt_spread(COM1,read_buff,18);
  WAITDONE
  //printf("Response is %s\n",read_buff);
  //printf("%s\n",commandString);
  */
  u3kSerCom(velocityLoopDgain,strlen(velocityLoopDgain));
  /*
  rt_spwrite(COM1, commandString,strlen(velocityLoopDgain)+4);
  WAITDONE
  rt_spread(COM1,read_buff,18);
  //printf("Response is %s\n",read_buff);
  //printf("Done!\n");
  //printf("%s\n",commandString);
  */

  //printf("Enable Drive...");
  u3kSerCom(enableDrive,strlen(enableDrive));
  /*
  rt_spwrite(COM1, commandString,strlen(enableDrive)+4);
  WAITDONE
  rt_spread(COM1,read_buff,18);
  //WAITDONE
 // printf("Response is %s\n",read_buff);
    //printf("Done!\n");
  //printf("%s\n",commandString);

  */
}

/** \function
          set up BLDC drive mode to analog current mode

*/

void analog_current_mode(void)
{

  //disable drive
  char disableDrive[]="0006B100";

  //analog velocity mode
  char analogCurMode[]="0005A101";
```

```
  //analog current scale, in units of percentage of minimum of the motor \
  //intermittent current rating and drive intermittent current rating, per 10 volts.
  char analogCurScale[]="0011910030";

  //analog current offset in units of millivolts
  char analogCurOffset[]="000491FFFB";

  //enable drive
  char enableDrive[]="0006B101";



  //forward current limit, in unit of percentage of motor interrim current or drive interim current
  char forwardCurrentLimit[]="0002F120";

  //reverse current limit, in unit of percentage of motor interrim current or drive interim current
  char reverseCurrentLimit[]="00030120";

  //set speed limit,counts/sec
  char setSpeedLimit[]="00025100100000";

  //Disable drive
  u3kSerCom(disableDrive,strlen(disableDrive));


  //Setting forward current limit
  u3kSerCom(forwardCurrentLimit,strlen(forwardCurrentLimit));

  //Setting reverse current limit
  u3kSerCom(reverseCurrentLimit,strlen(reverseCurrentLimit));

  //Setting analog current mode
  u3kSerCom(analogCurMode,strlen(analogCurMode));

  //Setting analog current scale
  u3kSerCom(analogCurScale,strlen(analogCurScale));

  //Setting analog current offset
  u3kSerCom(analogCurOffset,strlen(analogCurOffset));

  //set speed limit,counts/sec
  u3kSerCom(setSpeedLimit,strlen(setSpeedLimit));

  //Enable Drive
  u3kSerCom(enableDrive,strlen(enableDrive));

}

/** \function
        set BLDC drive mode to analog out to torque output

*/


void analog_out(void)
```

```
{

  //disable drive
  char disableDrive[]="0006B100";


  //analog current feedback output
  char analogOutCurFed[]="0004B124";

  //analog current output scale
  char analogOutCurFedScale[]="0004D10111"; //1 amps/volts

  //analog current outptu offset in units of millivolts
  char analogOutCurFedOffset[]="0004910000";

  //Disable drive
  u3kSerCom(disableDrive,strlen(disableDrive));

  //Setting analog current feedback output
  u3kSerCom(analogOutCurFed,strlen(analogOutCurFed));

  //Setting analog current output scale
  u3kSerCom(analogOutCurFedScale,strlen(analogOutCurFedScale));

  //Setting analog current outptu offset in units of millivolts
  u3kSerCom(analogOutCurFedOffset,strlen(analogOutCurFedOffset));


  /*
  //analog output---position
  char analogOutCurFed[]="0004B101";

  //analog position output scale, counts/volts
  char analogOutCurFedScale[]="0004D17FFF";

  //analog current outptu offset in units of millivolts
  char analogOutCurFedOffset[]="0004910252";

  //Disable drive
  u3kSerCom(disableDrive,strlen(disableDrive));


  //Setting analog current feedback output
  u3kSerCom(analogOutCurFed,strlen(analogOutCurFed));

  //Setting analog current output scale
  u3kSerCom(analogOutCurFedScale,strlen(analogOutCurFedScale));

  //Setting analog current outptu offset in units of millivolts
  u3kSerCom(analogOutCurFedOffset,strlen(analogOutCurFedOffset));
  */
}

/** \function
        send BLDC drive mode through RTAI serial port
```

```
*/
void *setMode(int mode)
{
            unsigned long serialTaskName = nam2num("TESTCOM");
            RT_TASK *serialTask;
            RTIME rtTime;
        struct sched_param mysched;
            mysched.sched_priority = 90;

            if( sched_setscheduler( 0, SCHED_FIFO, &mysched ) == -1 )
            {
            puts(" ERROR IN SETTING THE SCHEDULER UP");
             perror( "errno" );
             exit( 0 );
             }

            if (!(serialTask = rt_task_init(serialTaskName, 1, 0, 0))) {
                        printf("CANNOT INIT MASTER TASK\n");
                        exit(0);
            }

            rt_task_use_fpu(serialTask,1); // floating point for real time task "stdtsk"
            rt_linux_use_fpu(1); // floating point for foreground Linux processes

      //rt_make_hard_real_time();
       //printf("open serial port ... ");
       if          (rt_spopen(COM1,          38400,          8,          1,          RT_SP_PARITY_NONE,
RT_SP_NO_HAND_SHAKE,RT_SP_FIFO_SIZE_1)) {
        printf("serial port open failed!\n");
            exit(0);
       }
            //printf("Done!\n");

            rtTime=rt_get_time_ns();

            //set up analog output
            analog_out();

            //chose bldc motor mode
            switch(mode)
            {
            case 1:
            analog_velocity_mode();//velocity command mode
            break;
            case 2:
            analog_current_mode();//
            break;
            default:
            break;
            }


            rtTime=rt_get_time_ns()-rtTime;
            //printf("time spending is %f seconds\n",(float)(rtTime)/GIGA);

            rt_spclose(COM1);
```

```
        //rt_make_soft_real_time();

        rt_task_delete(serialTask);

        return;
        //exit(1); //program only run once
}
```

shiftcontrol.h

```
/* Transmission based servor actuator system control
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file shiftcontrol.h


*/
#ifndef _SHIFTCONTROL_H
#define _SHIFTCONTROL_H

#include <stdio.h>
#include <comedilib.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <getopt.h>
#include <ctype.h>
#include "examples.h"
#define WAITTIME 4000000

int setGear(void);
#endif
```

shiftcontrol.c

```
/* Transmission based servo actuator system control
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file shiftcontrol.c
        This file defines the function acssociated with send the gear control action



*/
#include <stdio.h>
#include <comedilib.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <getopt.h>
#include <ctype.h>

#define KEEP_STATIC_INLINE
#include <rtai_lxrt_user.h>
#include <rtai_lxrt.h>
#include <rtai_fifos.h>
#include <rtai_msg.h>

