Non Linear State Estimation of a Multi Axis Surgical Robot

Srikrishnan Ramadurai

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science
in Mechanical Engineering

University of Washington

2011

Program Authorized to Offer Degree: Mechanical Engineering
This is to certify that I have examined this copy of a master’s thesis by

Srikrishnan Ramadurai

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final examining committee have been made.

Committee Members:

__________________________
Blake Hannaford

__________________________
Joseph Garbini

__________________________
Martin Berg

Date: __________________________
In presenting this thesis in partial fulfillment of the requirements for a master’s degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this thesis is allowable only for scholarly purposes, consistent with “fair use” as prescribed in the U.S. Copyright Law. Any other reproduction for any purpose or by any means shall not be allowed without my written permission.

Signature

Date
Minimally invasive surgical robots often have cable driven power transmission mechanisms. An example is the RAVEN surgical robot developed at the Biorobotics Laboratory, University of Washington for research on robotic surgery. The use of flexible cable based power transmission often causes a difference between the motor angle and joint angles during operation due to the elasticity of the cable. To achieve good control, controllers typically need to account for the dynamics and the elastic power transmission element. Recent control methodologies that can be used to improve performance often use a state space representation. To study the state estimation on the RAVEN, state estimates of a simulation of the RAVEN are obtained with the Unscented Kalman Filter (UKF) and compared with the known states available from the simulation. These state estimates are also utilized by two different controllers interacting with the simulation to test the UKF performance under closed loop control. We tested the UKF performance with perturbations in the UKF model cable stiffness parameter. The simulation is developed based on device file based I/O using the Filesystem in Userspace(FUSE)/Character device in Userspace(CUSE) library. We attempted to develop the simulation with real-time performance considerations combined with userspace development and ability to seamlessly switch the control software from simulation to real system.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td></td>
<td>ii</td>
</tr>
<tr>
<td>Chapter 1:</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Chapter 2:</td>
<td>Simulation System</td>
<td>4</td>
</tr>
<tr>
<td>2.1 System Description</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>2.2 Software Implementation</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>Chapter 3:</td>
<td>Robot Model</td>
<td>15</td>
</tr>
<tr>
<td>3.1 System Description</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>Chapter 4:</td>
<td>UKF on RAVEN</td>
<td>22</td>
</tr>
<tr>
<td>4.1 UKF Implementation</td>
<td></td>
<td>22</td>
</tr>
<tr>
<td>Chapter 5:</td>
<td>State Estimation and Control</td>
<td>27</td>
</tr>
<tr>
<td>Chapter 6:</td>
<td>Results</td>
<td>32</td>
</tr>
<tr>
<td>6.1 Results-Simulation</td>
<td></td>
<td>32</td>
</tr>
<tr>
<td>Chapter 7:</td>
<td>Discussion</td>
<td>41</td>
</tr>
<tr>
<td>7.1 Simulation</td>
<td></td>
<td>41</td>
</tr>
<tr>
<td>7.2 State estimation</td>
<td></td>
<td>42</td>
</tr>
<tr>
<td>7.3 Conclusion</td>
<td></td>
<td>44</td>
</tr>
<tr>
<td>Chapter 8:</td>
<td>Future Work</td>
<td>46</td>
</tr>
<tr>
<td>8.1 Simulation</td>
<td></td>
<td>46</td>
</tr>
<tr>
<td>8.2 State estimation</td>
<td></td>
<td>46</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>48</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Illustration of selection between simulation and real robot.</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Detailed illustration of interaction of the control software with simulation and real robot. The FUSE/CUSE library is used to create a device file with the same operations as the usb driver. The solid line indicates the path of a call. The dash-dot line represents the return for the calls.</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>Illustration of the simulation of the system over a time step. $x_k$ and $\tau_k$ are the states and motor input command of the robot at the present time step respectively. $x_{k+1}$ is the state of the robot at the subsequent time step.</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>Illustration of the sequence of operations occurring within 1 loop of the control software. ENC denotes the encoder values being returned. ACK denotes the acknowledgement of the torques. The simulation performs the numerical integration to obtain the states at the next time step from the states and torque at the current time step. The real robot has the sample and hold applying the torque until the torque changes from a control command at the subsequent time step.</td>
<td>11</td>
</tr>
<tr>
<td>2.5</td>
<td>Jitter in the control software while working with the simulation.</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>Illustration of power transmission model used in the robot. The directions of torques are illustrative and the actual directions depend on the gearbox used.</td>
<td>18</td>
</tr>
<tr>
<td>5.1</td>
<td>Illustration of PD control on joint. $q_l$ and $q_{ld}$ are the joint positions and desired joint positions respectively. $\dot{q}<em>l$ and $\dot{q}</em>{ld}$ are the joint velocities and desired joint velocities respectively. $\tau_m$ is the torque applied on the motor. $q_m$ is the measured motor position. $\dot{q}_m$ is the motor velocity.</td>
<td>30</td>
</tr>
<tr>
<td>5.2</td>
<td>Illustration of PD control on joint with approximate feedback linearization. $q_l$ and $q_{ld}$ are the joint positions and desired joint positions respectively. $\dot{q}<em>l$ and $\dot{q}</em>{ld}$ are the joint velocities and desired joint velocities respectively. $\tau_m$ is the torque applied on the motor. $q_m$ is the motor position. $\dot{q}_m$ is the motor velocity. $A$ is the inertia matrix of the robot. $H$ consists of centrifugal, Coriolis, frictional and gravitational components. $tr$ is the transmission ratio of the gearbox from the motor to the joint capstan.</td>
<td>31</td>
</tr>
</tbody>
</table>
6.1 Plot showing the actual position, position estimates and joint position calculated kinematically by neglecting cable deformation for the shoulder, elbow and z insertion when the reference trajectory is a sine of 3.05 Hz with motor PD control: Correct stiffness parameters ................................................................. 36
6.2 Plot showing the tracking using the UKF estimate with correct parameter and approximate feedback linearization and PD control on linearized system. .............................. 37
6.3 UKF Velocity estimates of the shoulder, elbow and z insertion when the reference trajectory is a sine of 3.05 Hz with motor PD control not using UKF estimates: Correct stiffness parameters ................................................................. 38
6.4 UKF Velocity estimates of the shoulder, elbow and z insertion when the reference trajectory is a sine of 3.05 Hz with motor PD control not using UKF estimates: 0.9x Correct stiffness parameters ................................................................. 39
6.5 Velocity estimates of the motor for the shoulder joint when the reference trajectory is a sine of 3.05 Hz with motor PD control: Correct stiffness parameters. The velocity estimates are poor possibly due to the truncation error in the ERK step. ......................... 40
Chapter 1

