CONTROL OF MASTER-SLAVE ACTUATION SYSTEMS FOR MRI/FMRI COMPATIBLE HAPTIC INTERFACES

Ganesh Gowrishankar

(B.Eng.(Hons.), Delhi University)

A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF SCIENCE
NATIONAL UNIVERSITY OF SINGAPORE
SINGAPORE
2005

Acknowledgement

First of all, I wish to express sincere thanks to my supervisor Dr. Etienne Burdet for giving me the opportunity to work in this wonderful inter-disciplinary project, for spending time and energy to guide me in research, offering fresh perspectives to help hone my critical thinking skills and for giving me the opportunity to collaborate with the Advanced Telecommunications Research Institute International (ATR) in Japan and EPFL (Switzerland), an experience which I found immensely enjoyable and rewarding.

My sincere thanks Dr. Teo Chee Leong for his technical ideas and all the help with the administration during my Masters.

Special thanks to Roger Gassert, for being a good friend and helping me out during my stay in Switzerland. The long hours of work and travel with him in five different countries were a great learning experience and a lot of fun.

Special thanks to a very sincere friend, Tee Keng Peng, for patiently helping me with the basics of Neuroscience at the start of my Masters, being a good friend through my stay in Singapore and for the big help during submission of this thesis.

I am also thankful to Dr. Mitsuo Kawato, Dr. Ted Milner, Dr. Rieko Osu and Dr. Dave Franklin at ATR for elucidating neuroscience concepts crucial to the implementation of MRI experiments with the MR compatible haptic device.

Thanks to Dominique Chapuis, for his help in the design of the cable test-bed and realization of experiments on it.

Last but not the least, I would like to sincerely thank Mrs. Hamidah Bte Jasman, Ms. Tshin Oi Meng, and Mrs. Ooi-Toh Chew Hoey (the three angels) and Mr Yee Choon Seng for the wonderful and timely handling of administrative and technical matters and Mr Zhang Yao Ming for his help during my teaching assignment.

Table of Contents

Acknowledgement **i**

Table of Contents **ii**

Summary **iv**

1 Introduction **1**
1.1 Robots and the MR Scanner 1
1.2 MR Compatible Actuation 2
1.3 Thesis Outline 4

2 An MR compatible Wrist Interface with Hydrostatic Transmission **6**
2.1 Hardware 6
2.2 Real-Time System and Software 8
2.2.1 xPC Target as a Real-Time Environment 9
2.2.2 Code Structure 10
2.2.3 User Interface 13

3 Modelling and Simulation of a Hydrostatic Transmission **14**
3.1 Modelling 14
3.1.1 Hydrostatic Transmission 14
3.1.2 Master and Slave Systems 17
3.2 Simulation and Data Analysis 18
3.2.1 Parameter Selection 18
3.2.2 Validity of the model 19
3.3 Results 20
3.3.1 Assessing the nonlinear model 20
3.3.2 Dependence on Hose Diameter 22
3.3.3 Change of Hose Length 24

3.3.4 System with Short Hose 25

4 Pragmatic Control **27**
4.1 Control of Periodic Trajectory 28
4.2 Implementation of Force Fields and Force Control 29
4.3 Iterative Control for a Master-Slave System 31

5 Investigation of Cable Transmission **39**
5.1 Mechanical Design 40
5.2 Data Analysis 42
5.2.1 Stiffness 42
5.2.2 Static Friction 44
5.2.3 Master Slave Trajectories 46
5.3 Discussion 47

6 Conclusion **50**

Bibliography **52**

Appendix - Matlab Code and Simulink Modules **55**

Summary

Due to its fine spatial resolution and absence of ionizing radiations, Magnetic Resonance Imaging (MRI) has established itself as a standard diagnostics and advanced brain research tool. Functional MRI or fMRI is an excellent indicator of cerebral activity and has allowed significant advances in neuroscience.

Robots guided by MR imaging can revolutionize surgery. A haptic device working with an fMRI has great potential: it would enable neuroscientists to investigate the brain mechanisms involved in motion control. However the compatibility of the actuation system presents a major hurdle in the development of MR compatible robots. This thesis analyzes two master-slave kind of actuation systems driven by hydrostatic and cable transmissions as MR compatible actuation systems.

Both the hydrostatic system and the cable transmission present complex non-linear dynamics. The thesis presents the dynamic model and numerical simulation which were developed to analyze the dynamics of the system. The analysis helped in understanding the novel systems and in the development of the position and force control implemented on them. An iterative learning algorithm was developed to further improve position control, which gave good results with the simulation and will be implemented on the real plant.

A real time computer architecture using Simulink and the 'xPC target' toolbox enabled control at $500Hz$ and data acquisition at $2KHz$ over eight channels. The programming in the real-time system is intuitive and it is compatible with common PC-based hardware.

Finally the thesis presents a test-bed developed with a novel cable transmission which enables flexibility in the power transmission. Experiments carried out on this test-bed were used to compare the cable transmission with the hydrostatic transmission.

List of Figures

1.1 Neuroscience and robots 2

2.1 Hydrostatic actuation concept 7
2.2 Block Diagram of Control Structure. 12
2.3 Experiment user interface 13

3.1 Modelling of a hydrostatic transmission 16
3.2 Friction modelling 18
3.3 Real and simulated trajectories 20
3.4 Frequency analysis of hydrostatic system 21
3.5 System dynamics and hose diameter 23
3.6 System dynamics and hose length 24
3.7 Analysis of system with $1m$ long hose. 26

4.1 Position control results 28
4.2 Force control algorithm for a back-drivable hydrostatic transmission. 29
4.3 Feed-forward functions 30
4.4 Iterative learning for a master slave system. 32
4.5 Monotonicity of the transmission 33
4.6 Iterative learning results-A 37
4.7 Iterative learning results-B 38

5.1 The cable transmission test-bed 40
5.2 Cable transmission hardware 41
5.3 Stiffness of cable transmission 43

5.4 Cable stiffness with loading cycles 43
5.5 Static friction-hydrostatic vs cable transmission 45
5.6 Average static friction vs cable tension 45
5.7 Bode plot-hydrostatic vs cable transmission 46
5.8 Master and slave trajectories-hydrostatic vs cable transmission 47

Chapter 1

Introduction

1.1 Robots and the MR Scanner

Magnetic Resonance Imaging (MRI) has established itself as a standard diagnostics and advanced brain research tool. MRI has a fine spatial resolution, is well suited for visualization of soft tissues, and does not use ionizing radiation or injection of radioactive liquid [22]. A next challenge will consist of migrating MRI from diagnostic radiology to the operating room. MR compatible robots guided by real-time 3D imaging could revolutionize surgery, enabling more reliable and precise minimally invasive interventions with minimal recovery time.

Functional MRI or fMRI is an excellent indicator of cerebral activity and has allowed significant advances in neuroscience [12]. Haptic interfaces [1, 11, 15] can dynamically interact with humans performing movements and deliver forces fast and smooth enough to study neuromuscular response. Investigating adaptation to virtual dynamic environments produced by such interfaces has brought major advances in neuroscience [21, 5] (Fig. 1.1). A robotic haptic interface in conjunction with fMRI has great potential: it would enable neuroscientists to 'view' and investigate the brain mechanisms involved in performing tasks

A) B)

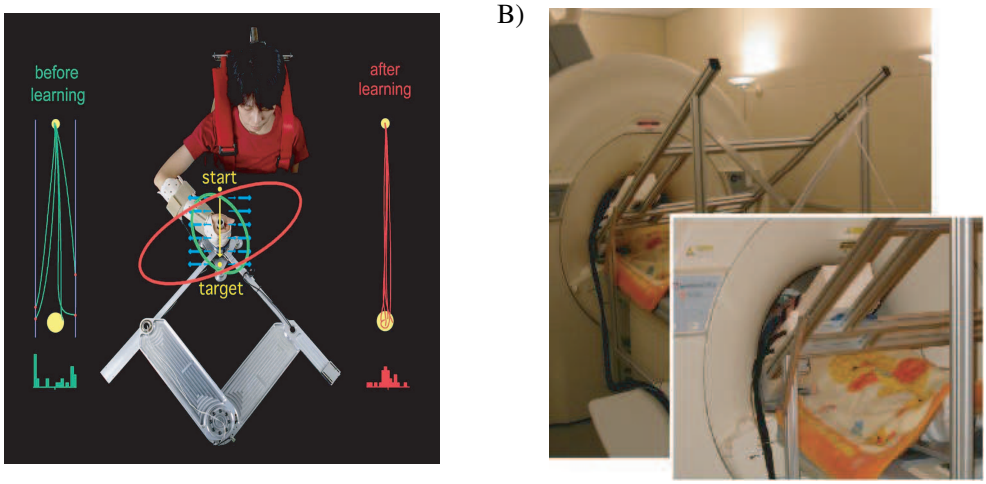


Figure 1.1: We investigate human motor control by examining the effect on motion and the adaptation to computer-controlled dynamics produced by a haptic interface. (A) Pictorial representation of the finding that the central nervous system stabilizes unstable dynamics by learning optimal impedance [5]. (B) fMRI compatible haptic interface installed at ATR in Japan.

with arbitrary dynamics. This could become a critical tool in neuroscience and rehabilitation.

1.2 MR Compatible Actuation

One main problem to create MR compatible robots is the electromagnetic compatibility of the actuation system. The MRI has a very high magnetic field of 1.5 to 7 Tesla, depending on the scanner. The actuation used to drive the robot should be inert to this magnetic field and should also not disturb the imaging. Various possibilities for actuation have been discussed in [3]. This thesis further investigates two possibilities, both consisting of placing a DC motor outside of the shielded room and using a transmission to bring power close to the scanner.

The haptic system installed at ATR, Japan at present, uses a master-slave kind of

arrangement driven by a novel fluid transmission. The transmission differs from conventional hydraulic transmissions which usually use a pump and valve system to regulate force. For the low working velocities that we work with, the force transfer in our system is essentially due to the static pressure transfer across the transmission fluid according to Pascal's law. We thus choose to refer to the transmission as a hydrostatic system instead of a hydraulic transmission.

The haptic system uses conventional actuators placed outside the shielded scanner room, hydrostatically connected to transmit power to a magnetically inert slave placed close to or inside the MRI scanner (Fig. 2.1). The pre-pressurized fluid in the pipes ensures that the delay time required to rise the pressure of the fluid is minimal, in comparison to a conventional hydraulic system with a pump. In contrast to a pneumatic transmission, a hydrostatic transmission should be stiff enough to transmit forces and motion with relatively short delays over the long distance (five to ten meters) from the master motor to the slave interface. Such a hydrostatic transmission can be used in many other applications, to produce large forces at a distant location and in any orientation [14], and a slave cylinder can have a much larger power/weight ratio than a motor placed at the slave.

Cables present a promising possibility for MR compatible transmission. A typical cable transmission will consist of a master actuator and a slave end effector connected by cables routed by pulleys. Such a cable transmission was designed and realized, enabling evaluation of performance and comparison with hydrostatic transmission. A special design provided a structurally rigid support for the cables with relatively flexible transmission routing.

Both the hydrostatic system and the cable transmission present complex non-linear dynamics with large static friction, and dynamic friction of Stribeck type [6, 20, 16]. In

particular, friction with the hydrostatic transmission varies with pressure, speed, acceleration and temperature. Dynamic models and numerical simulation were developed to analyze the dynamics of the systems. Simulations on the model gave an insight into the influence of various physical parameters on the plant.

The analysis helped in understanding the novel system and in the development of the position and force control implemented on the plant. Position control was used for experiments with guided movements, where the interface guided the subject hand in sinusoidal trajectories. The interface is inherently non-back drivable due to the presence of large static friction. Force control was used to counter the non-back drivability and to implement different force fields at the end effector.

The control of the system required communication at at least $500Hz$ and the experiments with the interface required collection of data at over $1kHz$ over eight channels. A real time computer architecture using Simulink and the 'xPC target' toolbox was able to satisfy these requirements. The programming in the real-time system is intuitive and it is compatible with common PC-based hardware. Further a interlinked programming structure using Matlab and Simulink provides an interactive interface which allows users to use the system easily.

1.3 Thesis Outline

The contribution of this thesis is the investigation of hydrostatic and cable systems as transmissions for MR compatible master-slave haptic devices. Numerical modelling is used to understand and compare the dynamics of the two transmissions. The modelling helped to develop suitable control architectures and implement them on the hardware developed by the Swiss Institute of Technology (EPFL). A real time computer structure is proposed

which was used to implement the control architectures on the real system and develop experiments.

The thesis is divided into six chapters. The 1-DOF MR compatible interface developed for ATR, is described in chapter 2. Chapter 2.1 describes the hardware and the principle of actuation. Chapter 2.2 describes the low cost real-time control system, which was used to control the two haptic devices with hydrostatic and cable transmissions. An elaborate support program with an interactive user interface was developed for running experiments with the haptic device. Chapter 3 presents the mechanical analysis of the hydrostatic transmission. It describes the modeling of the transmission system and simulation results. Chapter 4 describes the position and force control algorithms implemented on the system. It also describes a learning algorithm which was tested with simulations and will be implemented on the real plant in the future. Chapter 5 describes the test-bed developed to analyze a cable transmission and experimental results obtained with it. Finally Chapter 6 presents conclusions and propositions for future work.

Chapter 2

An MR compatible Wrist Interface with Hydrostatic Transmission

2.1 Hardware

The one-DOF haptic interface developed at EPFL (Fig. 2.1A) consists of a master, slave and the transmission. The master system consists of a conventional motor system running a multi-phase induction motor. The direct drive motor drives the master cylinder using a pulley and belt arrangement. The two chambers of the master cylinder are connected to the corresponding chambers of the slave cylinder using two $10m$ long, metal free hydrostatic pipe lines. The master and slave cylinders are shown in Fig. 2.1B,C. The slave side is made entirely of poly-oxy-methylene (POM) and is completely inert to magnetic fields. Any movement of the master cylinder is transfered to the slave due to the stiffness of the transmission lines. The motion of the slave cylinder is converted into a 1-DOF rotational movement using a pulley-belt arrangement on the slave side. An MRI compatible torque sensor, connected to the slave output, helps record the torque input on the slave. The torque sensor is essentially a plastic torque cell. The deflection of the torque cell is calculated by measuring the intensity of a reflected light beam [8]. The light beam is sent and received back from the slave side using fibre optic cables.

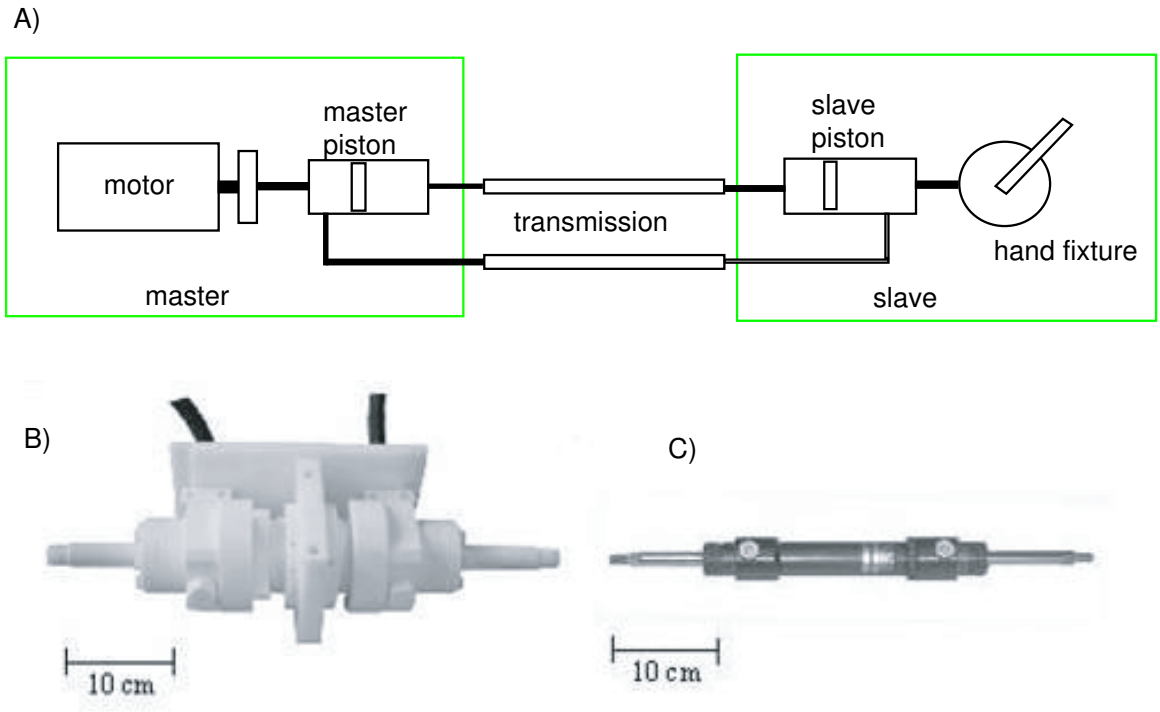


Figure 2.1: MR compatible actuation concept with hydrostatic transmission. (A) System components - the transmission lines link the master and slave systems in a closed loop.(B) MR compatible piston placed at the slave. (C) Equivalent commercial metallic piston used on the master side.

The computer hardware on the master side acquires sensor data (including master position, slave torque and upto six EMG[1] inputs), controls the system and runs the experiment programs. A Barbone Shuttle PC, with the following components makes up the computer hardware of the system:

- a $2GHz$ Intel Celeron processor.
- $256MB$ of RAM (for EMG and sensor data storage during the experiment).

[1]EMG: Electromyography, uses surface electrodes to detect the motor signals to the muscles. The EMG signal amplitude is correlated with muscle tension.

- a $60GB$ hard-disk to store experiment data as well as data from the scanner.
- an ethernet card for communication with a host PC and easy data transfer during and after an experiment.
- a NI-PCI 6024E data acquisition card form National Instruments (identical to the one used in the control of the 1DOF interface) to acquire sensor and EMG data and control the master actuators.
- an APCI-1710 encoder board to read in the position of the master actuators.

2.2 Real-Time System and Software

The one DOF interface was earlier controlled by a PC laptop running a LabWindows interface under Windows. A multifunction data acquisition card from National Instruments linked the control system to the hardware. This system had several drawbacks:

- Windows allows a maximal temporal resolution of $1ms$, limiting the maximum control frequency to $1kHz$.
- The control frequency is additionally limited by the Windows operating system, consuming an important part of the processing power and preventing a regular sampling. As a consequence, on the first prototype the maximum reachable control frequency was $500Hz$.
- Simultaneous acquisition of EMG is not possible, as EMG signals cover frequencies from 10 to $400Hz$ and should thus be sampled at $1kHz$.

However we required a control system that:

- can control interfaces with several degrees of freedom at $1kHz$ (two or three DOF).

- can handle additional sensors (force/torque and high resolution position encoders).
- can acquire EMG data at $2kHz$ in parallel to the system control.
- assures high flexibility for program code modification and integration of additional hardware.
- allows easy transfer of data for post-processing.
- can communicate with other devices to generate visual and auditive feedback and send and receive synchronization pulses.

A good real-time system would serve all these requirements and enable a better performance of the system. There are several real-time systems (hardware and software) available on the market. Out of these a limited choice were highlighted of interest for our application. The advantages and disadvantages of these systems were analyzed in [10], where xPC Target was found to be most suitable for the current application.

2.2.1 xPC Target as a Real-Time Environment

[2]xPC Target is an environment that uses a target PC, separate from a host PC, for running real-time applications. In this environment the desktop computer is used as a host PC with MATLAB, Simulink, and Stateflow (optional) to a create a model using Simulink blocks and Stateflow charts. After the model has been created, it can be simulated on the host PC in non real time. xPC Target enables addition of I/O blocks to a model, and then uses the host PC with Real-Time Workshop, Stateflow Coder (optional) and a C/C++ compiler to create an executable code, which can be executed on any processor that can run MS DOS. The executable code is downloaded from the host PC to the target PC running the xPC

[2]This section is based on the xPC online help manual - http://www.mathworks.com/ access/ helpdesk/help/ toolbox/xpc/xpc.shtml

Target real-time kernel. After downloading the executable code, the target application can be tested and run in real time, using full processing power of the target computer. The xPC kernel (OS running on target computer) fits on a floppy disk and allows basic graphical interfaces to directly display data on the target computer.

With respect to the current application xPC presents the following additional advantages:

- xPC is programmed with Simulink. As Simulink is very modular and allows creating "blocks", it is ideal for the block design of fMRI experiments.
- Unlike RealTime Windows target from MathWorks, xPC is a small operating system that runs on a target computer (without Microsoft Windows), presenting higher time resolution, more processing power and higher stability.
- Any data acquired within such a program is stored in MATLAB format and can easily be transferred to the host computer. This is a big advantage, as post-treatment of data is done in MATLAB at ATR.
- The numerical model developed of the hydraulic transmission was done with MATLAB, and can be used to generate control code with this system.
- xPC supports the NI-PCI 6024E data acquisition card used on the 1DOF interface prototype, and is thus compatible with the current hardware.