#include "common.h"
#include "ni6711.h"


extern int quit;

/** \function
        send a gear action command through NI-PCI6711 digital out


*/

void *setGear(void *pmbx)
{
        struct sched_param mysched;
        int to_gear;
        RT_TASK *gearshift_tsk;


        mysched.sched_priority = 98;

        if( sched_setscheduler( 0, SCHED_FIFO, &mysched ) == -1 )
```

```
{
puts(" ERROR IN SETTING THE SCHEDULER UP");
perror( "errno" );
exit( 0 );
}

if(!(gearshift_tsk = rt_task_init(nam2num("GEARSHIFT"), 1 , 0 ,0)))
{
puts("CANNOT INIT GEAR SHIFT TASK\n");
exit(0);
}

//rt_task_make_periodic(gearshift_tsk,(RTIME)(PERIOD*ticks_per_second+1.0),(RTIME)
(PERIOD*ticks_per_second));

rt_task_use_fpu(gearshift_tsk ,1);
rt_linux_use_fpu(1);

//set gear to default(1st gear )
comedi_dio_write(ni6711.ni_daq,DIO_DEVICE,DIO_1,ENGAGE);
comedi_dio_write(ni6711.ni_daq,DIO_DEVICE,DIO_2,DISENGAGE);
comedi_dio_write(ni6711.ni_daq,DIO_DEVICE,DIO_3,DISENGAGE);



while(!quit)
{
//printf("quit=%d\n",quit);
rt_mbx_receive_timed(pmbx,(void*)&to_gear,sizeof(int),(RTIME)1000000000);
//rt_mbx_receive_if(pmbx,(void*)&to_gear,sizeof(int));

//printf("gear= %d ",to_gear);

//printf("quit=%d\n",quit);
//printf("gear shiftpriority=%d\n",mysched.sched_priority);
switch(to_gear)
{

case 3:
comedi_dio_write(ni6711.ni_daq,DIO_DEVICE,DIO_3,ENGAGE);
//printf("engage 3rd gear\n");

comedi_dio_write(ni6711.ni_daq,DIO_DEVICE,DIO_2,DISENGAGE);
//printf("disengage 2nd gear\n");

comedi_dio_write(ni6711.ni_daq,DIO_DEVICE,DIO_1,DISENGAGE);
//printf("disengage 1st gear\n");
break;


case 2:
comedi_dio_write(ni6711.ni_daq,DIO_DEVICE,DIO_2,ENGAGE);
//printf("engage 2nd gear\n");

comedi_dio_write(ni6711.ni_daq,DIO_DEVICE,DIO_3,DISENGAGE);
//printf("disengage 3rd gear\n");
```

```
            comedi_dio_write(ni6711.ni_daq,DIO_DEVICE,DIO_1,DISENGAGE);
            //printf("disengage 1st gear\n");

            break;


            case 1:
            comedi_dio_write(ni6711.ni_daq,DIO_DEVICE,DIO_1,ENGAGE);
            //printf("engage 1st gear\n");

            comedi_dio_write(ni6711.ni_daq,DIO_DEVICE,DIO_2,DISENGAGE);
            //printf("disengage 2nd gear\n");

            comedi_dio_write(ni6711.ni_daq,DIO_DEVICE,DIO_3,DISENGAGE);
            //printf("disengage 3st gear\n");
            break;

            case 0:

            comedi_dio_write(ni6711.ni_daq,DIO_DEVICE,DIO_1,DISENGAGE);
            //printf("engage 1st gear\n");

            comedi_dio_write(ni6711.ni_daq,DIO_DEVICE,DIO_2,DISENGAGE);
            //printf("disengage 2nd gear\n");

            comedi_dio_write(ni6711.ni_daq,DIO_DEVICE,DIO_3,DISENGAGE);
            //printf("disengage 3st gear\n");

            default:

            break;

            }

            }

//rt_task_wait_period();

rt_mbx_delete(mbxgear);
rt_task_delete(gearshift_tsk);
return 1;
}
```

tbacontrol.c

```
/* Transmission based servor actuator system control
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file tbacontrol.c
        This is the main function of the TBA control software
        It does the following tasks by calling specific function calls:

-# Initialize servo
-# Clibrate NI-6023e
-# Initialize the braking motors
-# Startup TBA GUI interface
-# Manage all the task threads

*/

#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <getopt.h>
#include <ctype.h>
#include <signal.h>
#include <comedilib.h>
#include <pthread.h>

#define KEEP_STATIC_INLINE
#include <rtai_lxrt_user.h>
#include <rtai_lxrt.h>
#include <rtai_fifos.h>
#include <rtai_msg.h>

#include "ni6711.h"
#include "ni6023e.h"
#include "common.h"
#include "setmode.h"
#include "control.h"

int quit=0;

int main()
{

int drive_mode,controller_id,to_gear;

pid_t pid;
```

```
int status;


drive_mode=2;
controller_id=2;
to_gear=1;


if(init_hrt()==ERROR)
{
printf("hard real timer initilization failed!\n");
exit(0);
}


//initialize NI-6711

if(!(ni6711.ni_daq=init_6711()))
        {
                printf("NI6711 initializtion failed!\n");
                exit(ERROR);
                }


//initilize NI-6023e


if(!(ni6023e.ni_daq=init_6023e()))
        {
                printf("NI6023e initializtion failed!\n");
                exit(ERROR);
                }
printf("\n\n*************************************************************\n");
printf("\nwait for analog input calibaration finshed!\n");
printf("\n*************************************************************\n");
sleep(5);


//initialize rtai fifo

if (!(rtfifo = rtf_open_sized("/dev/rtf0", O_RDWR, 2000))) {
                printf("ERROR OPENING FIFO0\n");
                exit(ERROR);
          }

//start gui

pid=fork();

if (pid==-1)
{
printf("fork failed!\n");
exit(0);
}

if(pid==0)
```

```
{
execv("/home/robin/Dissertation/src/qtgui/realtime","");
_exit (EXIT_FAILURE);
}

if (pid==1)
{
printf("I am parent, waiting\n");
waitpid(pid,&status,0);
}

printf("\n\n************************************************************\n");
printf("\nWait for gui to start,click 'START' button on the gui!\n");
printf("\n************************************************************\n");
sleep(3);


//set drive mode

setMode(drive_mode);
printf("\n\n************************************************************\n");
printf("\nwait for u3k initialization ...\n");
printf("\n************************************************************\n");


printf("Done!\n");



pthread_create(&shift_thrd,NULL,setGear,(void*)mbxgear);
pthread_create(&stdout_thrd,NULL,stdOut,(void*)mbxstdout);


motorControl(controller_id,drive_mode);
quit=1;


pthread_join(shift_thrd,NULL);
pthread_join(stdout_thrd,NULL);