INTRODUCTION

The RAVEN surgical robot is a 7 Degree of Freedom (DOF) cable driven surgical robot developed at the Biorobotics Laboratory, University of Washington for research on minimally invasive robotic surgery.[16, 22]. Cable drive between motor and mechanism joint have been used in haptic devices other surgical robots, for example [2, 18]. The use of cable driven power transmission reduces the mass and inertia of the arms by moving the actuators to the base. This also reduces the profile of the arm and increases the portability and suitability for use in minimally invasive surgical procedures.

Although the cables are reasonably stiff, they still deform, especially under high dynamic loads or interaction with heavier objects. As a consequence, the joint positions are not always proportional to the motor positions and are instead related by the dynamics of power transmission through the elastically deformed cable. The elasticity in the cable also means that there could be oscillations in the joint position. Surgical robots can be automated to improve the speed and/or accuracy of surgical robots. Partial autonomy can also be used to reduce the tedium of repetetive surgical tasks for the surgeon. This requires design of control systems that implements such automation on the robot. Recent control system methodologies use a state space representation for designing controllers. Furthermore, it may not be possible (due to cost, complexity, sterilization requirements etc) to implement sensors for every quantity that need to be determined. State estimators can be used to obtain estimates of the internal state of a system from some inputs and outputs provided that the system is operating in a region where it is observable. Additionally, the state estimation can also provide cleaner signals of the measured quantity from noisy sensor data.

For the RAVEN, the task considered here is the determination of the robot states from the motor positions and motor torque inputs. The UKF is used as a state estimator where the forward dynamic
model of the robot along with the elastic power transmission is used to estimate the joint position and velocities.