2.2.2 Code Structure

A modular code structure was developed using MATLAB, Simulink and the xPC toolbox to control the system. The control is supervised on the whole by a MATLAB program running on the Host Computer. The experiment itself is divided into several subprograms or modules

which, working in a sequence formed the entire experiment. The modules are modelled in Simulink using standard library blocks and specialized S-function blocks wherever required. The modules are modelled in a general format with some variable parameters which are supplied values from the MATLAB program. Each module is treated as a separate real time program. The main experiment sequence is programmed in the MATLAB program. The MATLAB Program activates the respective Simulink Modules following the plan of the experiment. When a module is to be activated, the MATLAB program builds the Simulink model corresponding to the Module and runs it on the Target computer. At the end of the run, the Module is unloaded from the Target PC and the next one is loaded. Some other functions performed by the Matlab program are given below :

- It controls the front end functions of the GUI including display of instructions to the subject.
- When the program is loaded for the first time it checks for connection errors.
- It takes in the experiment parameters and builds the different programs.
- It structures the experiment according to user defined parameters.
- During the experiment it loads and runs the different Simulink models.
- In between the execution of each model it acquires stored data from the 'Target' to be stored onto the 'Host'.
- While it performs the other tasks it synchronizes the experiment according to the system clock on the Target.

More details of the Matlab and Simulink codes can be found in the Appendix.

Advantages of the Code Structure

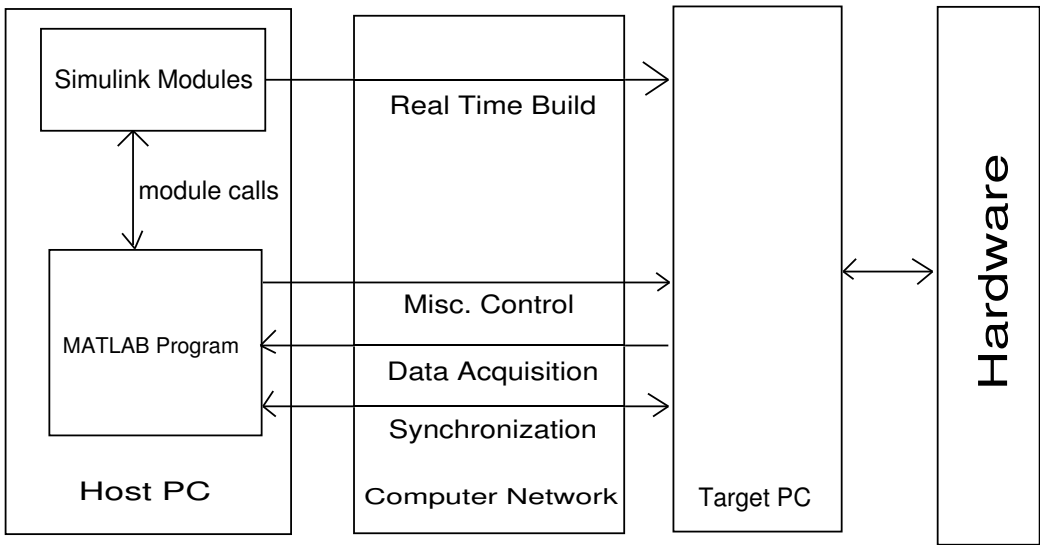


Figure 2.2: Block Diagram of Control Structure.

- The main advantage of this architecture is to divide the whole experiment into smaller real time programs. This is very useful for data acquisition from the Target PC as data cannot be retrieved during the run time of a program. By having smaller programs or modules, data can be retrieved after each of these modules without the requirement of a large buffer on the Target PC.

- As the whole experiment is basically managed in the MATLAB environment it is easy to use MATLAB models in parallel with the actual system to aid in control. It thus forms a good way to incorporate actual and virtual systems in parallel with each other using values of each others variables to update their own behavior.

- The present experiment consists of similar repetitive sessions whose programming becomes much easier if done in a modular fashion.

- Modular programming makes the whole control process more organized and easy to debug.

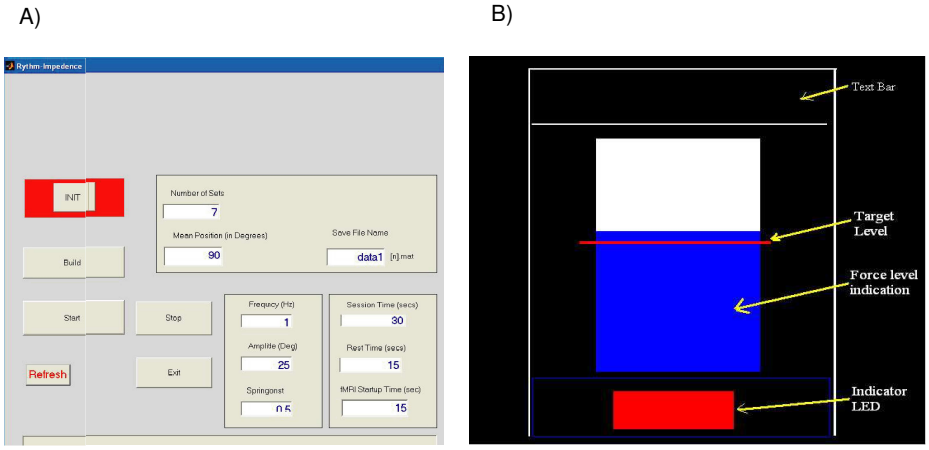


Figure 2.3: A) Experiment setup screen on the Host Computer, B) Subject screen used to give feedback to the subject in the scanner during the experiment.

- Planning and modifications in experiments become easier as the experiment structure is in a separate MATLAB file.

Disadvantage

- The architecture requires a continuous synchronization of the Host and the Target clocks.

2.2.3 User Interface

The user interface provides visual feedback to the subject and allows easy setting of experiment parameters. The user interface in the present setup was designed using the MATLAB GUIDE (MATLAB Graphic User Interface Developing Environment). The interface consists of two screens, one for the experiment setup and the other to give feedback to the subject inside the MRI scanner. The experiment setup screen provides a interactive interface to help the user to setup the various experiment parameters and control the experiment. Snapshots of the user interface screens are shown in Fig. 2.3.

Chapter 3

Modelling and Simulation of a Hydrostatic Transmission

The MR compatible haptic interface of Chapter 2 consists of master and slave systems connected in a cyclic arrangement by two hydrostatic transmission lines. It has non-negligible compliance, in particular due to the transmission lines. Friction in the master and the slave systems and fluid friction in the transmission lines affect the behavior significantly. A model was developed to further analyze these dynamics and help developing the control. The model's main assumptions are: *i*) the velocity of the fluid is approximately constant throughout the length of the pipeline; *ii*) the change in cross-sectional area of the pipes due to bulging is negligible; *iii*) the motor can be modeled as a flywheel with inertia and friction at the bearings.

3.1 Modelling

3.1.1 Hydrostatic Transmission

Even though fluid in the pipe has very low compressibility, the long pipes will result in non negligible compliance. We now show that the pipe dynamics can be modeled in a natural

way as a spring. Let the volume of the fluid in the pipeline be

$$V = A\,L\,, \tag{3.1.1}$$

where A is the cross section of the pipe and L its length. Assuming that the variation in cross-sectional area is negligible, we can write

$$dV = A\,dL\,. \tag{3.1.2}$$

The bulk modulus B of the oil is defined as

$$B = \frac{dP}{dV/V} \quad \text{or} \quad B\,dV = dP\,V \tag{3.1.3}$$

where P is the oil pressure in the pipes. Substituting (3.1.1) and (3.1.2) in (3.1.3) yields

$$B\,A\,dL = dP\,A\,L \tag{3.1.4}$$

A force F applied by the piston on the fluid line will lead to a pressure change dP in the pressurized system described by

$$dP = \frac{F}{A}\,. \tag{3.1.5}$$

Substituting (3.1.5) in (3.1.4) yields

$$F = \frac{B\,A}{L}\,dL \equiv K\,dL\,, \tag{3.1.6}$$

which is the equation of a linear spring. The fluid stiffness

$$K \equiv \frac{B\,A}{L} \tag{3.1.7}$$

decreases with L and increases with B and A, as expected.

We model each of the two pipelines connecting the actuator with the slave as a mass between two springs of stiffness $2\,K$ (Fig. 3.1). The dynamics of the two transmission lines are modeled as

$$m_l\,\ddot{s}_1 = -F_{lf1} - F_m(s_1, \dot{s}_1) - F_s(s_1, \dot{s}_1) \tag{3.1.8}$$

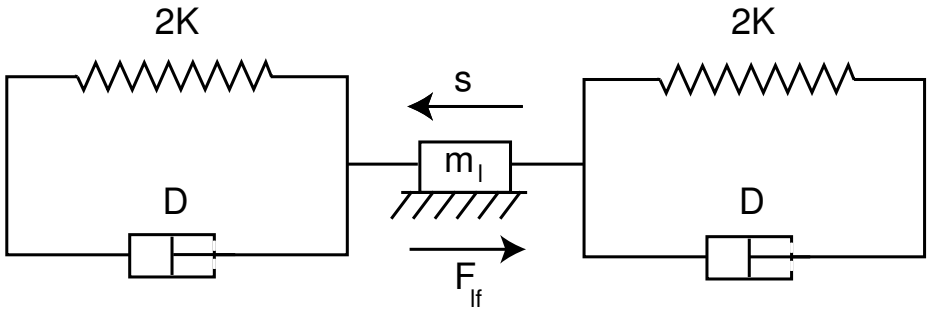


Figure 3.1: The hydrostatic transmission is modeled as a spring damper system. Two such systems are used to represent the two lines connecting the master actuator with the slave.

$$m_l \, \ddot{s}_2 = -F_{lf2} - F_s(s_2, \dot{s}_2) - F_m(s_2, \dot{s}_2)$$

where the fluid mass in the pipes m_l is the product of the pipe volume $A\,L$ by the fluid density ρ: $m_l \equiv A\,L\,\rho$, the variables s_1 and s_2 represent the position states of the fluid in the two transmission lines, F_{lf1} and F_{lf2} the corresponding fluid friction (as described below Eqn. 3.1.10). The pipe spring-like property is described as

$$F_m(s, \dot{s}) = 2\,K(s - \alpha_m\, q_m\, r_{mp}) + D\,(\dot{s} - \alpha_m\, \dot{q}_m\, r_{mp})$$

$$F_s(s, \dot{s}) = 2\,K(s - \alpha_s\, q_s\, r_{sp}) + D\,(\dot{s} - \alpha_s\, \dot{q}_s\, r_{sp}) \tag{3.1.9}$$

where q_m and q_s are the angular displacements of the master and slave respectively, r_{mp} and r_{sp} are the radii of the belt pulley on the master and the slave sides, and α_m and α_s are the ratios of cross sectional areas of the master cylinder and slave cylinder by that of the transmission hose. The damping term D accounts for friction at the cylinder ports and friction due to bends in the pipes, not considered in the fluid friction modeling.

Fluid friction is given by

$$F_{lf} = H_f\, \rho\, g\, A\,. \tag{3.1.10}$$

In this equation, g is the gravity constant, and the friction head loss H_f is calculated from

the Darcy-Weisbach formula

$$H_f = \frac{f\,L\,v^2}{2\,g\,d}\,, \tag{3.1.11}$$

where L and d represent the pipe length and diameter respectively, and v is the fluid velocity. f is the friction factor which depends on the Reynolds number of the fluid given as $Re = \frac{\rho\,v\,d}{\nu}$, where ν is the viscosity of the fluid in Ns/m . For Reynold numbers of less than 2000, *i.e.*, laminar flow, f can be calculated as $f = 64/Re$. For turbulent flow the simulation uses a simplified form of the Colebrook equation for friction calculation.

3.1.2 Master and Slave Systems

The master and slave dynamics are described by

$$I_m\,\ddot{q}_m = \tau_m - \tau_{mf} + r_{mp}(-F_{mf} + F_m(s_1, \dot{s}_1) + F_m(s_2, \dot{s}_2))$$

$$I_s\,\ddot{q}_s = -\tau_{sf} + r_{sp}(F_s(s_1, \dot{s_1}) + F_s(s_2, \dot{s}_2)) \tag{3.1.12}$$

where τ_m represents the motor torque, τ_{mf} is the friction torque in the motor (Fig. 3.2A) and F_{mf} the friction in the master piston. The combined friction torque τ_{sf} on the slave side is due to the friction in the slave piston, and in the hand fixture. F_m and F_s are the interaction forces of the master and the slave with the transmission lines (Eqn. (3.1.9)). The master and slave side inertias I_m and I_s are given by

$$I_m = I + I_{mp} + M_m\,r_{mp}^2 \tag{3.1.13}$$

$$I_s = I_{sh} + I_{sp} + M_s\,r_{sp}^2\,,$$

where I and I_{sh} refer to the motor inertia and slave hand fixture inertia respectively, I_{mp} and I_{sp} are the moment of inertia of the master and slave side pulleys, M_m and M_s are the mass of the master and slave pistons.

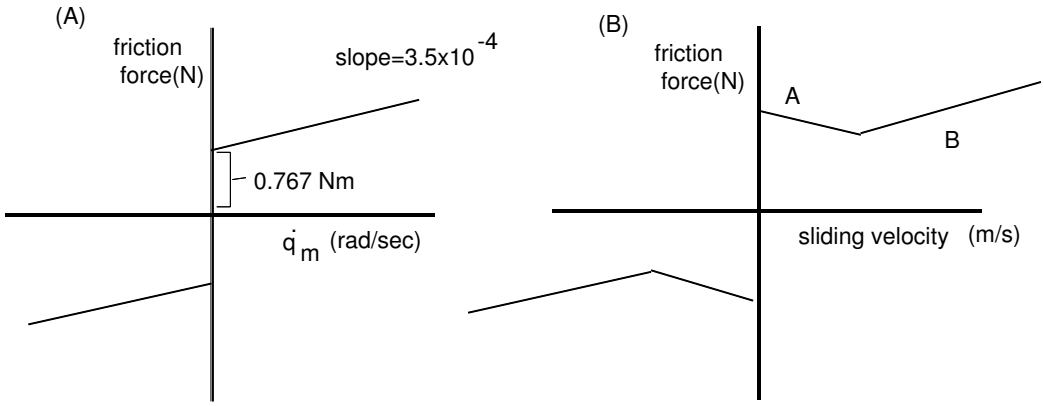


Figure 3.2: Modeling friction in the motor (A) and in the pulleys and pistons (B).

A piecewise linear function (Fig. 3.2B) corresponding to Stribeck friction [6, 16] models the friction in the pulleys and pistons on the master and slave sides, *i.e.*, F_{mf} and τ_{sf}.

3.2 Simulation and Data Analysis

The dynamics of the actuator and slave connected by the hydrostatic transmission are described by Eqns. (3.1.8 and 3.1.12). For simulation, this system of equations is Euler integrated at $2kHz$. Sinusoidal and ramp movements were simulated in order to compare the performance of the simulation with the actual system. The system's behavior and critical parameters were examined using frequency analysis and by examining the energy transmission. This measure of efficiency in the transmission, depending mainly on friction, is defined as the ratio of the amplitudes of the input and output energy curves:

$$\mu \equiv \frac{E_s}{E_m},\ E_s = \max(\tau_m\, \dot{q}_s),\ E_m = \max(\tau_s\, \dot{q}_m) \tag{3.2.1}$$

3.2.1 Parameter Selection

The diameter of the hose $d \equiv 0.009m$ and the cross-sectional area of the hose $A \equiv 1.760 \times 10^{-4}m^2$ are obtained from the hose data sheets, and the default length is $L \equiv 10m$. The

fluid bulk modulus $B \equiv 1.860 \times 10^9 N/m^2$, viscosity $\nu \equiv 8.470 \times 10^{-3} Ns/m$ and density $\rho \equiv 847 kg/m^3$ are from the data sheet. The fluid mass in each hose is obtained by multiplying the hose volume $A\,L$ with the density $M_l = A\,L\rho \equiv 1.197 kg$. The radii of the pulleys $r_{mp} \equiv 0.031 m$ and $r_{sp} \equiv 0.034 m$ are measured on the actual system. The motor inertia $I \equiv 2.500 \times 10^{-3} kg\,m^2$ is provided by the manufacturer. The inertias of the pulleys $I_{mp} \equiv 4.500 \times 10^{-4} kg\,m^2$ and $I_{sp} \equiv 1.270 \times 10^{-4} kg\,m^2$, the inertia of the slave hand fixture $I_{sh} \equiv 1.6 \times 10^{-3} kg\,m^2$ and the masses of pistons $M_m \equiv 0.239 kg$ and $M_s \equiv 0.254 kg$ are obtained from the design data. The area ratios $\alpha_m \equiv 1.780$ and $\alpha_s \equiv 1.900$ are computed as the ratio of the cylinder and the hose cross sectional areas measured on the master side and slave side respectively.

The motor friction $\tau_{mf} \equiv 0.767 + 3.5 \cdot 10^{-4}\,\dot{q}_m \;\; Nm$ is supplied by the manufacturer. The damping factor $D \equiv 0.3$ in the transmission modelling and the master piston friction F_{mf} and slave friction τ_{sf} parameters (Fig. 3.2B) are tuned by comparing the behavior of the system and the simulation in ramp and sinus movements of frequencies between 0.2 and $2 Hz$. We find:

$$F_{mf} \equiv 0.200 - 0.200\,\dot{q}_m\,r_{mp}\,, \qquad \dot{q}_m < 6 rad/s$$

$$F_{mf} \equiv 0.240 + 0.200(\dot{q}_m - 6)\,r_{mp}\;,\; \dot{q_m} \geq 6\,rad/s$$

and

$$\tau_{sf} \equiv 2.30 - 0.2\,\dot{q}_s\;Nm\,, \qquad \dot{q}_s < 6\,rad/s$$

$$\tau_{sf} \equiv 1.10 + 0.2(\dot{q}_s - 6)\;Nm\;\;,\;\; \dot{q}_s \geq 6\,rad/s$$

3.2.2 Validity of the model

The model does not take into account the bulging of the hoses due to fluid pressure, which was considered negligible as the working pressure $15 bar$ of the fluid is considerably lower than the design specification of $100 bar$ claimed by the manufacturer. The extension of the

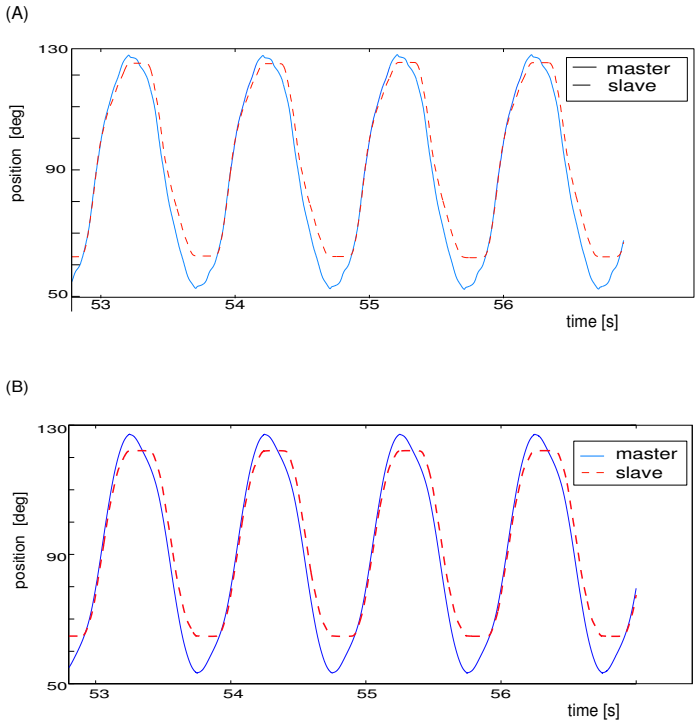


Figure 3.3: Actual trajectory on the master (solid) and slave (dashed) corresponding to a $1Hz$ sinusoidal desired trajectory. (A) is data measured on the real plant, (B) from the simulation. The simulation reproduces even the small kink in the master movement due to static friction.

belts under load was also neglected.

The actual system, which is essentially a spring with mass, is modelled as a 2-DOF system with massless springs. For low stiffness the second resonance of the model is excited and interferes with the model behavior. Consequently the model gives invalid results for low stiffness conditions, in particular very small diameter hoses.