//clean up before termination
close(rtfifo);

return;
}
```

TBAGUI

realtime.c

```
/* Transmission based servo actuator system control - GUI
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file realtime.c
        This file defines the function acssociated with send the gear control action



*/


#include <qapp.h>
#include "mainwindow.h"

int main(int argc, char **argv)
{
    QApplication a(argc, argv);

    MainWindow w;
    w.show();
    a.setMainWidget(&w);

    return a.exec();
}
```

mainwidow.h

```
/* Transmission based servo actuator system control - GUI
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file mainwidow.h
        Class definition for the mainwindow
*/
#ifndef _MAINWINDOW_H_
#define _MAINWINDOW_H_ 1
#include <qapp.h>
#include <qmainwindow.h>
#include <qtoolbutton.h>
#include <qcombobox.h>

class QSpinBox;
class QPushButton;
class RandomPlot;

/**
A mainwindow layout
*/
class MainWindow: public QMainWindow
{
   Q_OBJECT
public:

   MainWindow();

private slots:
   void showRunning(bool);
   void appendPoints(bool);

private:
   QToolBar *toolBar();
   void initWhatsThis();

private:
   QSpinBox *d_randomCount;
   QSpinBox *d_timerCount;
   QToolButton *d_startBtn;
   QToolButton *d_clearBtn;
   QString str;
   QComboBox *cb;
   RandomPlot *d_plot;
};

#endif
```

mainwindow.cpp

```
/* Transmission based servo actuator system control - GUI
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file mainwindow.cpp
        This file is the main window layout
*/

#include <qlabel.h>
#include <qhbox.h>
#include <qstatusbar.h>
#include <qtoolbar.h>
#include <qtoolbutton.h>
#include <qspinbox.h>
#include <qwhatsthis.h>
#include <qcombobox.h>
#include "randomplot.h"
#include "mainwindow.h"
#include "start.xpm"
#include "clear.xpm"

class MyToolBar: public QToolBar
{
public:
    MyToolBar(MainWindow *);
    void addSpacing(int);
    void addStretch();
};

MyToolBar::MyToolBar(MainWindow *parent):
    QToolBar(parent)
{
}

void MyToolBar::addSpacing(int spacing)
{
    QLabel *label = new QLabel(this);
    label->setFixedWidth(spacing);
}

void MyToolBar::addStretch()
{
    QLabel *label = new QLabel(this);
    setStretchableWidget(label);
}

MainWindow::MainWindow()
```

```
{
    setDockEnabled(DockTornOff, TRUE);
    setRightJustification(TRUE);

    (void)toolBar();
    (void)statusBar();

    d_plot = new RandomPlot(this);

    d_plot->setMargin(4);

    setCentralWidget(d_plot);

    connect(d_startBtn, SIGNAL(toggled(bool)), this, SLOT(appendPoints(bool)));
    connect(d_clearBtn, SIGNAL(clicked()), d_plot, SLOT(clear()));
    connect(d_plot, SIGNAL(running(bool)), this, SLOT(showRunning(bool)));
    connect(cb,SIGNAL(activated(int)),d_plot,SLOT(toggleCurve(int)));

    initWhatsThis();
}

QToolBar *MainWindow::toolBar()
{
    MyToolBar *toolBar = new MyToolBar(this);

    d_startBtn = new QToolButton(toolBar);
    d_startBtn->setUsesTextLabel(TRUE);
    d_startBtn->setPixmap(QPixmap(start_xpm));
    d_startBtn->setToggleButton(TRUE);

    d_clearBtn = new QToolButton(toolBar);
    d_clearBtn->setUsesTextLabel(TRUE);
    d_clearBtn->setPixmap(QPixmap(clear_xpm));
    d_clearBtn->setTextLabel("Clear", FALSE);

    QToolButton *helpBtn = QWhatsThis::whatsThisButton(toolBar);
    helpBtn->setUsesTextLabel(TRUE);
    helpBtn->setTextLabel("Help", FALSE);


    // Create a non-editable Combobox and a label below...
    toolBar->addSpacing(20);
    toolBar->addStretch();

    cb = new QComboBox( FALSE, toolBar);

    //...and insert 5 items into the Combobox
    str = QString( "Motor Speed" );
    cb->insertItem( str );
    str = QString( "DVT Output Speed" );
    cb->insertItem( str );
    str = QString( "Motor Torque" );
    cb->insertItem( str );
    str = QString( "Dynamometer Torque" );
    cb->insertItem( str );
    str = QString( "DVT Position" );
```

```
    cb->insertItem( str );
    str = QString( "COM Position" );
    cb->insertItem( str );

    toolBar->addSpacing(10);

    showRunning(FALSE);

    d_startBtn->setMinimumWidth(helpBtn->sizeHint().width() + 20);
    d_clearBtn->setMinimumWidth(helpBtn->sizeHint().width() + 20);
    helpBtn->setMinimumWidth(helpBtn->sizeHint().width() + 20);

    return toolBar;
}

void MainWindow::appendPoints(bool on)
{
    if ( on )
        d_plot->append();
    else
        d_plot->stop();
}

void MainWindow::showRunning(bool running)
{

    d_startBtn->setOn(running);
    d_startBtn->setTextLabel(running ? "Stop" : "Start", FALSE);
}

void MainWindow::initWhatsThis()
{
    QWhatsThis::add(d_plot,
        "Zooming is enabled until the selected area gets "
        "too small for the significance on the axes.\n\n"
        "You can zoom in using the left mouse button.\n"
        "The middle mouse button is used to go back to the "
        "previous zoomed area.\n"
        "The right mouse button is used to unzoom completely."
    );

    QWhatsThis::add(d_startBtn,
        "Start generation of random points.\n\n"
        "The intention of this example is to show how to implement "
        "growing curves. The points will be generated and displayed "
        "one after the other.\n"
        "To check the performance, a small delay and a large number "
        "of points are useful. To watch the curve growing, a delay "
        " > 300 ms and less points are better.\n"
        "To inspect the curve, stacked zooming is implemented using the "
        "mouse buttons on the plot."
    );
    QWhatsThis::add(d_clearBtn,
        "Remove all points."
    );
}
```

randomplot.h

```
/* Transmission based servo actuator system control - GUI
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file randomplot.h
        class definition of the data curve plot
*/
#ifndef _RANDOMPLOT_H_
#define _RANDOMPLOT_H_ 1
#include "incrementalplot.h"
#define UPDATERATE 10; //ms

class QTimer;

/**
A plot area to hold all data curves
*/
class RandomPlot: public IncrementalPlot
{
   Q_OBJECT

public:

   RandomPlot(QWidget *parent);

   virtual QSize sizeHint() const;

   enum tbaData
   {
      MOTORSPEED,
      DVTSPEED,
      MOTORTORQUE,
      NYNATORQUE,
      DVTPOSITION,
      COMPOSITION,
      NTbaData
   };

   int cmbIndex;

signals:
   void running(bool);
   void timeout();

public slots:
   void clear();
   void stop();
```

```
    void append();
    void toggleCurve(int);

private slots:
    void appendPoint();


private:

    void initCurve();

    long d_curveId;