The Unscented Kalman Filter (UKF) [11] is a state estimator for non linear systems based on the unscented transformation [11]. The UKF is able to estimate state without using derivatives. Furthermore, the UKF is more accurate compared to the Extended Kalman Filter (EKF) since the UKF captures the mean and covariance upto second order. Prior research at the Biorobotics Lab [20] has demonstrated the feasibility of using UKF for state and parameter estimation on cable driven mechanisms. For reasons of numerical stability, a square root form of the UKF [25] was implemented.

This implementation of the UKF considers only the first three DOFs of the RAVEN since the end effector position is uniquely determined by these axes. The remaining DOFs are used to control the specific functionality of each individual tool. The estimation of these DOFs is specific to each type of tool and will be future work.

In an earlier study, the UKF was experimentally evaluated on a 1 DOF cable driven device called the pulley board as described in [20]. The experiments conducted on the pulley board were used to test the feasibility of using the UKF on a device with elastic power transmission. [20] modeled the elastic transmission with a exponential spring and linear damping. The study also showed that there exists a measurable deformation of the cable during operation. The experiments also showed that the deformation depends on external loads. This deformation and the change in dynamics due to this elasticity compared to a rigid transmission should be accounted for if accurate control is desired.

In this work, we estimate the states of a simulated surgical robot with flexible cable power transmission. The input and output configuration of the simulation is similar to that of the real robot, where the motor angles are measured instead of the joint angles (due to size, wiring,complexity and sterilization issues). Typically, the motion of the joint is assumed to be directly related to the motion of the motor. We use a model that also considers the effect of cable deformation between the motor and the joint during robot operation. We also show that the joint states differ from those calculated by neglecting the cable deformation. Furthermore, the Robbins Monro stochastic approximation
scheme [6, 14, 23] is used for process noise correction to reduce the effect of incorrect a priori process noise information. We test the UKF with two controllers using the UKF estimates in closed loop. We also develop a real time simulation system to test controllers for the surgical robot which are unmodified compared to the real time controllers for the real robot. We implement the UKF on the control software interacting with this simulation system. Finally, we use this setup to study the implementation issues for using the UKF as a state estimator on cable driven surgical robots.
Chapter 2

SIMULATION SYSTEM

Software designers for robots often find simulations helpful for testing control algorithms, effect of parametric changes on the software, response of the robot to changes in the operating conditions, sensor failure testing etc. A system that mimics the interface electronics would be helpful in testing the robot software against the simulation and real robot with minimal changes to the robot control software. Avoiding changes in the software during the transition from simulation to real life robot would reduce the incidence of bugs during the transition. An example of such a system is presented in this paper. The system consists of RAVEN, a 7 DOF surgical robot designed at the Biorobotics Lab, University of Washington as a research platform for surgical robotics, the BRL USB board, an USB 2.0 based I/O interface and the control software running on a PC. The complete description of the RAVEN hardware consisting of the robot and the USB I/O interface can be found in [16].

Presently, a number of virtual environments exist for the simulation of robots. For example the Gazebo and Player/Stage project [1], the OpenRAVE project [3], to name a few. These packages emphasize multi robot simulations, virtual environment interaction and in some projects, a physics engine is incorporated to simulate the dynamics of rigid bodies and/or interaction with the environment. OROCOS project with the OROCOS RTT toolkit attempts to expose a set of communications interfaces which can potentially be used for simulation. [17] has run realtime simulations of an industrial robot based on a Modelica model using the OROCOS toolkit and using a COMEDI driver to communicate with the virtual robot. In this work, we attempt to design a simulation system for devices based on a character device file using the Filesystem in userspace(FUSE)/Character device in userspace (CUSE) library, which comes built in with linux kernel versions 2.6.33.

The real time platform that is used here will be the RT-Preempt [19] patch for linux. We attempt to keep most of the development in user space and keep all communications as device file based oper-
ations. Device file based operations provide simplicity and ease of use without building additional layers of abstraction and interfaces. In our applications, the device file based I/O also means that we do not have to link against any external library or interface with a middle ware that switches I/O between the simulation or the real robot. This also gives the ability to seemlessly run both the real device and the simulation concurrently. Additionally, our work attempts to avoid writing any kernel driver or kernel space implementation for the simulation.

Our simulation environment needed to satisfy the following requirements

- Control software does not know if it is interacting with real robot or a simulator
- Ability to modify the parameters of the simulated robot to test controller response
- Deterministic response at the control loop speed of the controller: 1000 kHz in our application
- Dynamic simulation of the robot.
- Monitoring parameters of the virtual robot like cable tension, joint frictions etc.
- Induce noise and failures in simulated electronics.
- Keep as much development as possible in the user space to ease development and debugging.