3.3 Results

3.3.1 Assessing the nonlinear model

Fig. 3.3 shows the actual and simulated sinusoidal movements of $1Hz$ frequency. We see that the model's behavior is close to that of the plant. The model can also reproduce the

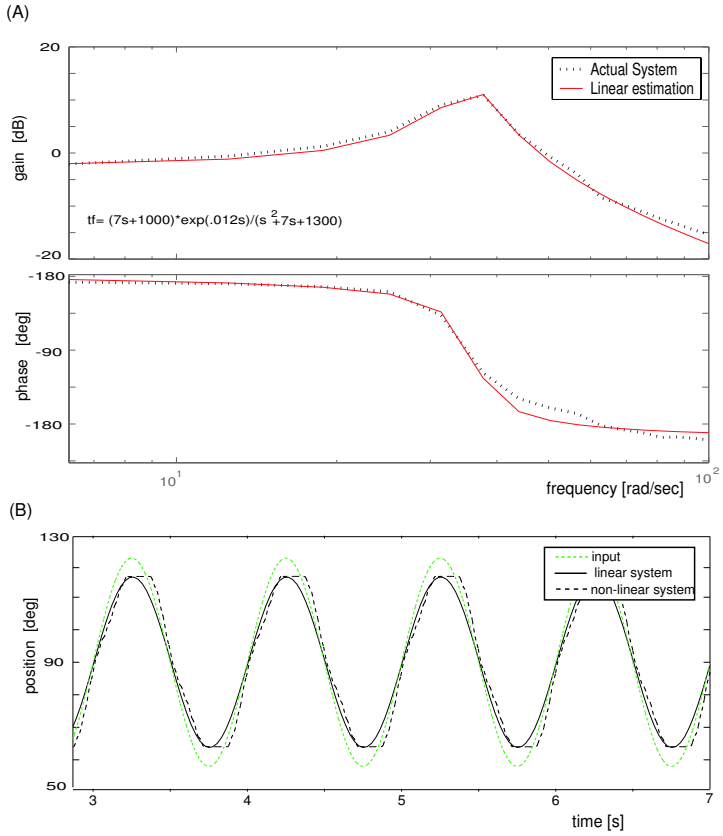


Figure 3.4: (A): Magnitude and phase of 0.2 to 24Hz sinusoidal movements with the nonlinear model (dashed) and with a linear system obtained by frequency response (solid). (B): Comparison of periodical movement with nonlinear model (dashed), with the linear approximation (solid) and input (dotted). The major discrepancies lie where the movement changes direction.

small kink in the master movement due to stick slip between the master piston and cylinder, as well as the plateau when the slave changes its direction. Similar results are obtained at other frequencies. When the frequency of the input signal is increased, the output amplitude decreases. The output, *i.e.*, the slave movement, almost disappears for frequencies above approximately 20Hz.

Could a simple linear system reproduce the dynamic behavior of the plant sufficiently well? To examine this we determine a linear model approximating the nonlinear model's

behavior using the frequency response method [19]. The resulting second order linear model with a transport lag has the transfer function:

$$\frac{7\,s + 1000}{s^2 + 7\,s + 1300}\,e^{-0.012s} \tag{3.3.1}$$

This linear system acts as a low pass filter with a cut-off frequency of around $20Hz$ and a resonance frequency of $7Hz$, corresponding to the results observed in the actual system. Although the linear system has roughly similar Bode plots to the nonlinear model (Fig. 3.4A), it is unable to predict small oscillations that would be detected by haptic senses. The major source of non-linearity of the real system is nonlinear static friction. Therefore the major discrepancies between the nonlinear model and its linear approximation occur when the movement is changing direction (Fig. 3.4B).

It would be difficult to predict the system's behavior heuristically, because the system variables influence the mass of fluid, friction, and stiffness of the transmission lines simultaneously. However, the nonlinear model behaves similarly to the real system and can be used to evaluate how the system would behave in various conditions as well as to identify the critical parameters.

3.3.2 Dependence on Hose Diameter

The main variable physical parameters of the system are the length and diameter of the hose. Fig. 3.5A shows the Bode plot for different diameters of hose when the length is kept at $10m$. The phase lag increases when the diameter decreases. The peak of the gain plot, corresponding to the resonance frequency of the system, increases with increase in hose diameter.This is due to the decrease in friction with increase of diameter (see Eqn. (3.1.11)). However the resonance frequency of the system is found to be independent of the hose diameter. This may be due to the fact that the resonant frequency depends on the

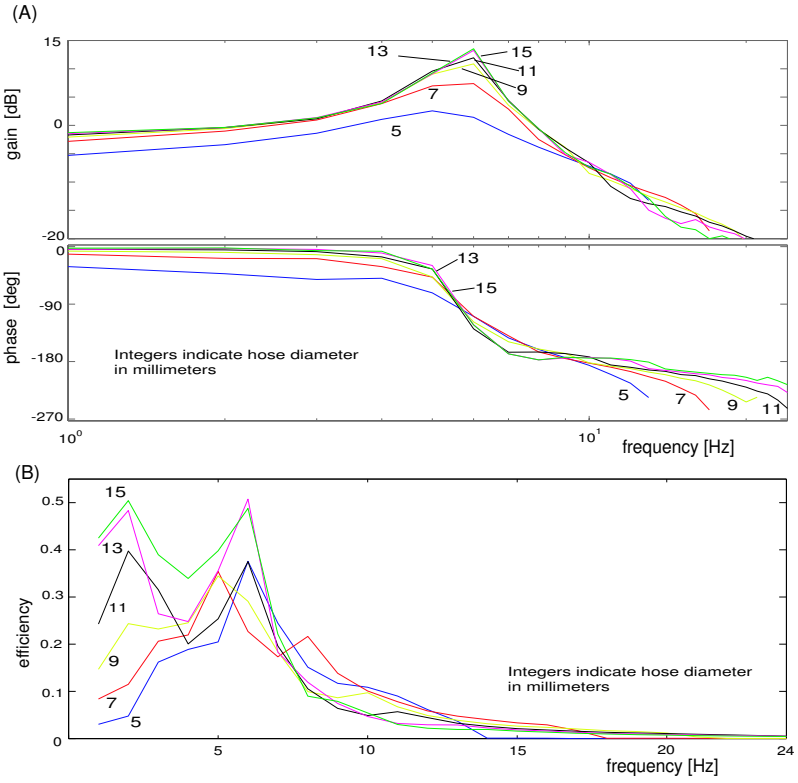


Figure 3.5: Influence of the hose diameter on the transmission dynamics. (A) Bode plot of the system with varying hose diameter from $0.005m$ to $0.015m$. (B) Efficiency curve for different hose diameters over a frequency range of 0 to $24Hz$.

ratio of stiffness and inertia, and both the stiffness of the system and its inertia decrease with the hose diameter.

The Bode plot informs us about changes in the system's mass and stiffness. Another major factor affected by the change in hose diameter is the friction in the hose. To analyze the effect of this non-conservative force we examine the losses in the pipe using the measure of transmission efficiency (Eqn. 3.2.1). The efficiency measure for various hose diameters decreases with frequency (Fig. 3.5B), corresponding to the low-pass system's characteristics. In general the efficiency of the system is found to increase with increase in hose diameter. This can be attributed to the decrease in friction losses with the increase of diameter. In

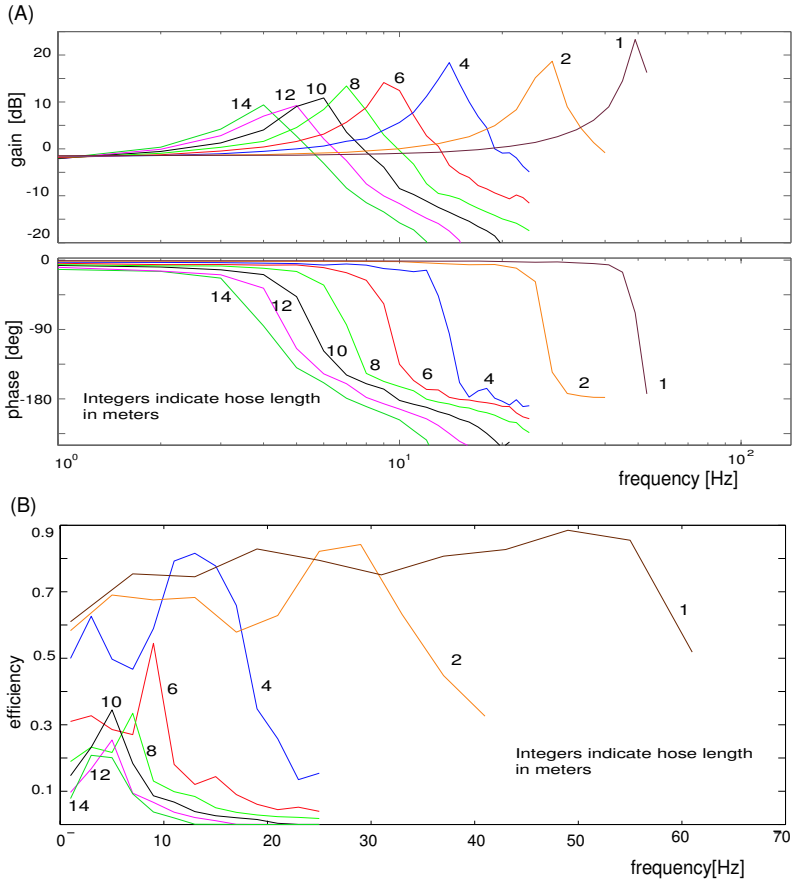


Figure 3.6: Influence of hose length. (A) Bode plot of system with different hose lengths over the frequency range of 0 to 24*Hz*. (B) Efficiency curve with varying hose length. With increase in length the frequency corresponding to maximum efficiency decreases. In general, the efficiency decreases with hose length.

addition an increase in the hose diameter also also increases the inertia of the system. At lower speeds, when stick-slip critically affects the behavior, higher inertia can overcome it better, giving higher efficiencies in the system.

3.3.3 Change of Hose Length

Fig. 3.6A shows that when the hose length becomes shorter the peak of the Bode gain curve shifts to higher frequencies, corresponding to an increase in the natural frequency of

the system. This may be attributed to the fact that with decrease in the hose length the mass of fluid in the transmission decreases and its stiffness increases, both contributing to increase the resonance frequency. The peak of the gain plot decreases in magnitude with the increase in hose length. At small lengths (i.e. below $5m$) the phase lag remains very small until resonance.

Fig. 3.6B shows the efficiency curves for the different hose lengths. The efficiency becomes larger at lower lengths as the friction decreases. For each length the efficiency values fall sharply after a particular frequency, which approximately corresponds to the cut-off frequency for that length.

In summary, with smaller hose lengths the system's inertia is reduced and the stiffness increases, which results in a more rigid coupling between the master and slave systems and also reduces friction.

3.3.4 System with Short Hose

Fig. 3.7A shows the Bode diagram of the system with $1m$ long hose, and of a linear approximation with transfer function

$$\frac{8.0 \times 10^4}{s^2 + 18\,s + 95500} \tag{3.3.2}$$

identified using the frequency response method. This second order linear system has a natural frequency of $49.5Hz$ and a (low pass) cutoff frequency of approximately $100Hz$. We see that the Bode plot of the linear system is very close to that of the nonlinear model. Correspondingly, in this short hose case the behavior predicted by the linear model is very close to the behavior simulated with the nonlinear model (Fig. 3.7B), in particular for low frequencies.

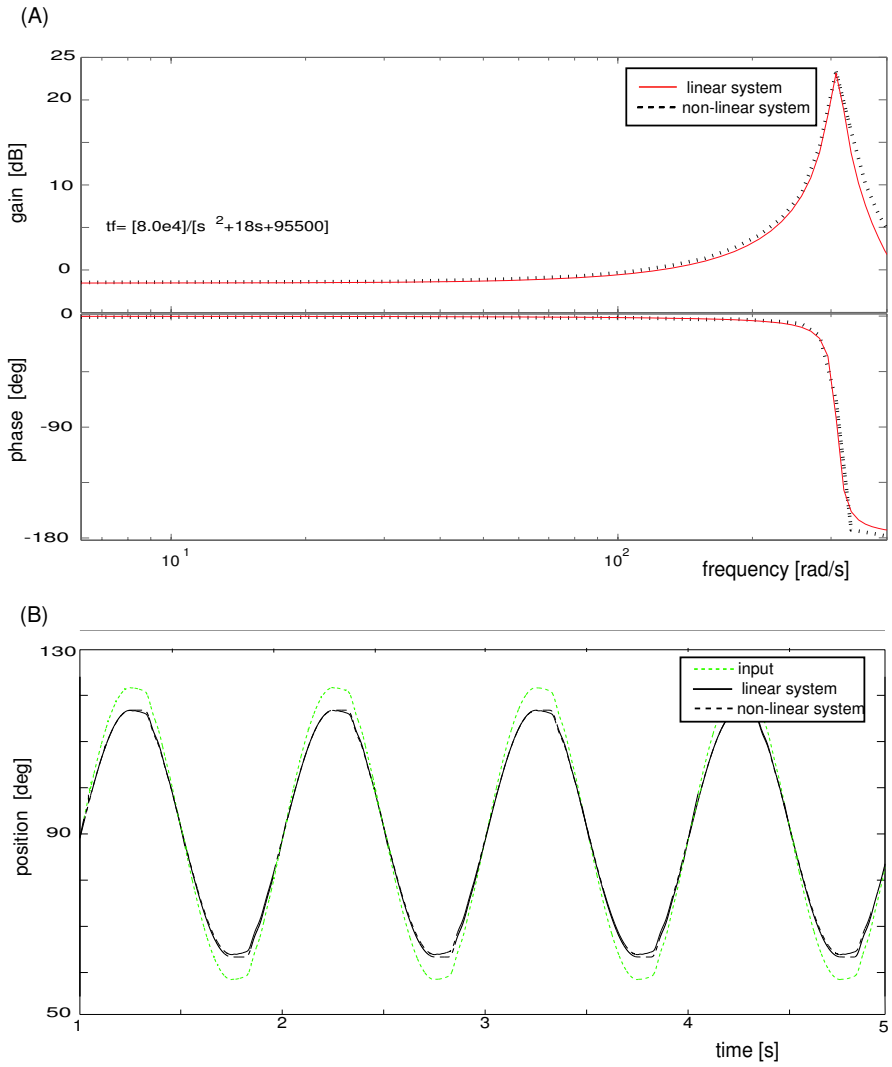


Figure 3.7: Analysis of the master-slave system with $1m$ long hose. (A) Bode diagram of the simulated system (dashed) and behavior of the approximated linear system (solid). (B) Outputs from the transfer function and the simulated system for similar inputs.

Chapter 4

Pragmatic Control

It is difficult to use the nonlinear model of chapter 3 directly to compute feedforward dynamics and control the real plant. Moreover, this model does not account for all effects of the real plant's dynamics. For example, measurement of the motor signal at very low speed showed that static friction varies with position and is slightly higher at the boundaries of the piston range. Finally, model-based control using a realistic model may not improve the control of our system with hydraulic transmission much. As an analogy, while nonlinear adaptive control can typically reduce the tracking error of robots equipped with DC motors by a factor of 10 or more [2, 4], it seems to hardly improve the behavior of robots with hydraulic actuators [13] with similar friction characteristics to our system.

A *pragmatic control* was developed using the main features of the model of chapter 3 and offering the least resistance as felt by the operator. Friction dominates at the relatively low frequency at which the interface will be used to study the control of human movement, i.e., 0.2 to $2Hz$. Static friction of $1.5 \pm 0.3\,V$ was identified from a very slow $0.03Hz$ sinusoidal movement of 50^o amplitude. Also, it was observed that the resistance was least when the friction feedforward decreased with velocity, corresponding to the small magnitude part of the Stribeck curve (region A in Fig. 3.2B).

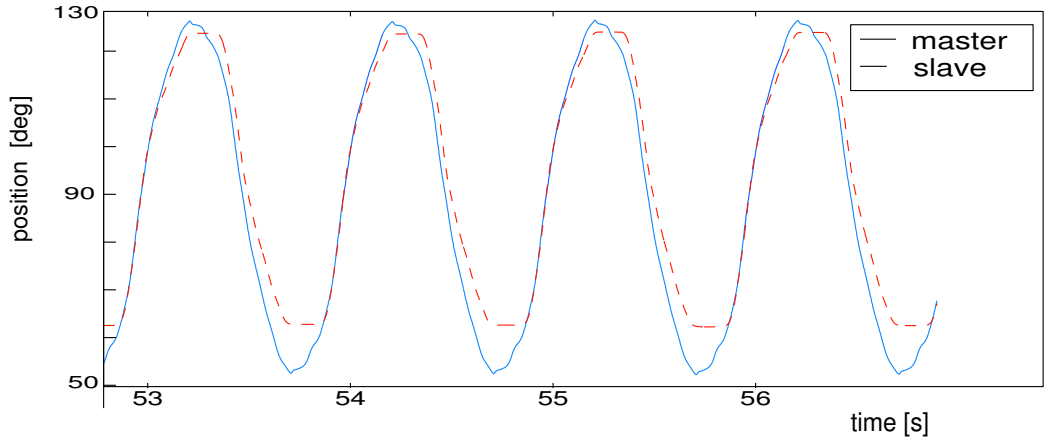


Figure 4.1: The figure shows master and slave side movements achieved with position control. The slave movements are smooth overall except at the peaks where the movement is cut off because of the static friction.

Two particular kinds of control were developed with feedforward and feedback for tasks necessary to investigate human motor control: *i*) guided periodic movements (requiring trajectory control) and *ii*) goal directed movements without resistance and with computer-controlled force fields (requiring force control to move the non back-drivable plant).

4.1 Control of Periodic Trajectory

Trajectory tracking of sinusoidal movements with various frequencies was achieved using a feedforward term and a proportional derivative feedback term of the trajectory error. The feedforward was the addition of a sine compensating for inertia and a triangular term corresponding to friction, providing extra torque at the extremes of the movement when movement is slow and lower values otherwise. The overall feedforward curve is shown in Fig.4.3A. The magnitude of the feedforward was taken to be the average motor input voltage corresponding to the static friction in the system. The gain values are $P \equiv 4.5 \times 10^{-3}\ V/counts$ and $D \equiv 7 \times 10^{-4}\ V/counts/sec$, where *counts* refers to encoder counts.

This control results in smooth trajectories (slave curve in Fig.4.1), and the subjects do

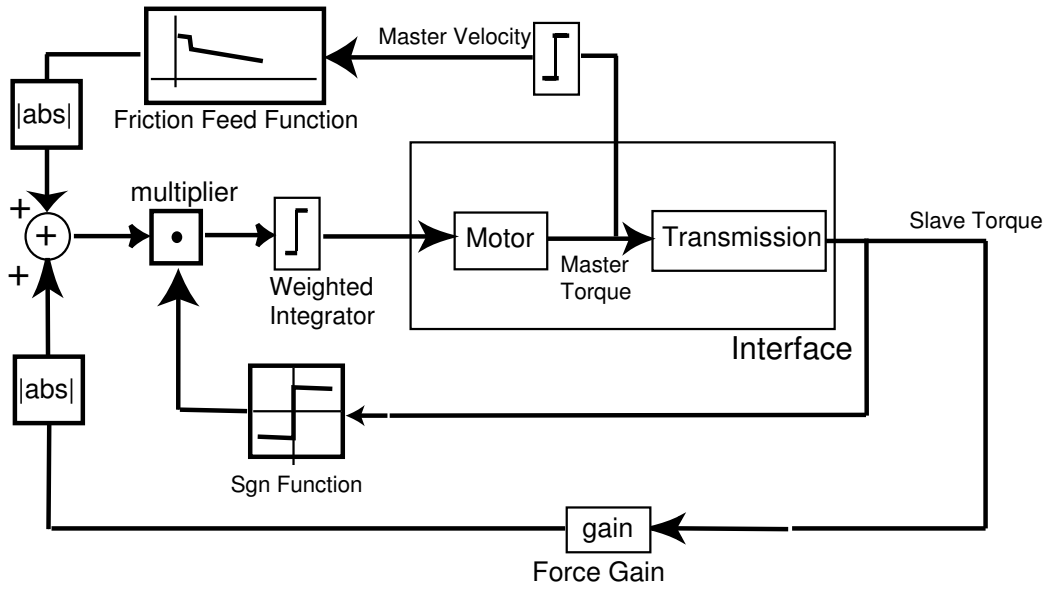


Figure 4.2: Force control algorithm for a back-drivable hydrostatic transmission.

not perceive a kink.

4.2 Implementation of Force Fields and Force Control

It is practically impossible to move the motor by moving the handle because of the large friction, *i.e.*, our transmission is non back-drivable. To enable 'free' movement of the hand, the torque exerted by the hand is measured and torque control is used to minimize it [7]. Further, smooth control requires compensation for static friction, which is difficult to realize without detecting the direction in which the subject wants to move. The sign of the torque sensor voltage is the only signal available to infer the movement direction, and at low speed (due to the stick slip behavior and the inertia of the hand) the torque sensor signal regularly changes sign even though the movement is in the same direction. This occurs for example when slowing down a movement while continuing in the same direction.

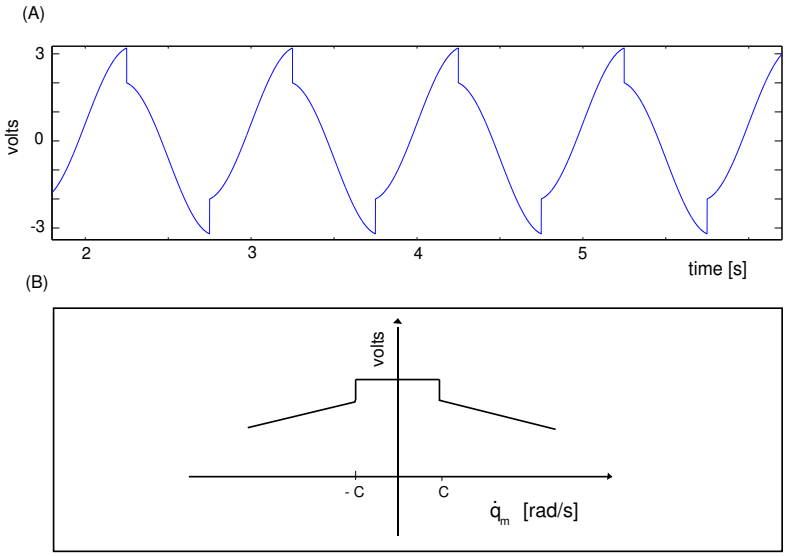


Figure 4.3: (A) The feedforward curve used in sinusoidal trajectory tracking. The peaks at the top compensate for the static friction. (B) The friction compensation function (Eqn. 4.2.3) which was found to provide movements with least resistance. A higher magnitude of compensation is given at low speed. The compensation decreases with increasing velocity.