    QTimer *d_timer;
    int d_timerCount;
    int fd;
    int len;

    /*
    struct datapack{
    double time;
    int dvt_position;
    int motor_position;
    int gear_current;
    int command_position;
    double motor_speed;
    double dvt_speed;
    double load;
    //double motor_torque;
    };
    */
    struct datapack{
    float time;
    int dvt_position;
    int motor_position;
    int gear_current;
    int command_position;
    float motor_speed;
    float dvt_speed;
    float load;
    float motor_torque;
    };
     typedef struct datapack Datapack;

    Datapack data_pack;
    double x,y;

    struct
    {
        long curve;
        double data[100000];
    } data[NTbaData];

    };

#endif // _RANDOMPLOT_H_
```

mainwindow.cpp

```
/* Transmission based servo actuator system control - GUI
 *
 * Renbin Zhou <zhourb@gmail.com>
 *
 * This file may be freely modified, distributed, and combined with
 * other software, as long as proper attribution is given in the
 * source code.
 */

/** \file mainwindow.cpp
        This file is the data curve plot
*/

#include <stdlib.h>
#include <qtimer.h>
#include <fcntl.h> //open()
#include <unistd.h> //close()
#include "scrollzoomer.h"
#include "randomplot.h"

const unsigned int c_rangeMax = 10;

RandomPlot::RandomPlot(QWidget *parent):
    IncrementalPlot(parent),
    d_curveId(0),
    d_timer(0),
    d_timerCount(0)
{
    setFrameStyle(NoFrame);
    setLineWidth(0);
    setCanvasLineWidth(2);

    enableGridX(TRUE);
    enableGridY(TRUE);
    setGridMajPen(QPen(gray, 0, DotLine));

    //setCanvasBackground(QColor(29, 100, 141)); // nice blue
    setCanvasBackground(QColor(255, 255, 255));

    setAxisScale(xBottom, 0, 100);
    setAxisScale(yLeft, -50, 300);


    for ( int i = 0; i < QwtPlot::axisCnt; i++ )
        setAxisLabelFormat(i, 'g', 8);

    // enable zooming

    ScrollZoomer *zoomer = new ScrollZoomer(canvas());
    zoomer->setRubberBandPen(QPen(Qt::red, 0, Qt::DotLine));
    zoomer->setCursorLabelPen(QPen(Qt::yellow));
```

```
    initCurve();
    replot();


    fd=open("/dev/rtf0",O_RDWR,O_NDELAY);

    //printf("fd=%d\n",fd);

    if (fd==-1) {
    printf("fifo open failed!");
    exit(1);
    }

}

QSize RandomPlot::sizeHint() const
{
    return QSize(600,450);
}

void RandomPlot::initCurve()
{
    /*
    if ( d_curveId > 0 )
    {
        removeCurveData(d_curveId);
        removeCurve(d_curveId);
    }

    d_curveId = insertCurve("Motor speed curve");

    setCurveStyle(d_curveId, QwtCurve::Lines);
    setTitle("Realtime TBA control");

    //long line_marker=insertLineMarker("",QwtPlot::yLeft);
    //setMarkerYPos(line_marker,0.0);

    //const QColor &c=black;

//setCurveSymbol(d_curveId,QwtSymbol(QwtSymbol::XCross,QBrush(c),QPen(c),QSize(2,2)));

    setAxisTitle(xBottom, "Time (second)");
    setAxisTitle(yLeft, "Speed (radius/second)");
    setAxisAutoScale(xBottom);
    //replot();
    */

    //plotLayout()->setAlignCanvasToScales(TRUE);
    //setCanvasBackground(Qt::darkGray);

    //setAutoLegend(TRUE);
    //setLegendPos(Qwt::Right);

    setAutoLegend(TRUE);
    setLegendPos(Qwt::Bottom);
```

```
    setTitle("Realtime TBA control");

    setAxisTitle(QwtPlot::xBottom, " Time (seconds)");
    setAxisScale(QwtPlot::xBottom, 0, 100);


    setAxisScale(QwtPlot::yLeft, -50, 300);


    data[MOTORSPEED].curve = insertCurve("Motor Speed");
    setCurvePen(data[MOTORSPEED].curve, QPen(black));
    setCurveStyle(data[MOTORSPEED].curve, QwtCurve::Lines);


    data[DVTSPEED].curve = insertCurve("DVT Speed");
    setCurvePen(data[DVTSPEED].curve, QPen(green));
    setCurveStyle(data[DVTSPEED].curve, QwtCurve::Lines);

    data[MOTORTORQUE].curve = insertCurve("Motor Torque");
    setCurvePen(data[MOTORTORQUE].curve, QPen(blue));
    setCurveStyle(data[DVTSPEED].curve, QwtCurve::Lines);

    data[NYNATORQUE].curve = insertCurve("Dyna Torque");
    setCurvePen(data[NYNATORQUE].curve, QPen(red));
    setCurveStyle(data[NYNATORQUE].curve, QwtCurve::Lines);


    data[DVTPOSITION].curve = insertCurve("DVT Position");
    setCurvePen(data[DVTPOSITION].curve, QPen(gray));
    setCurveStyle(data[DVTPOSITION].curve, QwtCurve::Lines);

    data[COMPOSITION].curve = insertCurve("Command Position");
    setCurvePen(data[COMPOSITION].curve, QPen(black));
    setCurveStyle(data[COMPOSITION].curve, QwtCurve::Lines);


    //toggleCurve(1);


    replot();

}

void RandomPlot::appendPoint()
{
   /*
    x = rand() % c_rangeMax;
    x += ( rand() % 100 ) / 100;

    printf("x=%.2f ",x );


    y = rand() % c_rangeMax;
    y += ( rand() % 100 ) / 100;
    printf("y=%.2f \n",y );
   */
```

```
   len=read(fd,(void*)&data_pack,sizeof(data_pack));

   if (len == 0) {
      printf("buffer empty\n");
            return;
   }

  else if(len==sizeof(data_pack))

  {

   x=data_pack.time;

   y=data_pack.load;

   //printf("time= %.4f\n",x);

  }

  //printf("cmbIndex = %d",cmbIndex);