2.1 System Description

2.1.1 Hardware

The simulation is used to simulate RAVEN, a surgical robot developed at the University of Washington as a research platform for robotic surgery. A complete description of the robot can be found in [16].
RAVEN

The RAVEN is a 7 DOF teleoperated robot. The robot is a serial chain robot with cable driven power transmission. The first three DOFs are used to position the tool and the remaining 4 DOFs are for controlling the tool. In this work, the first 3 DOFs of the robot are simulated.

USB Interface

The BRL USB board is a USB 2.0 based I/O device developed for use with various robotic and haptic devices under development at the lab. The board consists of 8 24-bit quadrature encoder inputs and 8 16-bit +/-10V DACs.

Computer

The computer used for all tasks in this paper was x86-64 architecture based computers. The system consisted of an Intel Core2Quad Q8300 CPU, 4 GB ram and an onboard NVIDIA graphics processor.

2.1.2 Software

RT-Preempt patch

The RT-Preempt patch [19] by Ingo Molnar, Thomas Gleixner and others, attempts to provide real-time performance by suitably patching the standard Linux kernel. This architecture provides the benefit of both real-time performance and access to many external libraries to ease the development of complex real-time systems. This system also has the disadvantage that the user must take precautions in using libraries that may not have been written for real-time development. However, libraries such as math libraries are often designed with performance in mind and so their use in real-time applications is fairly straight forward with little modification. An additional benefit of using the RT-Preempt patch but not limited to this patch alone is the use of higher level languages like C++ in real-time applications. [4] provides a set of guidelines to develop real-time applications with C++.
**FUSE/CUSE**

Filesystem in Userspace (FUSE) [24] is a library to create filesystems from userspace. Character devices in userspace (CUSE) [7] is built upon FUSE to create character device files. FUSE/CUSE comes packaged with the Linux kernel 2.6.33 that is used in this work. FUSE/CUSE provides interfaces to the read, write, ioctl etc filesystem calls on a custom filesystem. In the present application, the driver for the USB board’s kernel driver exposes read, write and ioctl calls on a device file. The read and write calls are typically used to communicate data with the board and the ioctl calls are used for control operations on the board. FUSE/CUSE is used to create a virtual device file that exposes the same set of calls that the actual USB board driver exposes. Thus, with this type of interface, the control software would be unaware of the reality of the device it is communicating with.

**SUNDIALS/CVODE**

SUite of Nonlinear and DIfferential/ALgebraic equation Solvers (SUNDIALS) provides a set of time integrators and nonlinear solvers for incorporation into simulation code [8]. CVODE is a ODE initial value problem (IVP) solver for stiff and non-stiff systems. The CVODE library is used in the simulation to integrate the system dynamics over a timestep.

**EIGEN library**

Eigen [5] is a C++ template library for linear algebra. Eigen is used as part of the simulation to perform linear algebra operations. The use of Eigen enables writing the code in a concise manner without a significant performance overhead. Eigen also provides many linear algebra operations like inverses, decompositions, etc built-in. Additionally, Eigen provides built-in support for using the processor’s vector operation features like SSE etc. This accelerates the calculations without additional effort and helps in finishing the simulation quickly.

**Control software**

The existing control software was implemented as a Robotics Operating System (ROS) [21] node running on RT-Preempt patched linux. The robot was teleoperated using a sensable phantom omni.
The underlying controller was a PD controller on each joint with gravity compensation. Figure 2.2 shows the schematic illustration of the setup when the controller is working with the real robot.

Simulation Software

For the simulation case, the FUSE/CUSE library is used to create a device file with the same calls as the USB driver device file. Figure 2.2 shows the schematic of such a setup. The figure shows the calls made by the software and the paths taken by the simulation software. The simulation software performs some task according to the type of command received. For example, the write call records the torque sent from the control software, sends a return value immediately and then proceeds to simulate the system over a time step. This is similar to the USB board, which sets the torque applied on the motor based on values received from the control software. This torque is kept until the next time step, where the same process is repeated.
2.2 Software Implementation

2.2.1 Control software

As mentioned earlier, the control system software runs on a RT-Preempt patched linux kernel with real-time priority. The control software performs the task of reading from the I/O board, processing the encoder data, reading input data from a network based teleoperator interface, calculate the forward and inverse kinematics for the current and desired positions and to compute and send the desired motor control input command to the I/O board. The software also runs safety checks to keep the robot within operational limits. The controller presently uses a PD control law with gravity compensation. The software also has interfaces to set the gains of the controller. Data logging is
accomplished through ROS publisher / subscriber mechanism. The control loop is set to run at 1 kHz.