An integral term in the control will act as a low-pass filter and smoothen the signal from the torque sensor, which may solve the problem of unwanted direction changes in the motor control signal. However, integral control suffers from wind-up problems and has to be reset regularly or it becomes very sluggish in responding to direction change; one time resetting of the integral leads to jerk.

An integral control with a forgetting factor was used to achieve a smooth resetting. The control signal to the motor (in volts) is:

$$V_m(t) = (1 - \beta)\, V_m(t-1) + \beta\, v_m(t) \tag{4.2.1}$$

where $0 < \beta < 1$ represents the forgetting factor and $\beta = 0.4$ gave satisfying results. v_m is computed as

$$v_m(t) = P_1\, v_s(t) + G\left(\dot{q}_m\right), \tag{4.2.2}$$

where $v_s(t)$ (in volts) represents the torque sensor signal at time t and the offset function

$G(\dot{q}_m)$ is given (in volts) as

$$G(\dot{q}_m) = 1.5\,, \qquad \dot{q}_m < C \tag{4.2.3}$$

$$= 1.2 - P_2\,|\dot{q}_m - C|\,,\ \dot{q}_m \geq C$$

The velocity threshold $C \equiv 1000 counts/s$ and gains $P_1 \equiv 0.333$, $P_2 \equiv 1.11 \cdot 10^{-4} volts\ s\ /\ counts$ were tuned to provide least resistance. This offset function compensates for static friction and the decrease in friction with increase in velocity. The offset direction is the direction of the torque signal and not that of the velocity. This helps in two ways:

- A point-to-point movement is composed of an acceleration and a deceleration phases. During acceleration, the torque has the correct sign. During deceleration, when the direction of velocity and torque are opposite, the offset helps in faster resetting of the integral term. This in turn leads to faster system response to a change in direction.

- Our system has no velocity sensor on the slave side, close to the MRI, and using the master velocity for control causes jerks at low speed.

Using this control, the subjects are able to perform free point-to-point movements of amplitude up to 120^o and frequency up to $3Hz$. Various position and velocity dependent force fields (e.g. [17]) can be superimposed on this free movement, which will enable us to infer the brain mechanisms of adaptation to these novel dynamics.

4.3 Iterative Control for a Master-Slave System

Position control on the interface is achieved by defining a reference on the master side and controlling the master about this reference. Adding a feedforward approximating static friction improves the control performance. With this control corresponding to the sinus

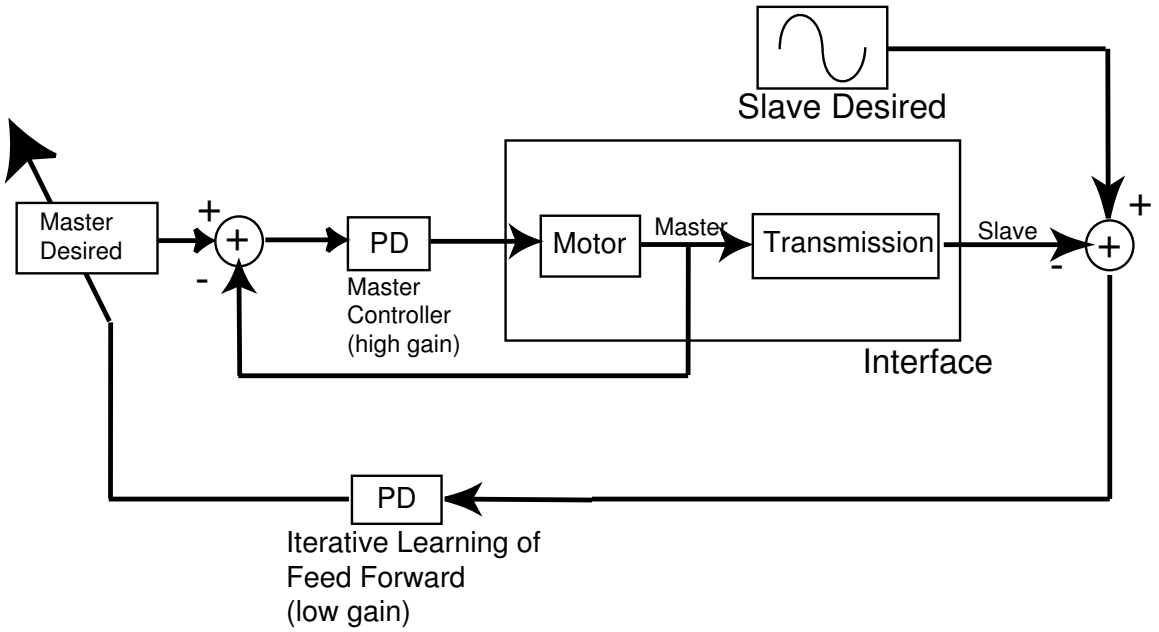


Figure 4.4: Iterative learning for a master slave system.

movement of the master, an approximate sinus is achieved on the slave side. A difference in the movement on the master and slave side develops due to the non-linearity in the plant chiefly due to static friction. Though non-linear, the transmission is inherently monotonic. That is, considering the master reference as input and the corresponding slave movement as output, a positive input always gives a corresponding positive output from the system within the limits of the transmission delays. If $y(t)$ and $u(t)$ represent the slave movement and master desired at any time instant t, then a change in the input u induces a change of position in the output y in the same direction:

$$\Delta y = g\,\Delta u \tag{4.3.1}$$

$$0 \leq \alpha \leq g \leq \beta < \infty$$

where g may vary with time but α and β are constants. The sign of the master and slave velocities during a period of simulated learning is shown in Fig. 4.5. The figure shows

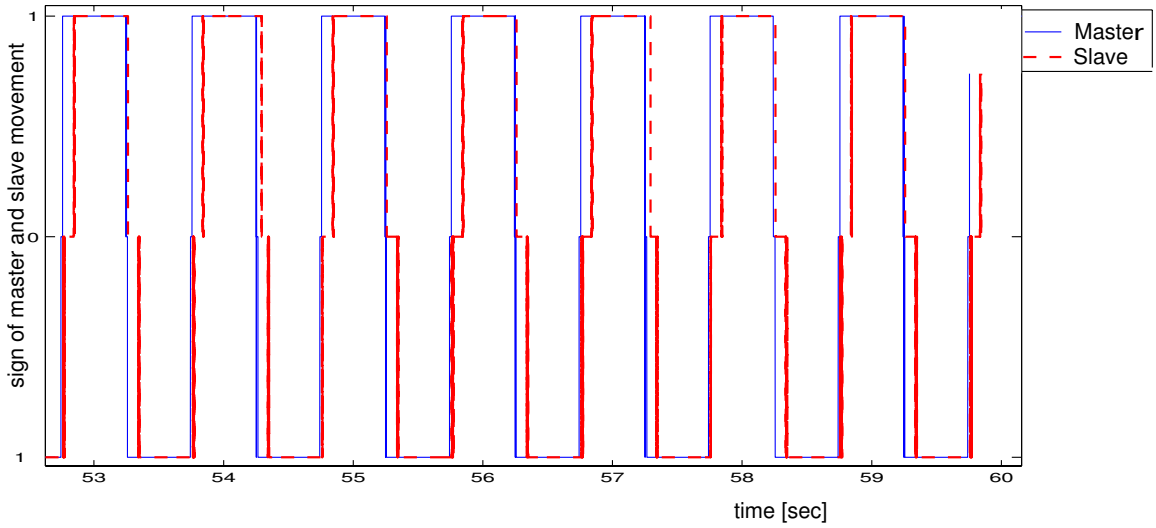


Figure 4.5: The monotonicity of the transmission is shown in this figure. The master velocity is represented by the solid line while the slave velocity is given by the dashed lines. It is seen that the master and slave velocities never have opposite signs at any one time.

monotonicity of the system where the master and slave velocities never have opposite signs. This suggests that the real system may have similar monotonicity.

The monotonicity and the cyclic, repetitive nature of the desired movement allows the use of iterative learning control (ILC) to improve the performance of the system. The iterative control aims at minimizing the slave tracking error in future sinus cycles by learning from the previous cycles.

A PD type ILC was implemented for the system. The ILC is used to learn the master reference signal. A conventional PD controller controls the master about this reference.

The iterative control can be mathematically formulated as follows. Let the sinus wave be divided into time blocks of N discrete control points in one sinus cycle. The numerical value of N in each time block depends on time period T and control frequency f: $N = Txf$.

$$\mathbf{u}_k = \begin{bmatrix} u(0) & u(1) & \dots & u(N-1) \end{bmatrix}^T$$

$$\mathbf{y}_k = \begin{bmatrix} y(1) & y(2) & \dots & y(N) \end{bmatrix}^T$$

$$\mathbf{r} = \begin{bmatrix} r(1) & r(2) & \dots & r(N) \end{bmatrix}^T$$

where $\mathbf{u}_k$ represents the master position reference vector and serves as the input to the k^{th} time block. $\mathbf{y}_k$ represents the slave position output from the k^{th} time block and $\mathbf{r}$ represents the slave position reference vector.

The PD ILC increments the master reference using the tracking error between the slave reference $\mathbf{r}$ and slave output $\mathbf{y}_k$. The tracking error for the k^{th} time-block is given as:

$$\mathbf{e'_k} = lowpassfilter(\mathbf{r} - \mathbf{y}_k) \tag{4.3.2}$$

The low pass filtering of the tracking error attenuates high frequency errors but gives a more stable learning. The cut-off frequency is selected heuristically so that there is stable learning with minimum neglect of errors.

The tracking error is then time shifted to account for the time delay in the transmission:

$$\mathbf{e_k}(1 : (N - N_d)) = \mathbf{e'_k}(N_d : N)$$

$$\mathbf{e_k}((N - N_d) : N) = \mathbf{e'_k}(1 : N_d)$$

where N_d represents the number of samples corresponding to the time delay. The time delay for $1Hz$ movement is seen from previous experiments to be around $30ms$. Therefore for simulations at a dynamic frequency of $2kHz$, $N_d = 60$.

The PD iterative learning of $\mathbf{u}_{k+1}$ for the subsequent time-block is given by

$$\mathbf{u_{k+1}} = \mathbf{u}_k + \gamma_p \, \mathbf{e_k} + \gamma_d \, \Delta \mathbf{e_k}$$

$$\Delta \mathbf{u_{k+1}} = \gamma_p \, \mathbf{e_k} + \gamma_d \, \Delta \mathbf{e_k} \tag{4.3.3}$$

where γ_p and γ_d represent the learning constants for the iterative learning. The learning factors are kept low compared to the online master control gains- P and D of chapter 4.1.

Proof of Convergence

This section shows the stability of the iterative learning and that the error converges to *zero* with learning. Assuming that the master side tracks the learnt reference perfectly, for the k^{th} time block the error element is given as

$$\mathbf{e_k} = \mathbf{r} - \mathbf{y_k} \tag{4.3.4}$$

Defining extended matrices to relate the input $\mathbf{u_k}$ and output $\mathbf{y_k}$ by a non linear function matrix $\mathbf{F_k}$,

$$\overline{\mathbf{y}}_\mathbf{k} = \mathbf{F_k} \, \overline{\mathbf{u}}_\mathbf{k} \tag{4.3.5}$$

where

$$\overline{\mathbf{y}}_\mathbf{k} = \begin{bmatrix} \mathbf{y_k} \\ 1 \end{bmatrix} \, , \overline{\mathbf{u}}_\mathbf{k} = \begin{bmatrix} \mathbf{u_k} \\ 1 \end{bmatrix} \quad \mathbf{F_k} = \begin{bmatrix} \mathbf{F_p}(\mathbf{u_k}) & \mathbf{F_s}(\mathbf{u_k}) \\ 0 & 1 \end{bmatrix}$$

$\mathbf{F_p}(\mathbf{u_k})$ is a NxN diagonal matrix which roughly gives the linear gains between the master and slave trajectories and can be represented by Fig. 3.4. $\mathbf{F_s}(\mathbf{u_k})$ has dimensions of $Nx1$ and represents the non-linear additive term due to static friction in the system.

The elements of $\mathbf{F_p}(\mathbf{u_k})$ depend on the frequency content of the master trajectory. As the desired movement is only $1Hz$ and the system itself is a low pass filtered, the elements of $\mathbf{F_p}(\mathbf{u_k})$ are therefore bounded:

$$-3dB < \|\mathbf{F_p}[\mathbf{i}, \mathbf{i}]\| < 12dB \ , \ 1 \leq i \leq N \tag{4.3.6}$$

where $\mathbf{F_p}[\mathbf{i}, \mathbf{i}]$ represent the diagonal elements of diagonal matrix $\mathbf{F_p}$. $-3dB$ and $12dB$ are the minimum and maximum gains corresponding to the cut-off and resonance frequency

of the system respectively (Fig. 3.4). Eqn. 4.3.6 corresponds to the monotonicity condition (Eqn. 4.3.1).

From 4.3.5 we get

$$\overline{\mathbf{y}}_{\mathbf{k-1}} = \mathbf{F}_{\mathbf{k-1}}\,\overline{\mathbf{u}}_{\mathbf{k-1}} \tag{4.3.7}$$

performing (4.3.5) - (4.3.7) and for small gains, assuming $\mathbf{F_k} \approx \mathbf{F_{k-1}}$:

$$\Delta\overline{\mathbf{y}}_{\mathbf{k}} = \mathbf{F_k}\,\overline{\mathbf{u}}_{\mathbf{k}} - \mathbf{F_{k-1}}\,\overline{\mathbf{u}}_{\mathbf{k-1}} = \begin{bmatrix} \mathbf{F_p}(\mathbf{u_k}) & 0 \\ 0 & 1 \end{bmatrix} \Delta\overline{\mathbf{u}}_{\mathbf{k}} = \mathbf{G_k}\,\Delta\overline{\mathbf{u}}_{\mathbf{k}} \tag{4.3.8}$$

Due to the monotonicity of the plant, all the elements of the diagonal matrix $\mathbf{G_k}$ should be positive (Eqn. 4.3.1). Further, because the elements of $\mathbf{F_p}(\mathbf{u_k})$ are bounded (Eqn. 4.3.6), $\mathbf{G_k}$ is also bounded.

Defining extended vectors

$$\overline{\mathbf{e}}_{\mathbf{k}} = \begin{bmatrix} \mathbf{e_k} \\ 1 \end{bmatrix} \;, \overline{\mathbf{r}} = \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix}$$

Eqn. 4.3.4 can be rewritten as

$$\overline{\mathbf{e}}_{\mathbf{k}} = \overline{\mathbf{r}} - \overline{\mathbf{y}}_{\mathbf{k}} \tag{4.3.9}$$

and

$$\overline{\mathbf{e}}_{\mathbf{k-1}} = \overline{\mathbf{r}} - \overline{\mathbf{y}}_{\mathbf{k-1}} \tag{4.3.10}$$

Forming a difference equation across trials (time blocks) by subtracting Eqn. 4.3.9 from Eqn. 4.3.10 , and substituting from 4.3.8

$$\overline{\mathbf{e}}_{\mathbf{k}} - \overline{\mathbf{e}}_{\mathbf{k-1}} = -\Delta\overline{\mathbf{y}}_{\mathbf{k}} = -\mathbf{G_k}\,\Delta\overline{\mathbf{u}}_{\mathbf{k}}$$

replacing from Eqn. 4.3.3,

$$\overline{\mathbf{e}}_{\mathbf{k}} - \overline{\mathbf{e}}_{\mathbf{k-1}} = -\mathbf{G_k}\left(\gamma_p\,\overline{\mathbf{e}}_{\mathbf{k-1}} + \gamma_d\left(\overline{\mathbf{e}}_{\mathbf{k-1}} - \overline{\mathbf{e}}_{\mathbf{k-2}}\right)\right)$$

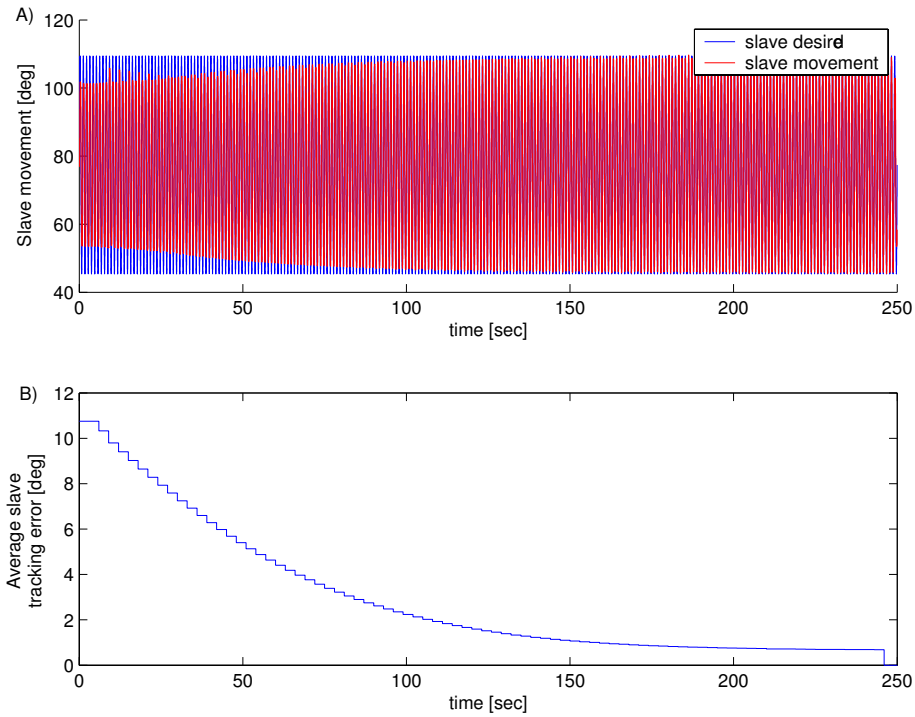


Figure 4.6: A) shows the desired movement (in blue) and the slave movement (in red) during learning. With learning, the slave tracks the desired trajectory better. B) shows the decrease of average tracking error. Learning time blocks of $3sec$ were used during the learning and the error was averaged over each time-block.

or

$$\bar{\mathbf{e}}_{\mathbf{k}} = (I - (\gamma_p + \gamma_d)\,\mathbf{G}_{\mathbf{k}})\,\bar{\mathbf{e}}_{\mathbf{k-1}} + \gamma_d\,\mathbf{G}_{\mathbf{k}}\,\bar{\mathbf{e}}_{\mathbf{k-2}} \tag{4.3.11}$$

where I is a NxN identity matrix. For small gains and $\gamma_d << \gamma_p$, this can be simplified to

$$\bar{\mathbf{e}}_{\mathbf{k}} = (I - (\gamma_p + \gamma_d)\,\mathbf{G}_{\mathbf{k}})\,\bar{\mathbf{e}}_{\mathbf{k-1}} \tag{4.3.12}$$

which is a geometric progression. As the the elements in diagonal matrix $\mathbf{G_k}$ are positive and bounded, positive values γ_p, γ_d can be found such that $(\mathbf{I} - (\gamma_p + \gamma_d)\,\mathbf{G_k})$ is negative definite ensuring that the error converges to zero.

The iterative learning algorithm was implemented on the model of the haptic device described in chapter 3. Fig 4.6A. shows the change in the slave movement with progression

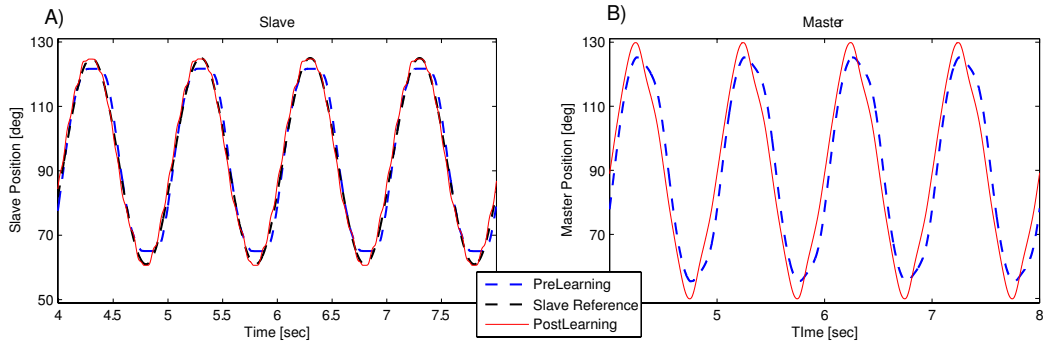


Figure 4.7: Comparison of movements before and after learning, A) shows the slave movement in the two cases. The slave reference is shown with dashed lines. The slave is found to track the desired well after learning. B) shows the master movement as in chapter 4.1 and the master movement after learning.

of learning. Fig. 4.6B shows the average tracking error on the slave side. The tracking error was averaged over $3s$ corresponding to the learning time blocks. The tracking error is found to decrease with learning from an average of about 11° to 1° in about $250s$ corresponding to 83 time blocks.