  //switch (cmbIndex)
  //{
  //case 0:
   appendCurvePoint(data[MOTORSPEED].curve, x, data_pack.motor_speed);
 //  break;
 // case 1:
   appendCurvePoint(data[DVTSPEED].curve, x, data_pack.dvt_speed);
 //  break;
 // case 2:
   appendCurvePoint(data[MOTORTORQUE].curve, x, data_pack.motor_torque);
 //  break;
 // case 3:
   appendCurvePoint(data[NYNATORQUE].curve, x, data_pack.load);
//  break;
 // case 4:
   appendCurvePoint(data[DVTPOSITION].curve,                                    x,
(double)data_pack.dvt_position/1024./*2.*3.1415926*/);
 //  break;
//  default:
//  break;
   appendCurvePoint(data[COMPOSITION].curve,                                    x,
(double)data_pack.command_position/1024.);
 // }
     //    stop();
}

void RandomPlot::append()
{

  if ( !d_timer )
    {
       d_timer = new QTimer(this);
       connect(d_timer, SIGNAL(timeout()), SLOT(appendPoint()));
    }
```

```
    emit running(TRUE);

    d_timer->start(10); //qt timer update rate ms
      }

void RandomPlot::stop()
{
    if ( d_timer )
    {
        d_timer->stop();
        emit running(FALSE);
    }
}

void RandomPlot::clear()
{
    removeCurves();
    initCurve();
    //removeCurves();
    //replot();
}

void RandomPlot::toggleCurve(int cmbBoxId)
{
    cmbIndex=cmbBoxId;

    //printf("cmbbox ID=%d \n",cmbBoxId);

    QwtPlotCurve *c1=curve(data[MOTORSPEED].curve);
    QwtPlotCurve *c2=curve(data[DVTSPEED].curve);
    QwtPlotCurve *c3=curve(data[MOTORTORQUE].curve);
    QwtPlotCurve *c4=curve(data[NYNATORQUE].curve);
    QwtPlotCurve *c5=curve(data[DVTPOSITION].curve);
    QwtPlotCurve *c6=curve(data[COMPOSITION].curve);


    switch (cmbBoxId)
    {
    case 0:

    if (c1)
    {
    c1->setEnabled(TRUE);
    }

    if (c2)
    {
    c2->setEnabled(FALSE);
    }

    if (c3)
    {
    c3->setEnabled(FALSE);
    }
```

```
if (c4)
{
c4->setEnabled(FALSE);
}

if (c5)
{
c5->setEnabled(FALSE);
}

if (c6)
{
c6->setEnabled(FALSE);
}

replot();
break;

case 1:

if (c1)
{
c1->setEnabled(FALSE);
}
if (c2)
{
c2->setEnabled(TRUE);
}
if (c3)
{
c3->setEnabled(FALSE);
}
if (c4)
{
c4->setEnabled(FALSE);
}
if (c5)
{
c5->setEnabled(FALSE);
}
if (c6)
{
c6->setEnabled(FALSE);
}

replot();
break;

case 2:

if (c1)
{
c1->setEnabled(FALSE);
}
if (c2)
{
```

```
c2->setEnabled(FALSE);
}
if (c3)
{
c3->setEnabled(TRUE);
}
if (c4)
{
c4->setEnabled(FALSE);
}
if (c5)
{
c5->setEnabled(FALSE);
}
if (c6)
{
c6->setEnabled(FALSE);
}

replot();
break;

  case 3:

if (c1)
{
c1->setEnabled(FALSE);
}
if (c2)
{
c2->setEnabled(FALSE);
}
if (c3)
{
c3->setEnabled(FALSE);
}
if (c4)
{
c4->setEnabled(TRUE);
}
if (c5)
{
c5->setEnabled(FALSE);
}
if (c6)
{
c6->setEnabled(FALSE);
}

replot();
break;

  case 4:

if (c1)
{
```

```
c1->setEnabled(FALSE);
}
if (c2)
{
c2->setEnabled(FALSE);
}
if (c3)
{
c3->setEnabled(FALSE);
}
if (c4)
{
c4->setEnabled(FALSE);
}
if (c5)
{
c5->setEnabled(TRUE);
}
if (c6)
{
c6->setEnabled(FALSE);
}
replot();
break;

  case 5:

if (c1)
{
c1->setEnabled(FALSE);
}
if (c2)
{
c2->setEnabled(FALSE);
}
if (c3)
{
c3->setEnabled(FALSE);
}
if (c4)
{
c4->setEnabled(FALSE);
}
if (c5)
{
c5->setEnabled(FALSE);
}
if (c6)
{
c6->setEnabled(TRUE);
}
replot();
break;

default:
replot();
```

```
    break;
    }
}

#ifndef _INCREMENTALPLOT_H_
#define _INCREMENTALPLOT_H_ 1

#include <qintdict.h>
#include <qwt_array.h>
#include <qwt_plot.h>

class CurveData
{
    // A container class for growing data
public:

    CurveData();

    void append(double *x, double *y, int count);

    int count() const;
    int size() const;
    double *x() const;
    double *y() const;

private:
    int d_count;
    QwtArray<double> d_x;
    QwtArray<double> d_y;
};

class IncrementalPlot : public QwtPlot
{
    Q_OBJECT
public:
    IncrementalPlot(QWidget *parent = 0, const char *name = 0);
    virtual ~IncrementalPlot();

    void appendCurvePoint(long curveId, double x, double y);
    void appendCurveData(long curveId,
        double *x, double *y, int size);

    void removeCurveData(long curveId);
private:

    QIntDict<CurveData> d_curveDictionary;
};

#endif // _INCREMENTALPLOT_H_
```

incrementalplot.cpp

```
/** incrementalplot.cpp
*/

#include <qwt_plot.h>
#include <qwt_plot_dict.h>
#include "incrementalplot.h"

CurveData::CurveData():
    d_count(0)
{
}

void CurveData::append(double *x, double *y, int count)
{
    int newSize = ( (d_count + count) / 1000 + 1 ) * 1000;

    if ( newSize > size() )
    {
        d_x.resize(newSize);
        d_y.resize(newSize);
    }

    for ( register int i = 0; i < count; i++ )
    {
        d_x[d_count + i] = x[i];
        d_y[d_count + i] = y[i];
    }
    d_count += count;
}

int CurveData::count() const
{
    return d_count;
}

int CurveData::size() const
{
    return d_x.size();
}

double *CurveData::x() const
{
    return d_x.data();
}

double *CurveData::y() const
{
    return d_y.data();
}

IncrementalPlot::IncrementalPlot(QWidget *parent, const char* name):
    QwtPlot(parent, name)
```

```
{
   d_curveDictionary.setAutoDelete(TRUE);
}

IncrementalPlot::~IncrementalPlot()
{
   removeCurves();
}

void IncrementalPlot::appendCurvePoint(long curveId, double x, double y)
{
   appendCurveData(curveId, &x, &y, 1);
}

void IncrementalPlot::appendCurveData(long curveId,
   double *x, double *y, int size)
{
   QwtPlotCurve *curve = IncrementalPlot::curve(curveId);
   if ( curve == 0 || size <= 0 )
      return;