2.2.2 Simulation Software

The simulation software is divided into parts that perform communication and simulation, visualization and data logging. The communication and simulation part runs under realtime priority. The visualization and data logging parts runs under lower priority. Depending on the requirement, it is possible to log data in real time with modifications to the logging methods. However, it was
absent in our implementation, since most of the data logging is handled by the control software.

To obtain quick response to device calls made by the control software, the CUSE interface was run in a single threaded loop with real-time priority. One of the sources of latency is memory references to pages outside of RAM (faulting). To overcome this, a memory pool was created and the memory allocation option commands of glibc - malloct was used to force any memory allocation from this pool in the ram. In addition, mlockall call was used to lock the address space of the program into physical memory. This helped in avoiding latency produced from any access to persistent storage. [19] describes the considerations to keep in mind while creating real time applications with the RT-Preempt patch.

Figure 2.4 shows an illustration of the events occurring within a time step. The diagram shows

![Diagram](image)

1 Loop cycle (1 ms)

Figure 2.4: Illustration of the sequence of operations occurring within 1 loop of the control software. ENC denotes the encoder values being returned. ACK denotes the acknowledgement of the torques. The simulation performs the numerical integration to obtain the states at the next time step from the states and torque at the current time step. The real robot has the sample and hold applying the torque until the torque changes from a control command at the subsequent time step.

the control software communicating with the subsystems such as DAC and rotary position encoder counter through the USB board. For the simulation, the simulation software exposes interfaces that
are similar to the real electronics. The control software reads the current encoder values (representative of current robot state). The control software sends the desired torque command and a foot pedal state at every loop cycle. The foot pedal state which is an on or off state is forwarded from the master side and determines the state of the motor brakes. The purpose of the foot pedal is to enable clutching while teleoperation. The control software immediately receives an acknowledgement after sending the torque command and foot pedal state. From here on, on the real robot, the sample and hold of the DAC sends the voltage corresponding to the torque command to the motor amplifier, and the I/O board simultaneously sends the signal to the motor brake to turn on/off depending on the received foot pedal state. For the simulated robot, the simulation software instructs the CUSE library to return the call similar to the real board, and proceeds to simulate the system over a time step with the received torque.

To accomplish the simulation, the ODE solver CVODE is configured to run with Adams-Moulton method with functional iteration. This is an adaptive step, variable order and derivative free method which works well for systems with non-stiff problems. In some cases where the system is numerically stiff, for example when the friction in the model is high, the backwad differentiation formulae (BDF) based solver with a diagonally approximated jacobian seems to work better. An adaptive solver was used because the robot has elastic power transmission. This means that the inertias on the motor side are much smaller than those on the joint side. This caused large derivatives for short durations in some configurations when the cable stretches. The adaptive step solver is able to step in small increments when derivatives are large and then make larger steps when the magnitude of the derivatives reduce. The solver used the system model to calculate the state of the system after a time step. In this computation, the zero order hold effect was produced by keeping the input constant over the whole time step. The system model, derived from the forward dynamics of the robot, also modeled the effects of coulomb, viscous and static friction in addition to the nonlinear dynamics of the robot and the elastic power transmission of the robot. The braking state was simulated by boosting the static friction present in the motors.

The time taken by the solver to complete the integration is affected by the desired accuracy. A low accuracy would improve the speed at the cost of drift and numerical instability over time. On
the other hand, a very high accuracy would increase the integration time causing missed deadlines in the 1ms realtime loop. A compromise was chosen by capping the number of iterations at 500 and choosing a sufficiently large accuracy (scalar relative tolerance and vector absolute tolerance both set to $1e-5$). In our experience, the integrator faced convergence issues mainly during braking. One of the reasons was that the CVODE in the current configuration keeps a history to pick good initial guesses for step sizes and integrator coefficients. However the braking events represented a change in system model and the numerical stiffness of the system. This problem was largely reduced by resetting the integrator with the current state just after changing the model.