After learning the learnt reference was used to control the modelled plant and to test the learning. Fig. 4.7A represents the slave movement with the control of chapter 4.1 and the movement learnt with iterative control. The shape of the slave movement is seen to track the reference well after learning. Sticking is still prevalent at the peaks but the duration is reduced considerably. Fig. 4.7B. compares the master movement before and after learning. It can be seen that the master movement learns a higher peak in order to counter the sticking due to the static friction.

Chapter 5

Investigation of Cable Transmission

This chapter analyzes a transmission driven by magnetically inert cables as an alternative MR compatible transmission and compares it with the hydrostatic transmission.

The major disadvantage of the cable system is its transmission in-flexibility. The cable system requires stiff support structures for the cables while the pipe of the hydrostatic system can be arranged as required to transmit power around obstacles and in any orientation of the slave interface.

A test-bed was developed to analyze the cable transmission. A novel cable routing design brings flexibility to the cable transmission, by eliminating the need of fixed support structures for the cables. Section 5.1 describes the design, which was developed in collaboration with Roger Gassert and Dominique Chapuis at EPFL and manufactured there. Section 5.2 analyzes results of experiments performed at EPFL and compares them with experiments performed with the hydrostatic transmission.

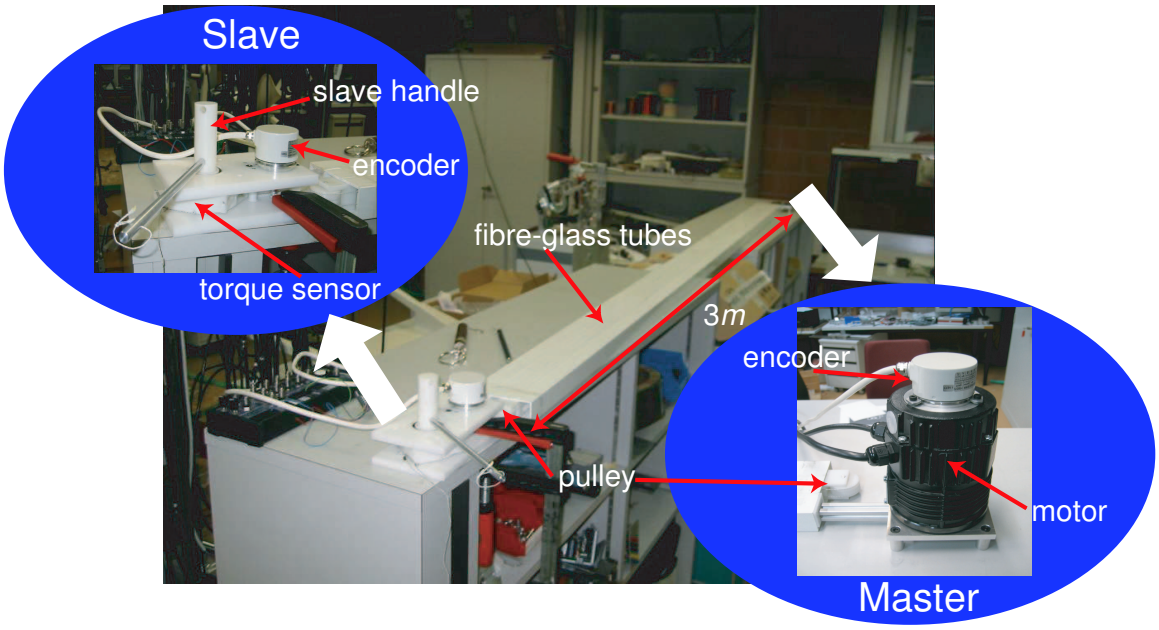


Figure 5.1: Cable transmission test-bed consisting of three major parts- Master, Slave and the fibre-glass tubes, realized at EPFL.

5.1 Mechanical Design

To avoid rigid structures to support and route the cables, we propose to pass the cables through fibre-glass tubular sections. Fibre-glass is a lightweight, high-strength, MR compatible material. Sections of these tubes can be joined by flexible joints to enable simple installation/removal and routing of the transmission inside the fMRI room. The cables are kept under tension and routed through these tubes using pulleys at the section joints. The cable transmission test-bed developed with this idea is shown in Fig. 5.1. The test-bed was developed for power transmission over a length of $9m$, corresponding to the length requirement in a MR scanner room. The test-bed has three major sections: the master, the transmission and the slave.

Fig.5.2A shows the PRO-E drawing of the master section. The master consists of a (Mavilor MS-22) DC motor and a (Hengstler,O 522672, $5000 lines/rev$) encoder. A $1mm$

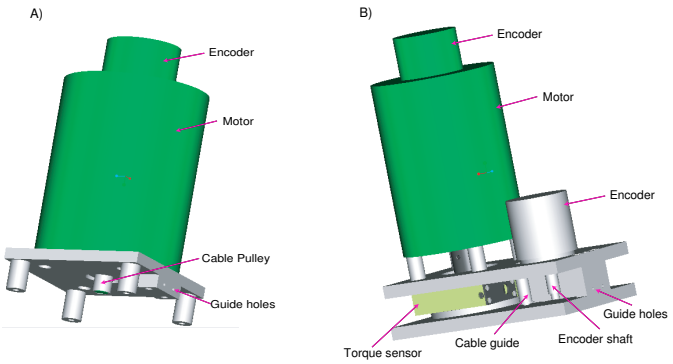


Figure 5.2: Pro E diagram of the A) master side of the cable transmission test-bed. The cable is wrapped around the master pulley and passes through the guide holes into the fibre glass tube. B) Slave side. Note that the current prototype does not involve a motor, which would enable a more systematic characterization of the transmission.

diameter, $18m$ long kevlar cable is used for the transmission. The cable is wrapped onto a pulley fixed to the motor axis and passes into the transmission tubes via guide holes on the master. The master motor is controlled by xPC target real-time system through a National Instruments data acquisition card (PCI 6014), as described in Chapter 4. The encoders on the master, as well as the slave are monitored using a APCI-1710 encoder board. The programming of the real-time system is done in Matlab and Simulink, from a host computer running on Windows.

The cables from the master pass through three fibre glass tubular sections arranged in series. Each tube section is $3m$ long and has a square cross section of side length $40mm$ and a wall thickness of $3mm$. The cables are routed through the tubes with pulleys at the section joints (Fig.5.1). In the actual plant the sections can be joined using flexible pipes that can have relative angular displacement, providing some flexibility to the transmission. The pipes can also help in routing the cables through the sections.

The Slave side of the test-bed is manufactured from poly-oxy-methylene (POM) (Fig.5.2B). The cable is connected to the slave end effector through a (Hengstler,O 522672, $5000 pulses/rev$)

encoder and a torque sensor. A motor can be added on the slave end effector to provide controlled loads to test the system.

The torque sensor is MR compatible and is based on a crossed strip mechanism. Optical sensors detect the deflection of a polymer probe and transmit the signal through optical fibers to the sensor circuit [18, 8]. The intensity of the reflected light beam is measured to derive the applied torque on the slave. The torque sensor signal and the encoder are monitored by the PCI-6014 card and the encoder card on the master side respectively.

The test bed enables us to control the motor on the master and slave side and collect position and velocity data of the master and slave side as well as the torque output on the slave side. This data were used to analyze the system and compare it with the hydrostatic transmission.

5.2 Data Analysis

5.2.1 Stiffness

Fig.5.3 shows the plot of extension of the cable with varying load. The extension was measured by blocking the master while applying a load on the slave side. The force on the cable and the cable extension were measured using a dynamometer from a ruler respectively. The extension is found to increase linearly with force except at low loads. The initial tension in the cable ensures that the working range is in the linear section of the curve. The slope of the force/extension curve determines stiffness $K_{cable} = 3.725x10^4 N/m$.

Another property observed with the cable is the change in stiffness when subjected to loading cycles. Fig. 5.4 shows the variation of load vs cable extension for two back to back trails with similar strain rates. A modified Zwick 1474 testing machine was used for this purpose. The cable piece was fixed between two supports and then extended by moving

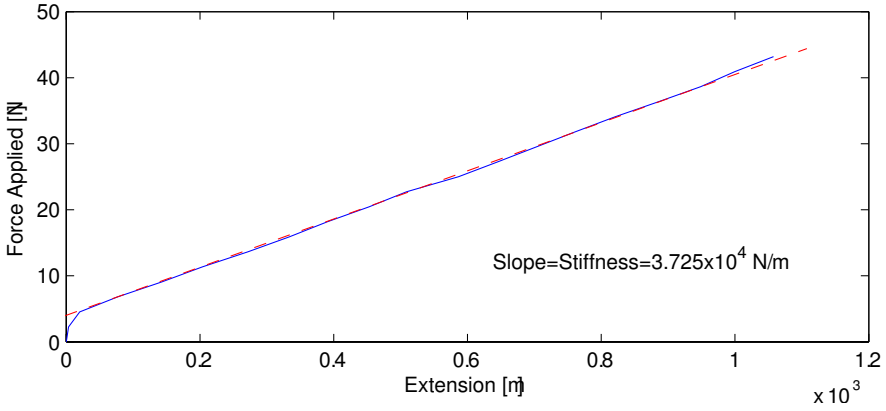


Figure 5.3: Extension against force applied measured for the $9m$ long cable. Except for lower forces, the extension is essentially linear. The dashed line is the least-square fit of the data in the interval $0.02x10^{-3} - 1.0x10^{-3}m$ corresponding to the linear part of this function. The initial tension in the cable ensures that the working range is in the linear section of the curve.

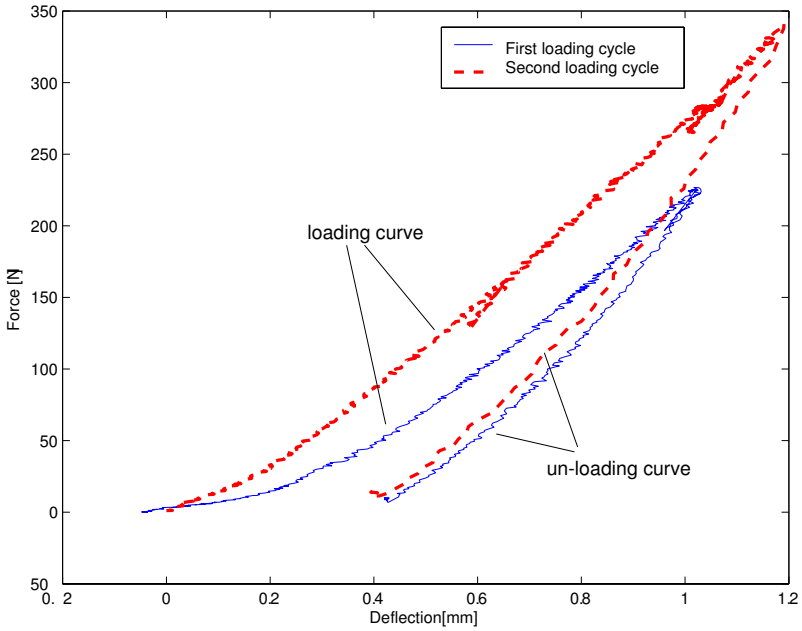


Figure 5.4: The figure shows the load-extension curves for a $20cm$ length of kevlar cable for two back to back loading-unloading cycles. The stiffness is larger during the second loading, probably due to the allignment of fibres along the direction of the force. The system also shows some hysteresis when the loading is decreased.

one of the supports at a constant velocity using a high reduction drive. The force during the extension was measured by a force sensor mounted on the fixed support. The stiffness of the cables is found to increase for the second trial. The left curve for each plot shows the extension during loading and the right one shows the unloading curve. The extension is seen to be different while loading and unloading of the system, showing hysteresis in the system.

5.2.2 Static Friction

Fig. 5.5 compares the static friction in the hydrostatic and cable interfaces. The interface was moved in a very low frequency sinusoidal trajectory ($0.008Hz$) with proportional control, at which the motor torque was assumed to represent the static friction in the system. Static friction is plotted against slave position to show how the magnitude varies with position. The positive and negative values show static friction corresponding to a positive and negative velocity of the slave respectively. Static friction in the hydrostatic interface is found to be of the order of $0.9Nm$ and almost twice as high as in the cable interface. In both the interfaces friction varies with position. However the variation is more prominent with the present hydrostatic transmission (this may be improved in the second generation hydrostatic transmission using brass pistons as in the second interface developed for the European Touch Hapsys project, though we have not yet tested it systematically). It is also seen that static friction magnitude is different even at the same position for different velocity directions. The average static friction for positive velocities with the hydrostatic transmission is around $0.9Nm$ while it is around $1.0Nm$ for negative velocities. The average value of static friction is around $0.5Nm$ and $0.45Nm$ for the positive and negative velocities respectively with the cable interface.

Fig. 5.6 shows the variation of average static friction with the magnitude of initial

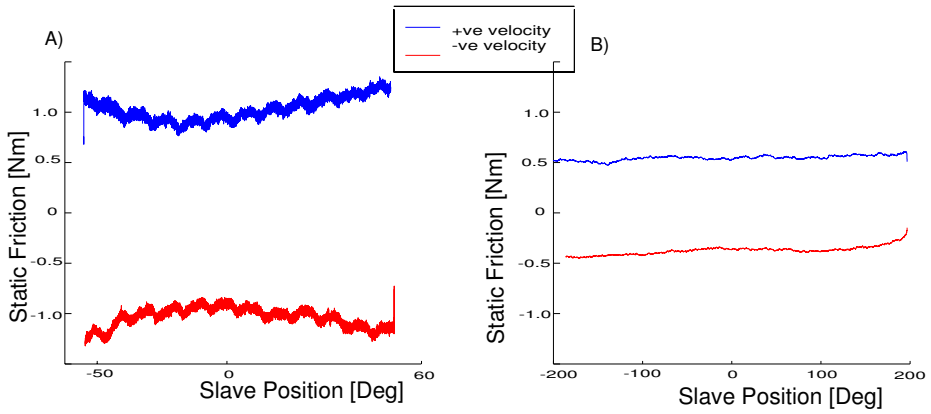


Figure 5.5: A)Static friction plotted against the slave position for the ATR system with hydrostatic transmission. B) Static Friction plotted against slave position for the cable transmission. The positive and negative friction values indicate the friction for positive and negative slave movements respectively.

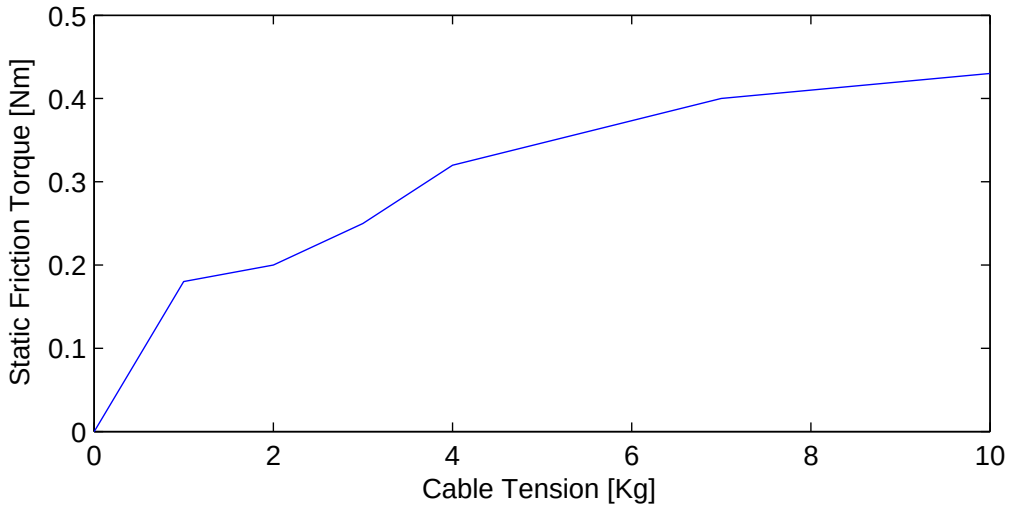


Figure 5.6: Average static friction against cable tension. The friction in the system increases with cable tension. It seems to be converging at high tensions. However higher cable tension limits the magnitude of force transmission possible.

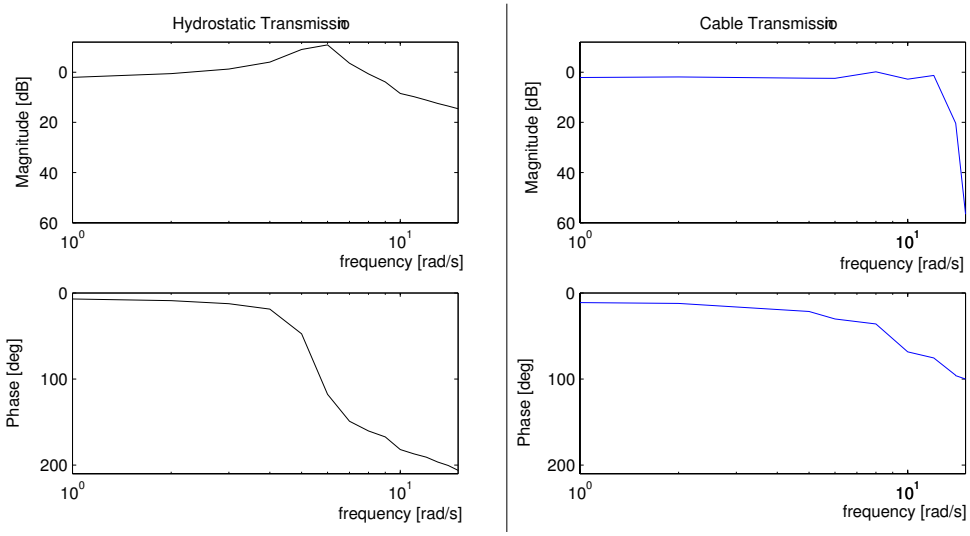


Figure 5.7: Bode plots of the hydrostatic and cable transmission systems over a frequency range of 0.2 to 15Hz. The resonance frequencies for the two transmissions are similar.

tension. Friction increases with the initial tension and seems to slowly converge to a constant value as the tension is increased.

5.2.3 Master Slave Trajectories

Fig. 5.7 compares the frequency response of the cable system with that of the hydrostatic transmission. The frequency response data represents the input and output across the transmission between the master and the slave. While the mechanical ratio between the master and slave in the hydrostatic transmission is ≈ 1, this value for the cable transmission is high (≈ 6) due to the difference in the pulley size on the master and slave side. For the sake of comparison with the hydrostatic transmission, the gain is normalized to remove this mechanical ratio in the cable system when the bode diagram is plotted. It is observed that the resonance frequency of the cable system is similar to the hydrostatic system and around $8Hz$. The cut-off frequencies for both the systems are around $12Hz$. However unlike the hydrostatic system, the gain value for the cable system remains same till after resonance,

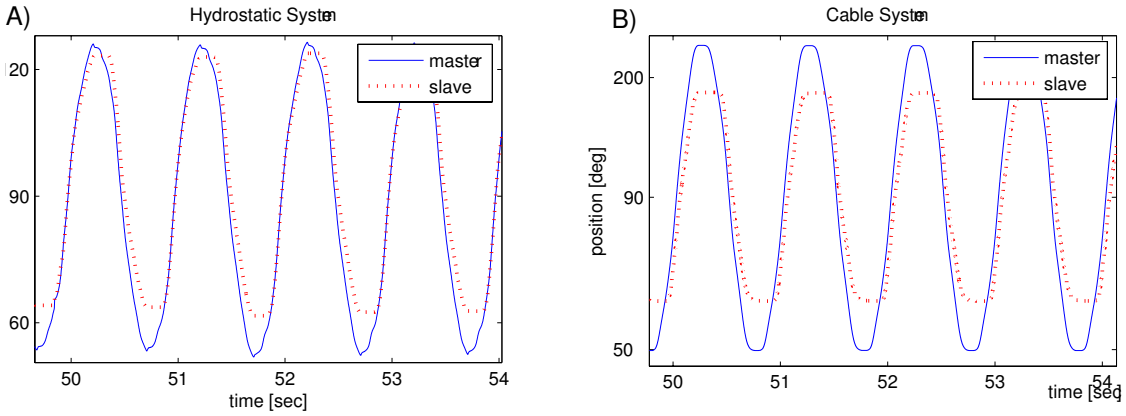


Figure 5.8: The figure shows movements of frequency $1Hz$, performed by systems with the hydrostatic transmission (A) and cable transmission (B). In both cases the slave movement is similar but the movements with the hydrostatic transmission have a visibly shorter flat region.

when it falls sharply.

Fig. 5.8 shows the master and slave movements at $1Hz$ frequency, in the case of the cable and hydrostatic systems. The master trajectory is represented by a solid line while the slave trajectory is shown with dashed lines. The control system for both the systems were similar during these experiments. The phase lag is similar in both the transmissions , however the sticking of the slave is visibly more in the case of the cable transmission.