   CurveData *curveData = d_curveDictionary.find(curveId);
   if ( curveData == 0 )
   {
      curveData = new CurveData();
      d_curveDictionary.insert(curveId, curveData);
   }

   curveData->append(x, y, size);
   curve->setRawData(curveData->x(), curveData->y(), curveData->count());

   drawCurve(curveId, curve->dataSize() - size, curve->dataSize() - 2);
   setAutoLegend(TRUE);
   //legendEnabled(data[DVTPOSITION].curve);
}

void IncrementalPlot::removeCurveData(long curveId)
{
   d_curveDictionary.remove(curveId);
}
```

scrollbar.h

```
/** scrollbar.h
*/

#ifndef _SCROLLBAR_H
#define _SCROLLBAR_H 1

#include <qscrollbar.h>

class ScrollBar: public QScrollBar
{
   Q_OBJECT

public:
   ScrollBar(QWidget *parent = NULL, const char *name = NULL);
   ScrollBar(Qt::Orientation,
      QWidget *parent = NULL, const char *name = NULL);
   ScrollBar(double minBase, double maxBase,
      Orientation o, QWidget *parent = NULL, const char *name = NULL);
   void setInverted(bool);
   bool isInverted() const;
   double minBaseValue() const;
   double maxBaseValue() const;
   double minSliderValue() const;
   double maxSliderValue() const;

   int extent() const;

signals:
   void sliderMoved(Qt::Orientation, double, double);
   void valueChanged(Qt::Orientation, double, double);

public slots:
   virtual void setBase(double min, double max);
   virtual void moveSlider(double min, double max);

protected:
   void sliderRange(int value, double &min, double &max) const;
   int mapToTick(double) const;
   double mapFromTick(int) const;
private slots:
   void catchValueChanged(int value);
   void catchSliderMoved(int value);
private:
   void init();

   bool d_inverted;
   double d_minBase;
   double d_maxBase;
   int d_baseTicks;
};

#endif
```

scrollbar.cpp

```
/** scrollbar.cpp
*/
#include <qstyle.h>
#include "scrollbar.h"

ScrollBar::ScrollBar(QWidget * parent, const char *name):
   QScrollBar(parent, name)
{
   init();
}

ScrollBar::ScrollBar(Qt::Orientation o,
       QWidget *parent, const char *name):
   QScrollBar(o, parent, name)
{
   init();
}

ScrollBar::ScrollBar(double minBase, double maxBase,
       Orientation o, QWidget *parent, const char *name):
   QScrollBar(o, parent, name)
{
   init();
   setBase(minBase, maxBase);
   moveSlider(minBase, maxBase);
}

void ScrollBar::init()
{
   d_inverted = orientation() == Qt::Vertical;
   d_baseTicks = 1000000;
   d_minBase = 0.0;
   d_maxBase = 1.0;
   moveSlider(d_minBase, d_maxBase);

   connect(this, SIGNAL(sliderMoved(int)), SLOT(catchSliderMoved(int)));
   connect(this, SIGNAL(valueChanged(int)), SLOT(catchValueChanged(int)));
}

void ScrollBar::setInverted(bool inverted)
{
   if ( d_inverted != inverted )
   {
      d_inverted = inverted;
      moveSlider(minSliderValue(), maxSliderValue());
   }
}

bool ScrollBar::isInverted() const
{
   return d_inverted;
}
```

```
void ScrollBar::setBase(double min, double max)
{
   if ( min != d_minBase || max != d_maxBase )
   {
      d_minBase = min;
      d_maxBase = max;

      moveSlider(minSliderValue(), maxSliderValue());
   }
}

void ScrollBar::moveSlider(double min, double max)
{
   const int sliderTicks = qRound((max - min) /
      (d_maxBase - d_minBase) * d_baseTicks);

   // setRange initiates a valueChanged of the scrollbars
   // in some situations. So we block
   // and unblock the signals.

   blockSignals(TRUE);

   setRange(sliderTicks / 2, d_baseTicks - sliderTicks / 2);
   int steps = sliderTicks / 200;
   if ( steps <= 0 )
      steps = 1;

   // setPageStep, setLineStep ???

   setSteps(steps, sliderTicks);

   int tick = mapToTick(min + (max - min) / 2);
   if ( isInverted() )
      tick = d_baseTicks - tick;

   directSetValue(tick);
   blockSignals(FALSE);

   rangeChange();
}

double ScrollBar::minBaseValue() const
{
   return d_minBase;
}

double ScrollBar::maxBaseValue() const
{
   return d_maxBase;
}

void ScrollBar::sliderRange(int value, double &min, double &max) const
{
   if ( isInverted() )
      value = d_baseTicks - value;
   const int visibleTicks = pageStep();
```

```
    min = mapFromTick(value - visibleTicks / 2);
    max = mapFromTick(value + visibleTicks / 2);
}

double ScrollBar::minSliderValue() const
{
    double min, dummy;
    sliderRange(value(), min, dummy);

    return min;
}

double ScrollBar::maxSliderValue() const
{
    double max, dummy;
    sliderRange(value(), dummy, max);

    return max;
}

int ScrollBar::mapToTick(double v) const
{
    return (int) ( ( v - d_minBase) / (d_maxBase - d_minBase ) * d_baseTicks );
}

double ScrollBar::mapFromTick(int tick) const
{
    return d_minBase + ( d_maxBase - d_minBase ) * tick / d_baseTicks;
}

void ScrollBar::catchValueChanged(int value)
{
    double min, max;
    sliderRange(value, min, max);
    emit valueChanged(orientation(), min, max);
}

void ScrollBar::catchSliderMoved(int value)
{
    double min, max;
    sliderRange(value, min, max);
    emit sliderMoved(orientation(), min, max);
}

int ScrollBar::extent() const
{
    int dim;
#if QT_VERSION >= 300
    dim = style().pixelMetric(QStyle::PM_ScrollBarExtent, this);
#else
    const QSize sz = style().scrollBarExtent();
    dim = (orientation() == Qt::Horizontal) ? sz.height() : sz.width();
#endif
    return dim;
}
```

scrollzoomer.h

```
/**scrollzoomer.h
*/

#ifndef _SCROLLZOOMER_H
#define _SCROLLZOOMER_H

#include <qscrollview.h>
#include <qwt_plot_zoomer.h>

class ScrollData;
class ScrollBar;

class ScrollZoomer: public QwtPlotZoomer
{
   Q_OBJECT
public:
   enum ScrollBarPosition
   {
      AttachedToScale,
      OppositeToScale
   };

   ScrollZoomer(QwtPlotCanvas *, const char *name = 0);
   virtual ~ScrollZoomer();

   ScrollBar *horizontalScrollBar() const;
   ScrollBar *verticalScrollBar() const;

   void setHScrollBarMode(QScrollView::ScrollBarMode);
   void setVScrollBarMode(QScrollView::ScrollBarMode);

   QScrollView::ScrollBarMode vScrollBarMode () const;
   QScrollView::ScrollBarMode hScrollBarMode () const;

   void setHScrollBarPosition(ScrollBarPosition);
   void setVScrollBarPosition(ScrollBarPosition);