5.3 Discussion

The initial tension of the cable in the interface is found to be an important factor to determine the dynamics of the system. The major source of friction in the system is from the bearing surfaces of the pulleys supporting the cables. With increase in initial tension the bearing pressure increases, thus increasing the average friction magnitude (Fig. 5.6). In addition with the increase in initial tension, the working tension nears the yielding tension of the cable. Theoretically, this limits the range of force transmission permissible with the

system and increases the fatigue wear of the wires.

It is desired that the transmission is as stiff as possible to counter the friction on the slave side. The large distance between the cable supports ($3m$, in the present test-bed) leads to a large slack in the cable due to its self weight, which reduces the stiffness and introduces an undesirable response delay in the transmission. The slack can be decreased by increasing the initial tension.

Thus it may be inferred that a low initial tension is desired to keep the friction in the system low and increase the force range. However a high initial tension ensures better stiffness and low response delay. Thus while setting the initial tension to the cable a compromise has to be reached between the friction, stiffness and response delay in the system. Unless the initial tension is kept very high, the slack in the cables introduces more compliance in the system, in addition to the compliance of the cable. This drawback is absent from the hydrostatic transmission.

Another property observed with the cable system is the variation of stiffness with loading cycles. It is seen that the stiffness is higher when the cable is loaded subsequently after unloading. This is probably due to the better aligning of fibres making up the cable in the direction of the force, which increases the stiffness of the cable. This may be seen as a drawback compared to a hydrostatic system where the mechanical parameters are more stable.

Even though the transmission is non-linear, the frequency response gives a good indication of the behaviour of the system by approximating it as a linear system. The bode plot of the two systems are found to have similar resonance and cut-off frequencies. The static friction in the cable system is lower than in the hydrostatic transmission but there is still notable sticking in the slave movement. This is probably due to the additional compliance

in the system due to the slack in the cables. Thus the present cable transmission does not seem to give any prominent improvement in the dynamics of the system over that with the hydrostatic transmission. However it introduces inflexibility in the routing of the transmission due to the cable supports, unlike in the hydrostatic transmission, where the pipes can be arranged as required to transmit power in any orientation of the slave interface.

A potentially major advantage of the cable transmission over the hydrostatic transmission may be that the stiffness of the cable transmission can be increased with minimal modification of the system, by using larger diameter or multiple cables. However the relative change in the friction in the system with multiple cables remains to be examined.

Chapter 6

Conclusion

Various kinds of actuation principles for MR compatible haptic interfaces were analyzed in [3] which suggests that the hydrostatic and cable transmissions are currently the best potential actuation principles.

A master-slave system with a hydrostatic transmission was developed to transmit force and motion from the control room to the interface near the scanner, using a non-conducting, non-ferromagnetic slave. A low cost real-time computer system allowed hardware control and acquisition of eight channels of data at $2kHz$. A support program allows easy use of the system for running of experiments in the MR scanner. To understand the dynamics of a hydrostatic transmission better, a nonlinear model of the transmission was developed and validated, which suggested performances suitable for haptic applications.

The model helped in the development of force and position control algorithms which allow precise and smooth movement control over a broad range of speed and force/torque. Experiments showed that the system with the hydrostatic transmission performs well within an MR scanner and allows image acquisition during motion without disturbances. To our knowledge, this interface on which the novel control was implemented is the first MRI/fMRI compatible haptic device. It will be used to investigate the brain mechanisms of motor

learning at ATR International, Japan.

Preliminary analysis of cable transmission as a possible MR-compatible actuation system, gave insight into the advantages and disadvantages of the system. Initial results indicate that the dynamics of the cable transmission are similar to that of a hydrostatic transmission. Even though the proposed design increases the transmission flexibility of a normally rigid cable transmission, it is still less practical than a hydrostatic system, where the transmission pipes can be arranged as required, to transmit power around obstacles and in any orientation of the slave interface. Therefore we believe that the hydrostatic transmission is currently the best solution for MR compatible transmission. However further tests are required with the cable interface, in particular to investigate the effect of multiple cables on friction and stiffness.

This study gave a good insight into the dynamics and control schemes for actuators for MR compatible master-slave haptic devices. Future studies will be done to test new designs and control ideas for such actuators:

- Chapter 4.3 described the learning algorithm which was found to work well in simulation. This algorithm will be implemented and tested on the real plant.
- Further tests will be done with the cable interface to analyze particularly, the influence of the use of multiple cables in the cable transmission. Though it is intuitively known that it will increase the stiffness of the system, it remains to be seen how friction will be influenced by using multiple cables.
- A currently tested idea, consists of developing MR compatible passive interfaces where the force fields are generated using gravity and springs as actuators [9]. This would simplify the system by avoiding the need to transmit power inside the MR scanner room.

Bibliography

[1] www.forcedimension.com.

[2] E Burdet, A Codourey, and L Rey. Experimental evaluation of nonlinear adaptive controllers. *IEEE Control Systems Magazine*, 18(2):39–47, 1998.

[3] E Burdet, R Gassert, G Gowrishankar, D Chapuis, and H Bleuler. fmri compatible haptic interfaces. In *9th International Symposium on Experimental Robotics ISER*, June 2004.

[4] E Burdet, M Honegger, and A Codourey. Controllers with desired dynamics and their implementation on a 6dof parallel manipulator. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems*, pages 39–45, 2000.

[5] E Burdet, R Osu, DW Franklin, TE Milner, and M Kawato. The central nervous system stabilizes unstable dynamics by learning optimal impedance. *Nature*, (414):446–449, 2001.

[6] C Canudas de Wit, H Olsson, KJ Åström, and P Lischinsky. A new model for control of systems with friction. *IEEE Trans. Aut. Cntrl.*, 40(3):419–425, March 1995.

[7] C CanudasDeWit, B Siciliano, and G Bastin, editors. *Theory of Robot Control*. Springer, 1996.

[8] D Chapuis, R Gassert, L Sache, E Burdet, and H Bleuler. Design of a simple mri/fmri compatible force/torque sensor. In *IEEE International Conference on Intelligent Robotic Systems IROS 2004*, September 2004.

[9] L Dovat. Passive interface to investigate human motor control using functional magnetic resonance imaging (fmri). *Masters Thesis, EPF Lausanne (submitted)*, 2004.

[10] G Ganesh, R Gassert, and E Burdet. Feasibility study for a low cost real time control system for an atr fmri compatible haptic interface. *Internal Report, ATR International, Kyoto, Japan*, 2004.

[11] H Gomi and M Kawato. Equilibrium-point control hypothesis examined by measured arm-stiffness during multi-joint movement. *Science*, 272:117–120, 1996.

[12] DJ Heeger and D Ress. What does the fmri tell us about neuronal activity? *Nature Reviews*, 3:142–151, 2002.

[13] M Honegger and P Corke. Model-based control of hydraulically actuated manipulators. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 2553–2559, 2001.

[14] S Imai, Y Saito, T Tajima, and K Ohnishi. Development of hydraulic bilateral-servo actuator for powered orthosis. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 2, pages 1237–1242, 2000.

[15] HI Krebs et al. Increasing productivity and quality of care: Robot-aided neurorehabilitation. *Journal of Rehabilitation Research and Development*, 37(6), 2000.

[16] P Lischinsky, C Canudas de Wit, and G Morel. Friction compensation for an hydraulic industrial robot. *IEEE Cntrl. Sys.*, 19(1):25–33, February 1999.

[17] TE Milner and C Cloutier. Compensation for mechanically unstable loading in voluntary wrist movement. *Exp Brain Res*, 94(3):522–532, 1993.

[18] R Moser, R Gassert, E Burdet, L Sache, HR Woodtli, J Erni, W Maeder, and H Bleuler. An mr compatible robot technology. In *IEEE International Conference on Robotics and Automation ICRA 03*, pages 670–675, September 2003.

[19] K Ogata. *Modern Control Engineering*. Prentice-Hall, second edition, 1998.

[20] H Olsson, KJ Åström, C Canudas de Wit, M Gäfvert, and P Lischinsky. Friction models and friction compensation. *Internal Report, Laboratoire díAutomatique de Grenoble*, 1997.

[21] R Shadmehr and HH Holcomb. Neural correlates of motor memory consolidation. *Science*, 277(5327):821–825, 1997.

[22] DD Stark and WG Bradley. *Magnetic Resonance Imaging*. St. Louis, 1988.

Appendix - Matlab Code and Simulink Modules

Matlab Code

APPENDIX

MATLAB code for Force- Matching Experiment

```
function varargout = Experiment(varargin)
```

PART A

```
% EXPERIMENT M-file for Experiment.fig
%      EXPERIMENT, by itself, creates a new EXPERIMENT or raises the existing
%      singleton*.
%
%      H = EXPERIMENT returns the handle to a new EXPERIMENT or the handle to
%      the existing singleton*.
%
%      EXPERIMENT('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in EXPERIMENT.M with the given input arguments.
%
%      EXPERIMENT('Property','Value',...) creates a new EXPERIMENT or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Experiment_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Experiment_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Experiment

% Last Modified by GUIDE v2.5 27-May-2004 16:37:58

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Experiment_OpeningFcn, ...
                   'gui_OutputFcn',  @Experiment_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

PART B

```
% --- Executes just before Experiment is made visible.
function Experiment_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
tg=xpc;  % defines a xpc object (data structure ) in  the workspace
load(tg,'STOPMOTOR');  %loads program to set Output to zero
tg.start;   % starts program run on target PC
pause(1);
tg.stop;    % Stops Program Rum on Target
unload(tg);
clear CENTER SETS REST_TIME SESSION_TIME
clear FILE_NAME FREQ AMP SPRINGC
set(findobj('tag','start_button'),'Enable','off'); % disable start, init buttons
set(findobj('tag','init_button'),'Enable','off');
bar(20,'w');

function varargout = Experiment_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;


% --- Executes on button press in init_button.
function init_button_Callback(hObject, eventdata, handles)
% message displayed on User screen and Subject screen
set(findobj('tag','comment'),'string','Interface is now initializing. Please Do NOT resist the movement');
set(findobj('tag','notes'),'string','Interface is now initializing. Please wait for the green light');
pause(0.01);
global SESSION
SESSION=0;
global CENTER
CENTER=str2double(get(findobj('tag','center_value'),'string'));
tg=xpc;
load(tg,'INITIALIZE2F');  % initialization module loading
tg.P32=1.0; % Enables motor output in the initialization program
tg.start;
done=0;
while(done==0)
 done=tg.S40;  % S40 (signal 40), is activated when the interface initialization is complete
end;
tg.P32=0.0; % motor disable
tg.stop;
unload(tg);
global INITIALIZED
INITIALIZED=1;
set(findobj('tag','start_button'),'Enable','on');  % enable buttons
set(findobj('tag','init_led'),'backgroundcolor','g');
set(findobj('tag','comment'),'string','Initialization completed') % display notes to subject and experiment instructor
```

```
set(findobj('tag','notes'),'string','Initialization completed. Make Parameter changes if required and "Build" the program')

% --- Executes on button press in build_button.
function build_button_Callback(hObject, eventdata, handles)
global CENTER SETS REST_TIME SESSION_TIME
global FILE_NAME FREQ AMP SPRINGC
%display notes to user and subject
set(findobj('tag','comment'),'string','The experiment will start soon.Start with the movements when the LED turns green.Do not apply any force when the LED is red ');
set(findobj('tag','notes'),'string','Building program.The "Start" button will be activated when building is completed.The MATLAB command window may be openend during this process.');
% read in experiment parameters
CENTER=str2double(get(findobj('tag','center_value'),'string'));
SETS=str2double(get(findobj('tag','sets_value'),'string'));
REST_TIME=str2double(get(findobj('tag','rest_value'),'string'));
SESSION_TIME=str2double(get(findobj('tag','session_value'),'string'));
FREQ=str2double(get(findobj('tag','freq_value'),'string'));
AMP=str2double(get(findobj('tag','amp_value'),'string'));
SPRINGC=str2double(get(findobj('tag','spring_value'),'string'));
FILE_NAME=get(findobj('tag','filename'),'string');
% build models in order to take in parameters
rtwbuild('IMPEDENCE');
rtwbuild('RYTHM');
rtwbuild('INITIALIZE2F');
% enable buttons
set(findobj('tag','init_button'),'Enable','on');
set(findobj('tag','build_button'),'Enable','off');
set(findobj('tag','notes'),'string','Program build completed');
```

PART C

```
% --- Executes on button press in start_button.
%This starts the execution of the experiment

function start_button_Callback(hObject, eventdata, handles)
set(findobj('tag','start_button'),'Enable','off');
disp=zeros(40,1);
global FILE_NAME SETS REST_TIME SESSION
SESSION=1;
index=1;
START_ON=1;
% read in file name fmri startup time and sets
FILE_NAME=get(findobj('tag','filename'),'string');
SETS=str2double(get(findobj('tag','sets_value'),'string'));
FMRI_START=str2double(get(findobj('tag','fmri_start_value'),'string'));
SESSION=1;
START_ON=1;
bar(20,'w'); % clears display

meansum=0;
tg=xpc;
oldtime=str2double(tg.sessiontime);
set(findobj('tag','comment'),'string','');
```

```
total_mean=0.0001; % initialization of mean value

for num=1:2*SETS
   unload (tg);
if(SESSION==1)
   load(tg,'RYTHM');
end;
if(SESSION==-1)
   load(tg,'IMPEDENCE');
end;
if(num==1)    % waiting for fmri triggers
set(findobj('tag','notes'),'string','Waiting for fMRI trigger');
   while(tg.s90>2.0)    %------fmri trigger
   %wait;
   end;
   oldtime=str2double(tg.sessiontime);
set(findobj('tag','notes'),'string','fMRI trigger recieved');
end;

tg.stoptime=35.00;
newtime=str2double(tg.sessiontime);
if (num==1)
  pause(FMRI_START-newtime+oldtime); %--time SYNCHRONIZATION
else
index=1;
% display control during rest phase (display of bar movement pattern  as recorded in exp session)
while ((newtime-oldtime)<REST_TIME)
  if (disp(index)<19)
   bar(disp(index));
else
   bar(19);
end;
 index=index+1;
 pause(1);
 newtime=str2double(tg.sessiontime);
end;
index=1;
end;
%_____REST PHASE ENDS

%------IMPEDENCE SESSION
if(SESSION==-1)

tg.P23=1.0;%-------motor enable switch for impedence module
   tg.start;
   set(findobj('tag','pushbutton5'),'backgroundcolor','g'); % change LED color
   set(findobj('tag','notes'),'string','Experiment in progress.   SESSION--IMPEDENCE');
begintime=str2double(tg.sessiontime);
while(tg.s72==0)    %-------s72 activated when session finishes
   pause(1);
   temp=tg.s87;  %---s87 carries the summation of the force signal
   temptime=str2double(tg.sessiontime);
   meansum=temp/(temptime-begintime);
   % display control of subject display
   if(meansum>=2*total_mean)
```

```
      bar(19);
   else
      bar(meansum*10/total_mean);
   end;
   disp(index)=meansum*10/total_mean ; % stores value for redisplay during rest phase
   index=index+1;
end;
set(findobj('tag','pushbutton5'),'backgroundcolor','r'); % change LED color
pause(.01);
while(tg.s85==0) %-------done
   %wait
end;
tg.stop;
tg.P23=0.0; % disable motor
end;

% RYTHEM SESSION
if(SESSION==1)
tg.P56=1.0; %---enable motor for spring module
   tg.start;
   set(findobj('tag','pushbutton5'),'backgroundcolor','g');
   set(findobj('tag','notes'),'string','Experiment in progress.  SESSION--RYTHEM');
begintime=str2double(tg.sessiontime);
while(tg.s81==0) % while session not finished.
    pause(1);
    temp=tg.s88;  %----signal 88 carries the summation of the force sensor voltage
    temptime=str2double(tg.sessiontime);
    meansum=temp/(temptime-begintime);
   % display control
    if (num>1)
      if(meansum>=2*total_mean)
       bar(19);
   else
      bar(meansum*10/total_mean); % scaling for display
   end;
   end;
   disp(index)=meansum*10/total_mean; % mean stored for redisplay during rest phase
   index=index+1;
end;
% change LED color
set(findobj('tag','pushbutton5'),'backgroundcolor','r');
pause(0.01);
while(tg.s89==0)
   %wait
end;
tg.stop;
tg.P56=0.0; %disable motor
end;

%--------REST SESSION STARTS
oldtime=str2double(tg.sessiontime);

set(findobj('tag','notes'),'string','Experiment in progress.   SESSION--REST');
pause(0.01);
%----data transfer--------
output=tg.output;
```

```
time=tg.time;
s=[FILE_NAME '[' int2str(num) ']'];
save(s,'output','time');
%--------------------
total_mean=meansum; % recalculation of mean
SESSION=SESSION*(-1);
end;

% end game
set(findobj('tag','start_button'),'Enable','on');
set(findobj('tag','comment'),'string','The experiment is now over.Wanna have more ??!!')
set(findobj('tag','notes'),'string','The experiment is now over.Wanna have more ??!!')

% ---end of  START button code
```

PART D

```
% --- Executes on button press in stop_button.
function stop_button_Callback(hObject, eventdata, handles)
global SESSION
% set motor enable to zero in module depending on the module currently running
if (SESSION==0)
   tg.P32=0.0;
end;
if (SESSION==1)
   tg.P56=0.0;
end;
if (SESSION==-1)
   tg.P23=0.0;
end;
set(findobj('tag','start_button'),'Enable','on'); % enable start button for restart



% --- Executes on button press in exit_button.
function exit_button_Callback(hObject, eventdata, handles)
global SESSION
% set motor enable to zero in module depending on the module currently running
if (SESSION==0)
   tg.P32=0.0;
end;
if (SESSION==1)
   tg.P56=0.0;
end;
if (SESSION==-1)
   tg.P23=0.0;
end;
exit; % exit matlab

% --- Executes on button press in refresh_button. Runs xpctest and refreshes connection between host and
Target
function refresh_button_Callback(hObject, eventdata, handles)
tg=xpc;
set(findobj('tag','start_button'),'Enable','off');
set(findobj('tag','build_button'),'Enable','off');
set(findobj('tag','stop_button'),'Enable','off');
```

```
set(findobj('tag','exit_button'),'Enable','off');
set(findobj('tag','init_button'),'Enable','off');
set(findobj('tag','refresh_button'),'Enable','off');
set(findobj('tag','notes'),'string','The connection is now being refreshed.This will take some time.Please refrain from using the computer during this time.Some MATLAB windows may be openend during this process.'); % notes display for user and subject
xpctest;
pause(10);
set(findobj('tag','stop_button'),'Enable','on');
set(findobj('tag','exit_button'),'Enable','on');
set(findobj('tag','init_button'),'Enable','on');
set(findobj('tag','notes'),'string','Connection Refreshed.You can now initialize the interface.If there is still a problem,please check the LAN connections to the computers. ');
```

PART E

```
%%% rest is supporting program.Do not edit or delete.
% the following code functions  are required  as part of the API even though many of them  are empty and do not carry any execution code .


% --- Executes during object creation, after setting all properties.
function sets_value_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes during session value creation, after setting all properties.
function session_value_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes during spring creation, after setting all properties.
function spring_value_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes during amp value creation, after setting all properties.
function amp_value_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
% --- Executes during ferq value creation, after setting all properties.
function freq_value_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes during filename creation, after setting all properties.
function filename_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes during center value creation, after setting all properties.
function center_value_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes during rest value creation, after setting all properties.
function rest_value_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes during stop button creation, after setting all properties.
function stop_button_CreateFcn(hObject, eventdata, handles)
% --- Executes during init button creation, after setting all properties.
function init_button_CreateFcn(hObject, eventdata, handles)
% --- Executes during start button creation, after setting all properties.
function start_button_CreateFcn(hObject, eventdata, handles)
% --- Executes on led press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)

function sets_value_Callback(hObject, eventdata, handles)
if (str2double(get(hObject,'String'))<1)
    set(hObject,'String','1');
end;

function filename_Callback(hObject, eventdata, handles)

%sets limits of session time user input box
function session_value_Callback(hObject, eventdata, handles)
```

```
set(findobj('tag','build_button'),'Enable','on');
set(findobj('tag','notes'),'string','The Program needs to be ReBuilt to take in the changes')
if (str2double(get(hObject,'String'))<5)
   set(hObject,'String','5');
end;
%sets limits of center user input box
function center_value_Callback(hObject, eventdata, handles)
set(findobj('tag','build_button'),'Enable','on');
set(findobj('tag','notes'),'string','The Program needs to be ReBuilt to take in the changes')
if (str2double(get(hObject,'String'))<70)
   set(hObject,'String','70');
end;
if (str2double(get(hObject,'String'))>130)
   set(hObject,'String','130');
end;
%sets limits of rest time user input box
function rest_value_Callback(hObject, eventdata, handles)
set(findobj('tag','build_button'),'Enable','on');
if (str2double(get(hObject,'String'))<10)
   set(hObject,'String','10');
end;

%sets limits of freq user input box
function freq_value_Callback(hObject, eventdata, handles)
set(findobj('tag','build_button'),'Enable','on');
set(findobj('tag','notes'),'string','The Program needs to be ReBuilt to take in the changes')
if (str2double(get(hObject,'String'))<0.2)
   set(hObject,'String','0.2');
end;
if (str2double(get(hObject,'String'))>3)
   set(hObject,'String','3');
end;

%sets limits of spring stiffness user input box
function spring_value_Callback(hObject, eventdata, handles)
set(findobj('tag','build_button'),'Enable','on');
set(findobj('tag','notes'),'string','The Program needs to be ReBuilt to take in the changes')
if (str2double(get(hObject,'String'))<0.1)
   set(hObject,'String','0.1');
end;
if (str2double(get(hObject,'String'))>2)
   set(hObject,'String','2');
end;

set(findobj('tag','build_button'),'Enable','on');

%sets limits of amplitude user input box
function amp_value_Callback(hObject, eventdata, handles)
set(findobj('tag','build_button'),'Enable','on');
set(findobj('tag','notes'),'string','The Program needs to be ReBuilt to take in the changes')
if (str2double(get(hObject,'String'))<0)
   set(hObject,'String','0');
end;
if (str2double(get(hObject,'String'))>40)
   set(hObject,'String','40');
end;
```

```
% --- Executes during passive object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','r');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit9_Callback(hObject, eventdata, handles)

function expled_CreateFcn(hObject, eventdata, handles)

function comment_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','k');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function comment_Callback(hObject, eventdata, handles)
function figure1_CreateFcn(hObject, eventdata, handles)
 set(hObject,'backcolor','k');
function refresh_button_CreateFcn(hObject, eventdata, handles)
function notes_CreateFcn(hObject, eventdata, handles)
if ispc
        set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function notes_Callback(hObject, eventdata, handles)
function fmri_start_value_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


%%sets limits of fmri startup time  user input box
function fmri_start_value_Callback(hObject, eventdata, handles)
if (str2double(get(hObject,'String'))<10)
    set(hObject,'String','10');
end;
```

PART C of MATLAB code for the EMG-Matching_Experiment .

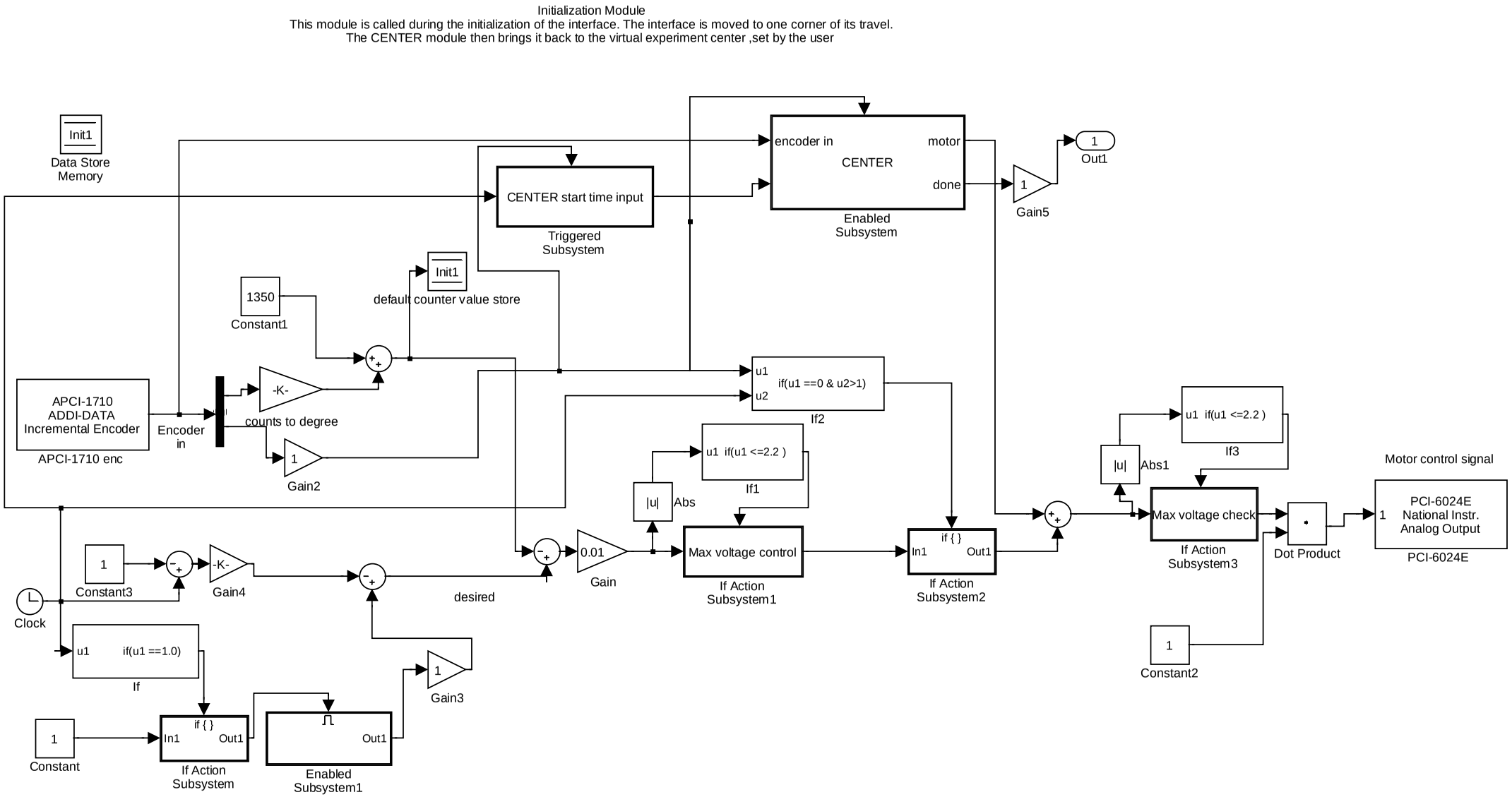
```
% --- Executes on button press in start_button.
%This starts the execution of the experiment

function start_button_Callback(hObject, eventdata, handles)
set(findobj('tag','start_button'),'Enable','off');
disp=zeros(40,1);
global FILE_NAME SETS REST_TIME SESSION
SESSION=1;
index=1;
START_ON=1;
% read in file name fmri startup time and sets
FILE_NAME=get(findobj('tag','filename'),'string');
SETS=str2double(get(findobj('tag','sets_value'),'string'));
FMRI_START=str2double(get(findobj('tag','fmri_start_value'),'string'));
SESSION=1;
START_ON=1;
bar(20,'w'); % clears display

meansum=0;
tg=xpc;
oldtime=str2double(tg.sessiontime);
set(findobj('tag','comment'),'string','');

total_mean=0.0001; % initialization of mean value

for num=1:2*SETS
   unload (tg);
if(SESSION==1)
   load(tg,'RYTHM');
end;
if(SESSION==-1)
   load(tg,'IMPEDENCE');
end;
if(num==1)    % waiting for fmri triggers
set(findobj('tag','notes'),'string','Waiting for fMRI trigger');
   while(tg.s90>2.0)    %------fmri trigger
   %wait;
   end;
   oldtime=str2double(tg.sessiontime);
set(findobj('tag','notes'),'string','fMRI trigger recieved');
end;

tg.stoptime=35.00;
newtime=str2double(tg.sessiontime);
if (num==1)
  pause(FMRI_START-newtime+oldtime); %--time SYNCHRONIZATION
else
index=1;
% display control during rest phase (display of bar movement pattern  as recorded in exp session)
while ((newtime-oldtime)<REST_TIME)
  if (disp(index)<19)
   bar(disp(index));
else
```

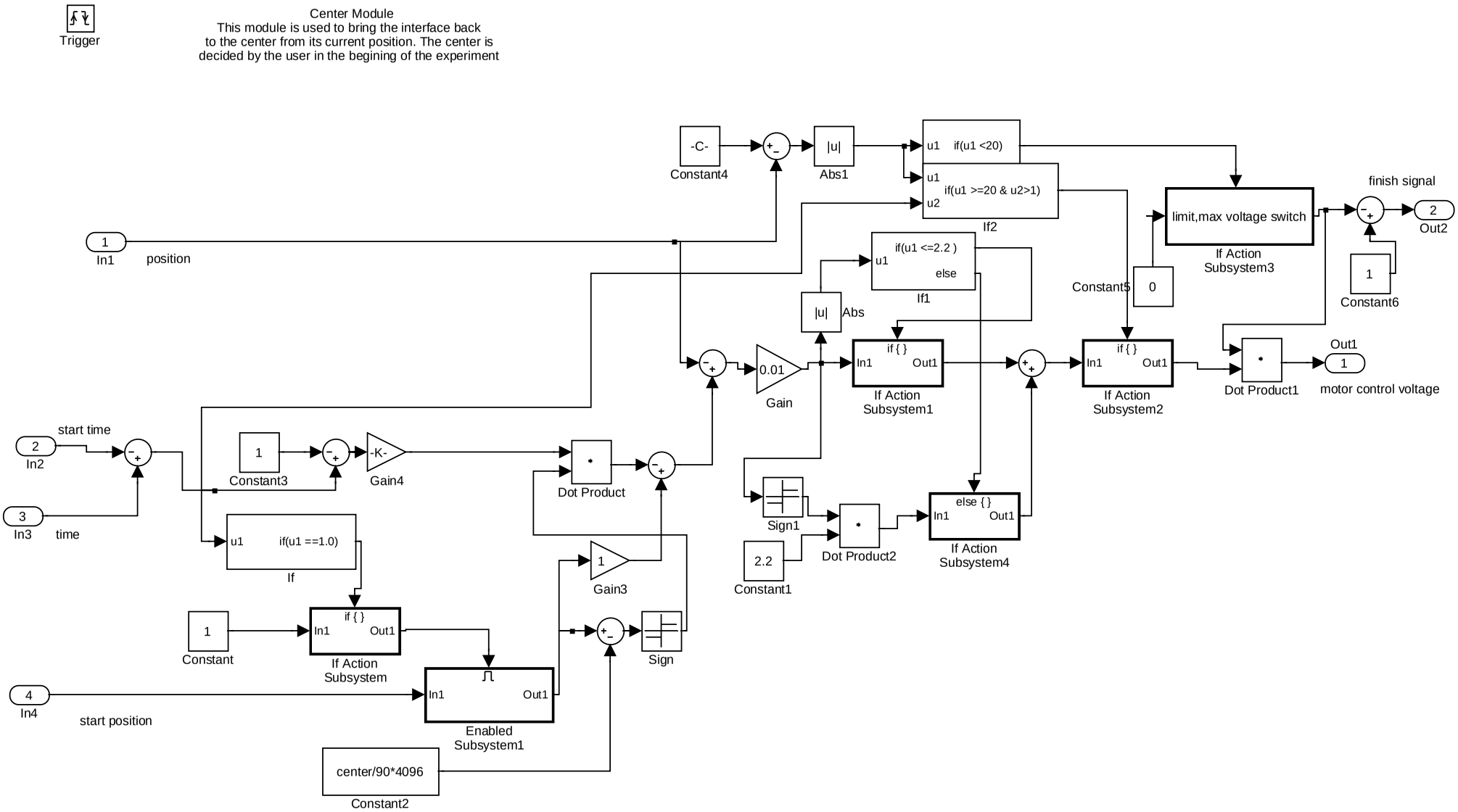
```
    bar(19);
end;
 index=index+1;
 pause(1);
 newtime=str2double(tg.sessiontime);
end;
index=1;
end;
%______REST PHASE ENDS

if(SESSION==-1) %------IMPEDENCE  Session

tg.P23=1.0;%-------motor enable on
    tg.start;
    set(findobj('tag','pushbutton5'),'backgroundcolor','g');  % change LED color
    set(findobj('tag','notes'),'string','EMG_Experiment in progress.   SESSION--IMPEDENCE');
begintime=str2double(tg.sessiontime);
hh=0;
while(tg.s80==0)    % session not finished
    hh=hh+1;   % counter to average the last three values to get a moving average
    if(hh>3)
        hh=1;
    end;
    pause(0.5);
    last3_e(hh)=tg.s95;  %---summation of agonst
    last3_f(hh)=tg.s97;  %---summation of antagonist
    % moving average
    sum_e=sum(last3_e)/3;
    sum_f=sum(last3_f)/3;
    temp_f=last3_f(hh);
     temp_e=last3_e(hh);
    temptime=str2double(tg.sessiontime);
    meansum_f=temp_f/(temptime-begintime);
    meansum_e=temp_e/(temptime-begintime);
    % display control  agonist display bar
    axes(handles.flexor_axis)
    if(sum_f>=2*total_mean_f)
        bar(19,'g');
    else
        bar(sum_f*10/total_mean_f,'g');
    end;
    axes(handles.ext_axes)
% antagonist display bar
if(sum_e>=2*total_mean_e)
        bar(19,'y');
    else
        bar(sum_e*10/total_mean_e,'y');
    end;

end;

set(findobj('tag','pushbutton5'),'backgroundcolor','r'); % LED color change
pause(.01);
while(tg.s93==0) %-------session done signal
    %wait  for 'CENTER'ing  to finish
end;
```

```
tg.stop;
tg.P23=0.0;% stop motor
end;


if(SESSION==1)
tg.P56=1.0; %---motor enable on
   tg.start;
   set(findobj('tag','pushbutton5'),'backgroundcolor','g');
   set(findobj('tag','notes'),'string','EMG_Experiment in progress.  SESSION--RYTHEM');
begintime=str2double(tg.sessiontime);
hh=0;
while(tg.s89==0) %----switch
    hh=hh+1;
   if(hh>3)
      hh=1;
   end;
   pause(0.5);
   last3_e(hh)=tg.s100;  %---summation of flexor
   last3_f(hh)=tg.s96;   % summation of extensors
    %----moving average
   sum_e=sum(last3_e)/3;
   sum_f=sum(last3_f)/3;
    temp_f=last3_f(hh);
    temp_e=last3_e(hh);
    temptime=str2double(tg.sessiontime);

  meansum_f=temp_f/(temptime-begintime);
   meansum_e=temp_e/(temptime-begintime);
 if (num>1)
   %Display control for agonist bar
   axes(handles.flexor_axis)
   if(sum_f>=2*total_mean_f)
      bar(19,'g');
   else
      bar(sum_f*10/total_mean_f,'g');
   end;
   axes(handles.ext_axes)
% display control for antagonist bar
if(sum_e>=2*total_mean_e)
      bar(19,'y');
   else
      bar(sum_e*10/total_mean_e,'y');
   end;
end;
end;
set(findobj('tag','pushbutton5'),'backgroundcolor','r');% change LEd color
pause(0.01);
while(tg.s97==0) %----done
   %wait  for 'CENTER'ing  to finish
end;
tg.stop;
tg.P56=0.0; % motor enable off
end;

%--------REST SESSION STARTS
```

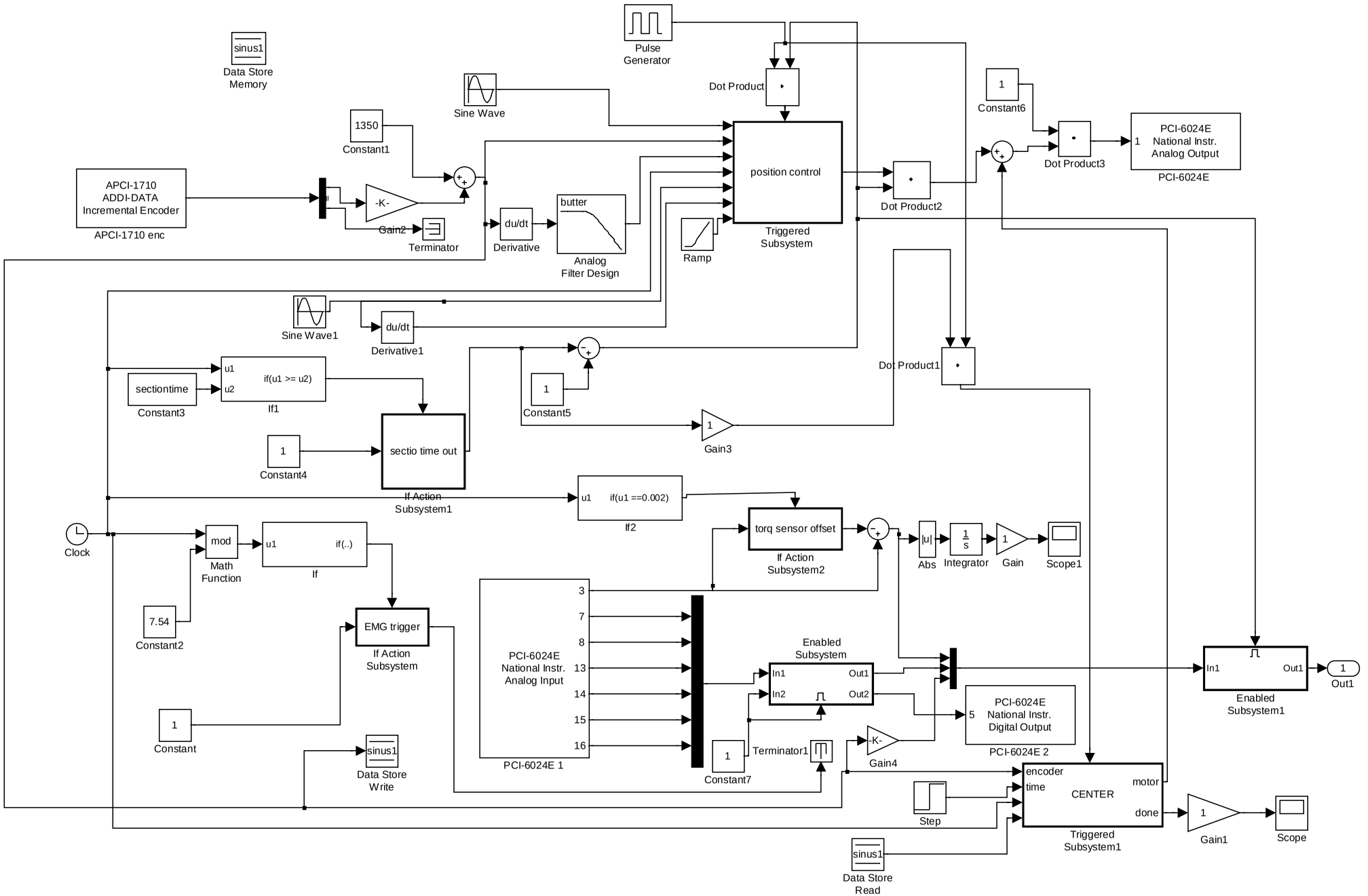
```
oldtime=str2double(tg.sessiontime);

set(findobj('tag','notes'),'string','Experiment in progress.   SESSION--REST');
pause(0.01);
%----data transfer--------
output=tg.output;
time=tg.time;
s=[FILE_NAME '[' int2str(num) ']'];
save(s,'output','time');
%--------------------
total_mean=meansum; % recalculation of mean
SESSION=SESSION*(-1);
end;

% end game
set(findobj('tag','start_button'),'Enable','on');
set(findobj('tag','comment'),'string','The experiment is now over.Wanna have more ??!!')
set(findobj('tag','notes'),'string','The experiment is now over.Wanna have more ??!!')