   ScrollBarPosition hScrollBarPosition() const;
   ScrollBarPosition vScrollBarPosition() const;

   QWidget* cornerWidget() const;
   virtual void setCornerWidget(QWidget *);

   virtual bool eventFilter(QObject *, QEvent *);

   virtual void rescale();

protected:
   virtual ScrollBar *scrollBar(Qt::Orientation);
   virtual void updateScrollBars();
   virtual void layoutScrollBars(const QRect &);
```

```
private slots:
    void scrollBarMoved(Qt::Orientation o, double min, double max);

private:
    bool needScrollBar(Qt::Orientation) const;
    int oppositeAxis(int) const;

    QWidget *d_cornerWidget;

    ScrollData *d_hScrollData;
    ScrollData *d_vScrollData;
};

#endif
```

scrollzoomer.cpp

```
/**scrollzoomer.cpp
*/

#include "qwt_plot_canvas.h"
#include "qwt_plot_layout.h"
#include "scrollbar.h"
#include "scrollzoomer.h"

class ScrollData
{
public:
    ScrollData():
        scrollBar(NULL),
        position(ScrollZoomer::OppositeToScale),
        mode(QScrollView::Auto)
    {
    }

    ~ScrollData()
    {
        delete scrollBar;
    }

    ScrollBar *scrollBar;
    ScrollZoomer::ScrollBarPosition position;
    QScrollView::ScrollBarMode mode;
};

ScrollZoomer::ScrollZoomer(QwtPlotCanvas *canvas, const char *name):
    QwtPlotZoomer(canvas, name),
    d_cornerWidget(NULL),
    d_hScrollData(NULL),
    d_vScrollData(NULL)
{
    if ( !canvas )
        return;

    d_hScrollData = new ScrollData;
    d_vScrollData = new ScrollData;
}

ScrollZoomer::~ScrollZoomer()
{
    delete d_cornerWidget;
    delete d_vScrollData;
    delete d_hScrollData;
}

void ScrollZoomer::rescale()
{
    QwtPlotZoomer::rescale();
    updateScrollBars();
```

```
}

ScrollBar *ScrollZoomer::scrollBar(Qt::Orientation o)
{
    ScrollBar *&sb = (o == Qt::Vertical)
        ? d_vScrollData->scrollBar : d_hScrollData->scrollBar;

    if ( sb == NULL )
    {
        sb = new ScrollBar(o, canvas());
        sb->hide();
        connect(sb,
            SIGNAL(valueChanged(Qt::Orientation, double, double)),
            SLOT(scrollBarMoved(Qt::Orientation, double, double)));
    }
    return sb;
}

ScrollBar *ScrollZoomer::horizontalScrollBar() const
{
    return d_hScrollData->scrollBar;
}

ScrollBar *ScrollZoomer::verticalScrollBar() const
{
    return d_vScrollData->scrollBar;
}

void ScrollZoomer::setHScrollBarMode(QScrollView::ScrollBarMode mode)
{
    if ( hScrollBarMode() != mode )
    {
        d_hScrollData->mode = mode;
        updateScrollBars();
    }
}

void ScrollZoomer::setVScrollBarMode(QScrollView::ScrollBarMode mode)
{
    if ( vScrollBarMode() != mode )
    {
        d_vScrollData->mode = mode;
        updateScrollBars();
    }
}

QScrollView::ScrollBarMode ScrollZoomer::hScrollBarMode() const
{
    return d_hScrollData->mode;
}

QScrollView::ScrollBarMode ScrollZoomer::vScrollBarMode () const
{
    return d_vScrollData->mode;
}
```

```
void ScrollZoomer::setHScrollBarPosition(ScrollBarPosition pos)
{
   if ( d_hScrollData->position != pos )
   {
      d_hScrollData->position = pos;
      updateScrollBars();
   }
}

void ScrollZoomer::setVScrollBarPosition(ScrollBarPosition pos)
{
   if ( d_vScrollData->position != pos )
   {
      d_vScrollData->position = pos;
      updateScrollBars();
   }
}

ScrollZoomer::ScrollBarPosition ScrollZoomer::hScrollBarPosition() const
{
   return d_hScrollData->position;
}

ScrollZoomer::ScrollBarPosition ScrollZoomer::vScrollBarPosition() const
{
   return d_vScrollData->position;
}

void ScrollZoomer::setCornerWidget(QWidget *w)
{
   if ( w != d_cornerWidget )
   {
      if ( canvas() )
      {
         delete d_cornerWidget;
         d_cornerWidget = w;
         if ( d_cornerWidget->parent() != canvas() )
            d_cornerWidget->reparent(canvas(), QPoint(0, 0));

         updateScrollBars();
      }
   }
}

QWidget *ScrollZoomer::cornerWidget() const
{
   return d_cornerWidget;
}

bool ScrollZoomer::eventFilter(QObject *o, QEvent *e)
{
   if (  o == canvas() )
   {
      switch(e->type())
      {
         case QEvent::Resize:
```

```
            {
                const int fw = ((QwtPlotCanvas *)canvas())->frameWidth();

                QRect rect;
                rect.setSize(((QResizeEvent *)e)->size());
                rect.setRect(rect.x() + fw, rect.y() + fw,
                    rect.width() - 2 * fw, rect.height() - 2 * fw);

                layoutScrollBars(rect);
                break;
            }
            case QEvent::ChildRemoved:
            {
                const QObject *child = ((QChildEvent *)e)->child();
                if ( child == d_cornerWidget )
                    d_cornerWidget = NULL;
                else if ( child == d_hScrollData->scrollBar )
                    d_hScrollData->scrollBar = NULL;
                else if ( child == d_vScrollData->scrollBar )
                    d_vScrollData->scrollBar = NULL;
                break;
            }
            default:
                break;
        }
    }
    return QwtPlotZoomer::eventFilter(o, e);
}

bool ScrollZoomer::needScrollBar(Qt::Orientation o) const
{
    QScrollView::ScrollBarMode mode;
    double zoomMin, zoomMax, baseMin, baseMax;

    if ( o == Qt::Horizontal )
    {
        mode = d_hScrollData->mode;
        baseMin = zoomBase().x1();
        baseMax = zoomBase().x2();
        zoomMin = zoomRect().x1();
        zoomMax = zoomRect().x2();
    }
    else
    {
        mode = d_vScrollData->mode;
        baseMin = zoomBase().y1();
        baseMax = zoomBase().y2();
        zoomMin = zoomRect().y1();
        zoomMax = zoomRect().y2();
    }

    bool needed = FALSE;
    switch(mode)
    {
        case QScrollView::AlwaysOn:
            needed = TRUE;
```