% ---end of  START button code
```

Simulink Modules

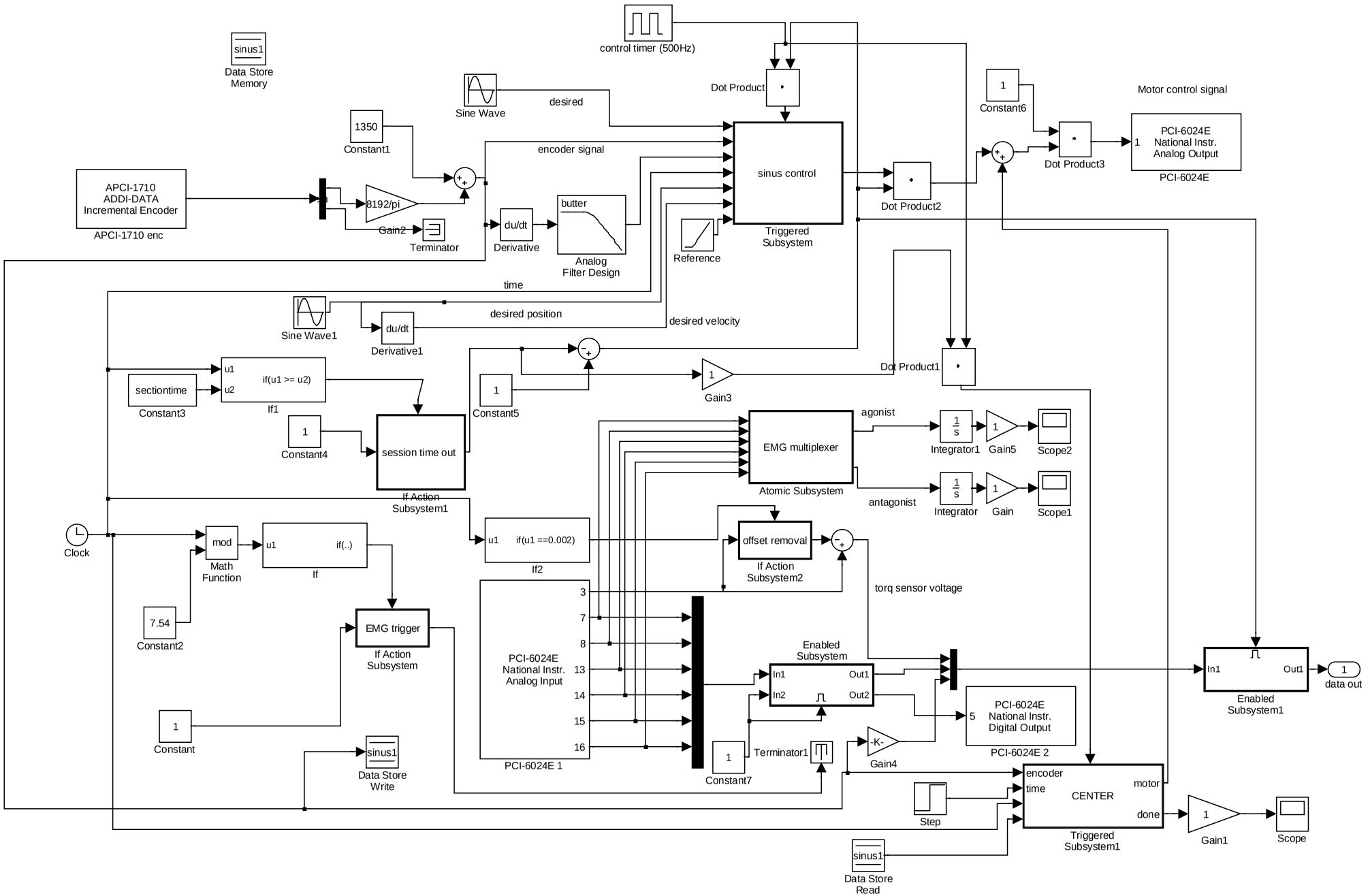

Initialization Module
This module is called during the initialization of the interface. The interface is moved to one corner of its travel.
The CENTER module then brings it back to the virtual experiment center ,set by the user
Init1
Data Store Memory
CENTER start time input
Triggered Subsystem
encoder in
motor
CENTER
done
Enabled Subsystem
1
Out1
Gain5
1350
Constant1
Init1
default counter value store
APCI-1710 ADDI-DATA Incremental Encoder
APCI-1710 enc
Encoder in
-K-
counts to degree
1
Gain2
u1
u2
if(u1 ==0 & u2>1)
If2
u1 if(u1 <=2.2)
If1
|u|
Abs
u1 if(u1 <=2.2)
If3
|u|
Abs1
Motor control signal
Max voltage check
If Action Subsystem3
Dot Product
PCI-6024E
National Instr.
Analog Output
PCI-6024E
1
Constant3
-K-
Gain4
desired
0.01
Gain
Max voltage control
If Action Subsystem1
if { }
In1
Out1
If Action Subsystem2
Clock
u1
if(u1 ==1.0)
If
1
Gain3
1
Constant2
1
Constant
if { }
In1
Out1
If Action Subsystem
Out1
Enabled Subsystem1


Trigger
Center Module
This module is used to bring the interface back
to the center from its current position. The center is
decided by the user in the begining of the experiment
-C-
Constant4
|u|
Abs1
u1 if(u1 <20)
u1
if(u1 >=20 & u2>1)
u2
If2
limit,max voltage switch
If Action
Subsystem3
finish signal
2
Out2
1
In1
position
if(u1 <=2.2)
u1
else
If1
Constant5
0
1
Constant6
|u|
Abs
0.01
Gain
if { }
In1 Out1
If Action
Subsystem1
if { }
In1 Out1
If Action
Subsystem2
Dot Product1
Out1
1
motor control voltage
start time
2
In2
1
Constant3
-K-
Gain4
Dot Product
3
In3
time
u1 if(u1 ==1.0)
If
Sign1
2.2
Constant1
Dot Product2
else { }
In1 Out1
If Action
Subsystem4
1
Gain3
1
Constant
if { }
In1 Out1
If Action
Subsystem
Sign
4
In4
start position
In1 Out1
Enabled
Subsystem1
center/90*4096
Constant2

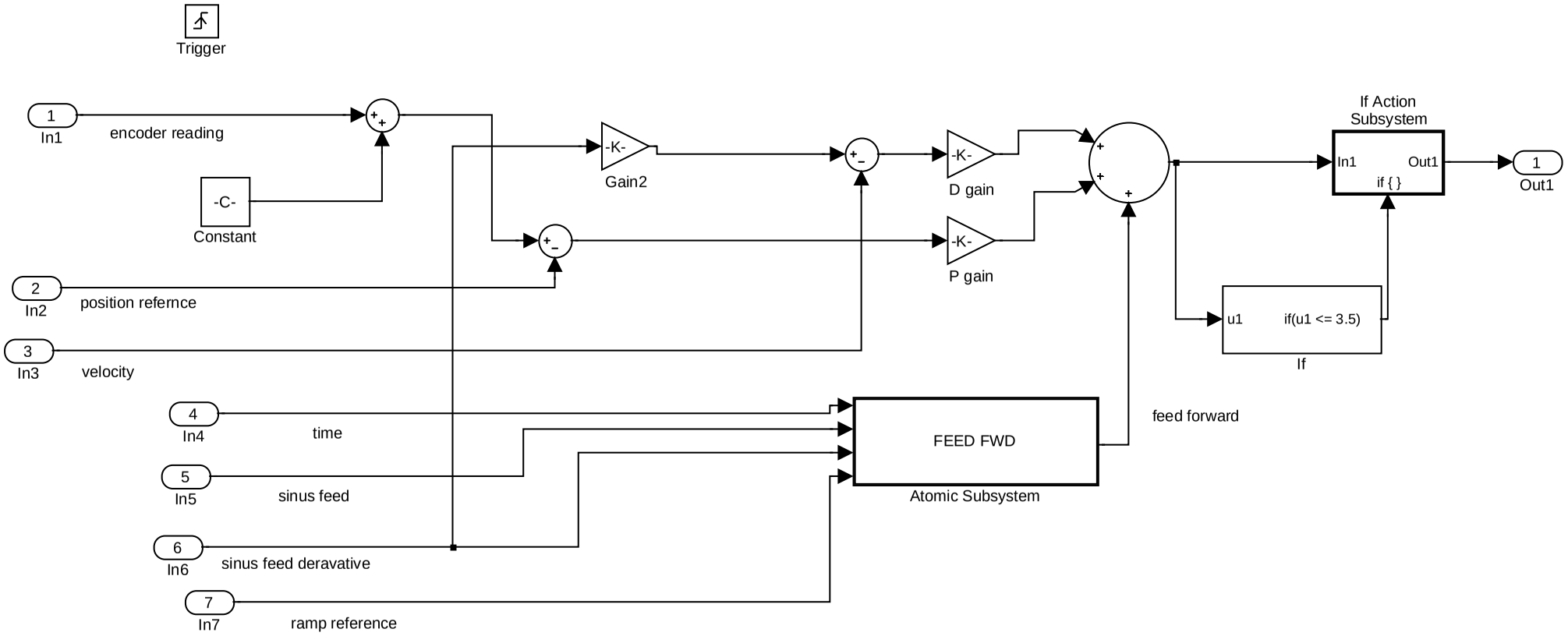
Impedence Module (Force Matching Experiment)
This module does position control to get sinosoidal movemnts of desired frequency and Aplitude.
It also acquires the EMG and subject torque data during the experiment. The summed up value of the torque sensor voltage is used to feedback to the subject.

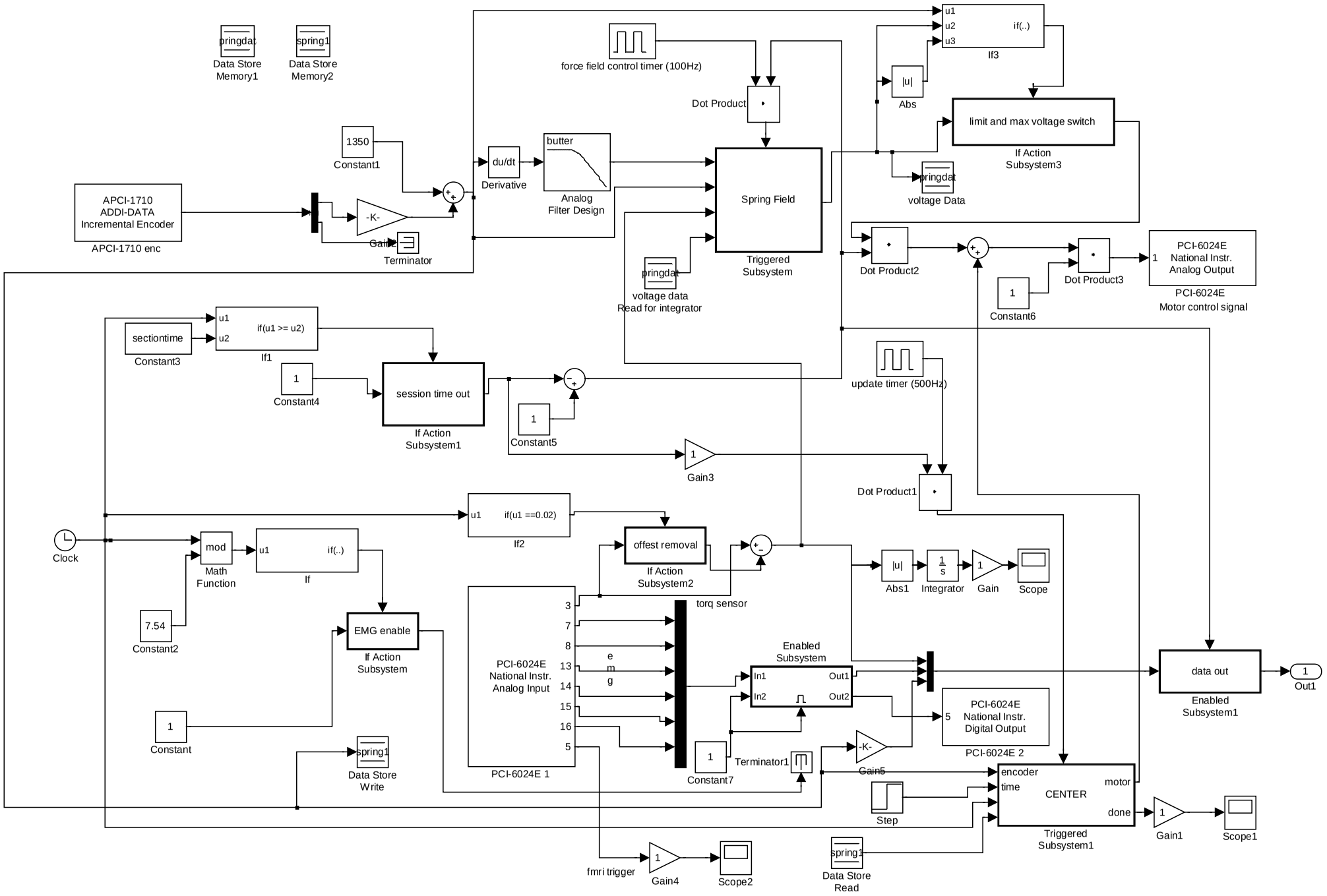


Impedence Control Module (EMG Matching Experiment).
This module does position control to get sinosoidal movemnts of desired frequency and Aplitude.
It also acquires the EMG and subject torque data during the experiment. The EMG data is summed up seperately as agonist and antagonist muscle signals.

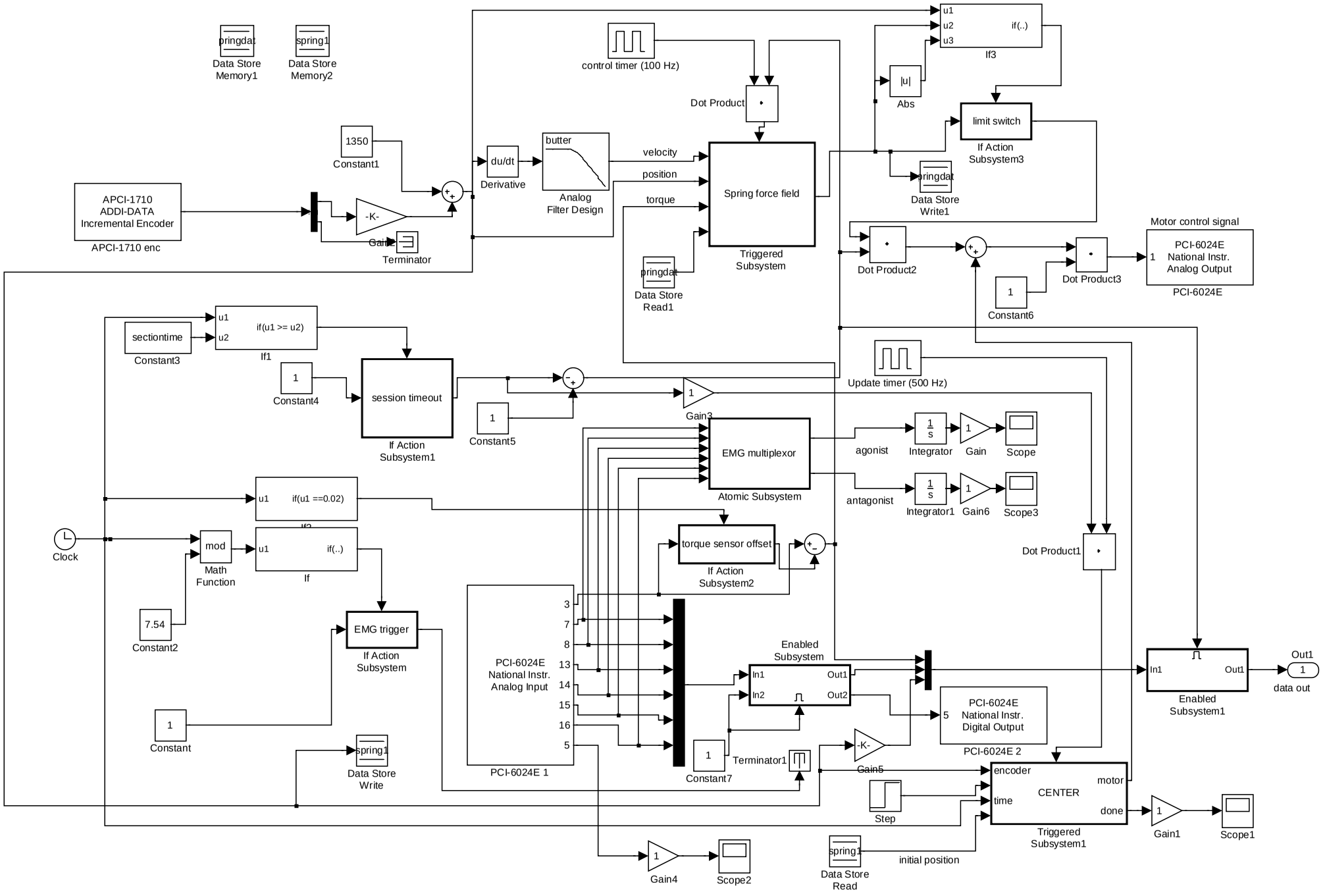


Position COntrol Block (IMPEDENCE MODULE)
This block is used to control the sinus movements in the IMPEDENCE session.
The feed forwardsignal is generated by the FEEDFWD block

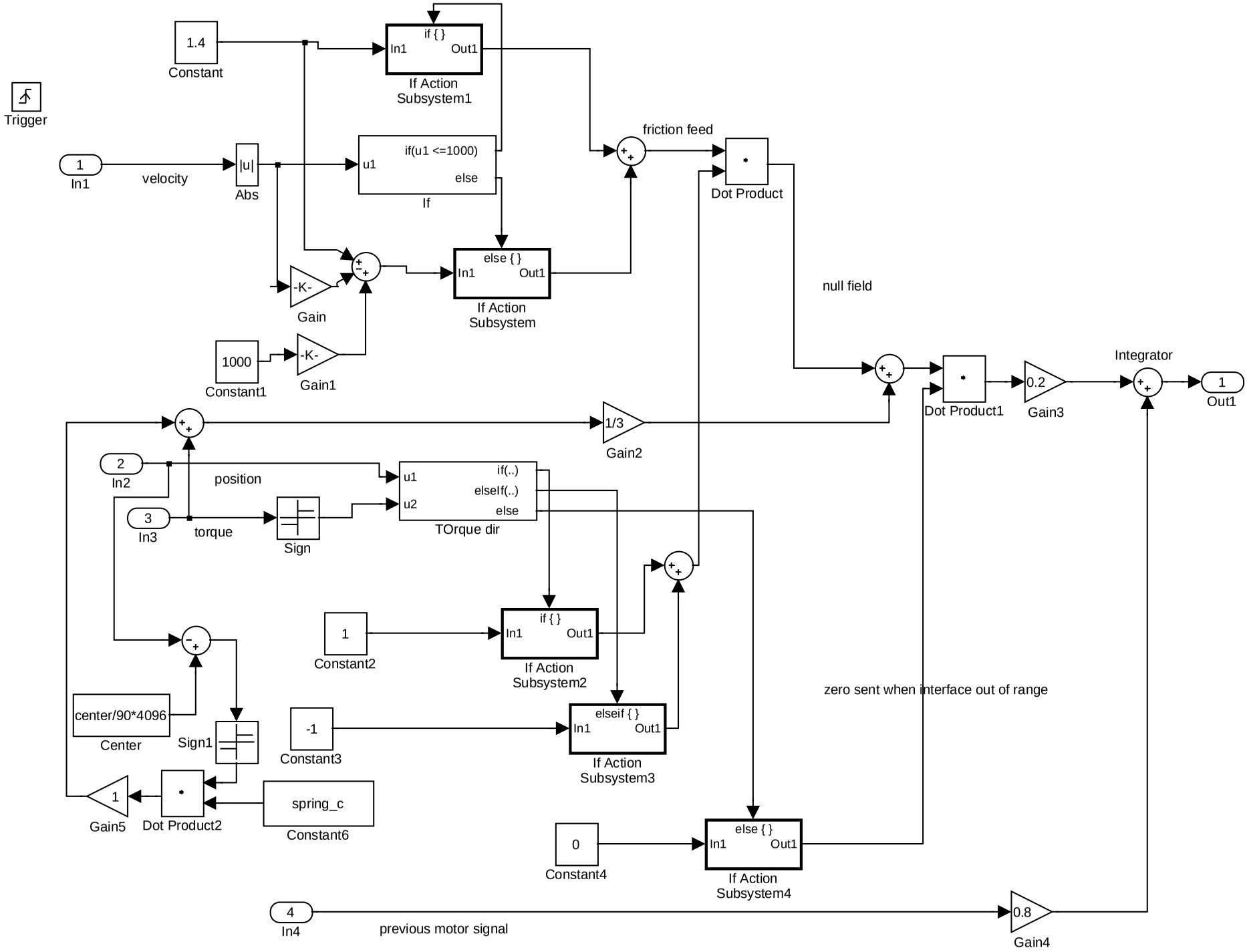



pringdat
Data Store Memory1
spring1
Data Store Memory2
force field control timer (100Hz)
u1
u2
u3
if(..)
If3
Dot Product
|u|
Abs
limit and max voltage switch
If Action Subsystem3
1350
Constant1
du/dt
Derivative
butter
Analog Filter Design
Spring Field
Triggered Subsystem
pringdat
voltage Data
APCI-1710 ADDI-DATA Incremental Encoder
APCI-1710 enc
-K-
Gain2
Terminator
pringdat
voltage data Read for integrator
Dot Product2
Dot Product3
PCI-6024E National Instr. Analog Output
PCI-6024E Motor control signal
Constant6
sectiontime
Constant3
u1
u2
if(u1 >= u2)
If1
1
Constant4
session time out
If Action Subsystem1
1
Constant5
update timer (500Hz)
Gain3
Dot Product1
u1
if(u1 ==0.02)
If2
Clock
mod
Math Function
u1
if(..)
If
offest removal
If Action Subsystem2
torq sensor
|u|
Abs1
1/s
Integrator
Gain
Scope
7.54
Constant2
EMG enable
If Action Subsystem
PCI-6024E National Instr. Analog Input
3
7
8
13
14
15
16
5
e
m
g
Enabled Subsystem
In1
In2
Out1
Out2
PCI-6024E National Instr. Digital Output
5
PCI-6024E 2
data out
Enabled Subsystem1
1
Out1
1
Constant
spring1
Data Store Write
PCI-6024E 1
1
Constant7
Terminator1
-K-
Gain5
encoder
time
CENTER
motor
done
Triggered Subsystem1
Step
Gain1
Scope1
fmri trigger
1
Gain4
Scope2
spring1
Data Store Read

Rythem Control Module (EMG Matching experiment)
This module controls the spring field implemented during the rythem session.
The module stores EMG and torque sensor data during the experiment. This module is specifically
for the EMG matching experiment and thus stores the summation of the agonist and antagonist EMG data.



Spring Field block (RYTHM Module , EMG_RYTHM module)
This block is used for control of the spring field



FEED FWD Block (RYTHM module/ Position COntrol block)
This block generated the feed forward command used in the control during the IMPEDENCE session