```
            break;
        case QScrollView::AlwaysOff:
            needed = FALSE;
            break;
        case QScrollView::Auto:
        default:
        {
            if ( baseMin < zoomMin || baseMax > zoomMax )
                needed = TRUE;
            break;
        }
    }
    return needed;
}

void ScrollZoomer::updateScrollBars()
{
    if ( !canvas() )
        return;

    const int xAxis = QwtPlotZoomer::xAxis();
    const int yAxis = QwtPlotZoomer::yAxis();

    int xScrollBarAxis = xAxis;
    if ( hScrollBarPosition() == OppositeToScale )
        xScrollBarAxis = oppositeAxis(xScrollBarAxis);

    int yScrollBarAxis = yAxis;
    if ( vScrollBarPosition() == OppositeToScale )
        yScrollBarAxis = oppositeAxis(yScrollBarAxis);


    QwtPlotLayout *layout = plot()->plotLayout();

    bool showHScrollBar = needScrollBar(Qt::Horizontal);
    if ( showHScrollBar )
    {
        ScrollBar *sb = scrollBar(Qt::Horizontal);

        sb->setPalette(plot()->palette());
        sb->setInverted(plot()->axisOptions(xAxis) & QwtAutoScale::Inverted);
        sb->setBase(zoomBase().x1(), zoomBase().x2());
        sb->moveSlider(zoomRect().x1(), zoomRect().x2());

        if ( !sb->isVisibleTo(canvas()) )
        {
            sb->show();
            layout->setCanvasMargin(layout->canvasMargin(xScrollBarAxis)
                + sb->extent(), xScrollBarAxis);
        }
    }
    else
    {
        if ( horizontalScrollBar() )
        {
            horizontalScrollBar()->hide();
```

```
            layout->setCanvasMargin(layout->canvasMargin(xScrollBarAxis)
                - horizontalScrollBar()->extent(), xScrollBarAxis);
        }
    }

    bool showVScrollBar = needScrollBar(Qt::Vertical);
    if ( showVScrollBar )
    {
        ScrollBar *sb = scrollBar(Qt::Vertical);

        sb->setPalette(plot()->palette());
        sb->setInverted(!(plot()->axisOptions(yAxis) & QwtAutoScale::Inverted));
        sb->setBase(zoomBase().y1(), zoomBase().y2());
        sb->moveSlider(zoomRect().y1(), zoomRect().y2());

        if ( !sb->isVisibleTo(canvas()) )
        {
            sb->show();
            layout->setCanvasMargin(layout->canvasMargin(yScrollBarAxis)
                + sb->extent(), yScrollBarAxis);
        }
    }
    else
    {
        if ( verticalScrollBar() )
        {
            verticalScrollBar()->hide();
            layout->setCanvasMargin(layout->canvasMargin(yScrollBarAxis)
                - verticalScrollBar()->extent(), yScrollBarAxis);
        }
    }

    if ( showHScrollBar && showVScrollBar )
    {
        if ( d_cornerWidget == NULL )
        {
            d_cornerWidget = new QWidget(canvas());
            d_cornerWidget->setPalette(plot()->palette());
        }
        d_cornerWidget->show();
    }
    else
    {
        if ( d_cornerWidget )
            d_cornerWidget->hide();
    }

    layoutScrollBars(((QwtPlotCanvas *)canvas())->contentsRect());
}

void ScrollZoomer::layoutScrollBars(const QRect &rect)
{
    int hPos = xAxis();
    if ( hScrollBarPosition() == OppositeToScale )
        hPos = oppositeAxis(hPos);
```

```
int vPos = yAxis();
if ( vScrollBarPosition() == OppositeToScale )
    vPos = oppositeAxis(vPos);

ScrollBar *hScrollBar = horizontalScrollBar();
ScrollBar *vScrollBar = verticalScrollBar();

const int hdim = hScrollBar ? hScrollBar->extent() : 0;
const int vdim = vScrollBar ? vScrollBar->extent() : 0;

if ( hScrollBar && hScrollBar->isVisible() )
{
    int x = rect.x();
    int y = (hPos == QwtPlot::xTop)
        ? rect.top() : rect.bottom() - hdim + 1;
    int w = rect.width();

    if ( vScrollBar && vScrollBar->isVisible() )
    {
        if ( vPos == QwtPlot::yLeft )
            x += vdim;
        w -= vdim + 1;
    }

    hScrollBar->setGeometry(x, y, w, hdim);
}
if ( vScrollBar && vScrollBar->isVisible() )
{
    int pos = yAxis();
    if ( vScrollBarPosition() == OppositeToScale )
        pos = oppositeAxis(pos);

    int x = (vPos == QwtPlot::yLeft)
        ? rect.left() : rect.right() - vdim;
    int y = rect.y();

    int h = rect.height();

    if ( hScrollBar && hScrollBar->isVisible() )
    {
        if ( hPos == QwtPlot::xTop )
            y += hdim;

        h -= hdim;
    }

    vScrollBar->setGeometry(x, y, vdim, h);
}
if ( hScrollBar && hScrollBar->isVisible() &&
    vScrollBar && vScrollBar->isVisible() )
{
    if ( d_cornerWidget )
    {
        QRect cornerRect(
            vScrollBar->pos().x(), hScrollBar->pos().y(),
            vdim, hdim);
```

```
            d_cornerWidget->setGeometry(cornerRect);
        }
    }
}

void ScrollZoomer::scrollBarMoved(Qt::Orientation o, double min, double)
{
    if ( o == Qt::Horizontal )
        move(min, zoomRect().y1());
    else
        move(zoomRect().x1(), min);

    emit zoomed(zoomRect());
}

int ScrollZoomer::oppositeAxis(int axis) const
{
    switch(axis)
    {
        case QwtPlot::xBottom:
            return QwtPlot::xTop;
        case QwtPlot::xTop:
            return QwtPlot::xBottom;
        case QwtPlot::yLeft:
            return QwtPlot::yRight;
        case QwtPlot::yRight:
            return QwtPlot::yLeft;
        default:
            break;
    }

    return axis;
}
```

Vita

Renbin Zhou was born in Feburary 4, 1971 in a village Northeast China to Delin Zhou and Shuhua Dai. Hs sister, Renying Zhou, was born two years later. He attended school in Donggang Second High School, and got high school diploma in 1990. Then he went to Harbin Institute of Technology and got a Bchaler of Enginnering in 1994. He had worked for six years as a mechanical engineer in China until in 2000, when he got admission to the Aerospace, Mechanical and Biomedical Engineering Department in the University of Tennessee, Knoxville with a graduate teaching assistant. In 2002, he got a Master of Science in Mechnical Engineering with a concentration in Robotics and Controls. In 2001, he passed the Ph.D. qualification exam and then started his Ph.D. study in the Robotics and Eletromechanical Systems Laboratory after the Master degree, and his graduate teaching assistant changed to a graduate research assistant. He completed his Ph.D. work in the field of control of a transmission based servo actuator system using hybrid dynamic system theory. He got his Ph.D. degree in May, 2006. He is trying to land his career in the robotics research and industry after the graduation.