Figure 2.5 shows a plot of the jitter present in the control loop when the control software is working with the simulation. The jitter was measured by calculating the time difference from the time stamps of a high resolution timer between each time step. The plot shows the jitter to be mostly around 50 to 100 $\mu$s which corresponds to about 5% to 10% at 1 kHz. The maximum jitter in this measured window is about 200 $\mu$s.

The choice of tools used in developing this system were made keeping in mind the considerations of performance and ease of use. One of the driving factors in the development of this system is the ability to test a controller prior to deployment on the robot. Consequently, the simulation system needed to satisfy the timing constraints posed by the robot controller. The processor used was a Intel Core2Quad multicore processor. To reduce the jitter caused by threads switching between cores of the processor[9], CPU affinity was used to lock the control software to 1 core, visualization and logging task to the last core and the simulation to the second and third cores. The system model was written keeping in mind the considerations for realtime development. One source of uncontrolled latency was the memory allocation calls made by the FUSE/CUSE library. This was largely eliminated by using the preallocated memory pool. However, to account for this, the latency caused by memory allocations and freeing was benchmarked. For memory allocations (after creating the memory pool) up to 1 MB, the worst case latency in our tests was found to be about 70 $\mu$s.
Figure 2.5: Jitter in the control software while working with the simulation
3.1 System Description

3.1.1 State space description

The first three DOFs of the RAVEN robot arm consists of 12 states: motor position, motor velocity, joint position and joint velocity for each DOF. The dynamic equations in state space form are derived from the forward dynamic equations. A description of the forward dynamics is presented in the subsequent sections. The dynamic equations are discretized by propagating the dynamics over a time step using a 4th order Explicit Runge Kutta (ERK) method.

The dynamics for the rigid robot can be expressed symbolically as [13]:

\[
\begin{align*}
\dot{v}_q &= \ddot{q} = A^{-1} \left[ \Gamma - H(q, v_q) \right] \\
\dot{q} &= v_q \\
H(q, v_q) &= C(q, v_q) + Q + \text{diag}(v_q)F_v \\
&+ \text{diag}(|v_q|)F_c + J^T f_{en}
\end{align*}
\]

where:

- $A$ - Inertia matrix of the robot
- $Q$ - Gravitational torques acting on robot
- $q$ - Joint angle(rotary) or displacement(prismatic)
- $v_q$ - Joint velocity (angular/linear)
• $C(q, v_q)$ - Coriolis and centrifugal force components

• $F_v$ - Viscous friction in the joint

• $F_c$ - Coulomb friction in the joint

• $f_{en}$ - External torques represented as a wrench

• $J$ - Jacobian of the robot

• $\Gamma$ - Joint torques

Similar to [20], the dynamics for each motor can be expressed as

$$
\dot{v}_{qm} = \dot{q}_m = \frac{1}{J_m}(\tau - \tau_{rn} - F_m) 
$$  
(3.4)

$$
q_m = v_{qm} 
$$  
(3.5)

$$
F_m = F_{cm} \text{sign}(v_{qm}) + F_{cm} v_{qm} 
$$  
(3.6)

$$
\gamma = k_c(e^\Delta - e^{-\Delta}) + 2b_c(v_{qmc}r_m - v_{qi}r_1) 
$$  
(3.7)

$$
\tau_{rnc} = -r_m \gamma 
$$  
(3.8)

$$
\Delta = q_m r_m - q_i r_1 
$$  
(3.9)

$$
\tau_{rn} = \frac{\tau_{rnc}}{t_r} 
$$  
(3.10)

$$
v_{qmc} = \frac{v_{qm}}{t_r} 
$$  
(3.11)

$$
q_{mc} = \frac{q_m}{t_r} 
$$  
(3.12)

and

$$
\Gamma_i = r_i \gamma 
$$  
(3.13)

where:

• $\tau$ - Torque applied on a motor
• $q_m$ - Motor position

• $v_{qm}$ - Motor velocity

• $J_m$ - Motor rotor inertia

• $F_{cm}$ - Motor coulomb friction

• $F_{vm}$ - Motor viscous friction

• $q_i$ - Corresponding joint angle

• $v_{qi}$ - Corresponding joint angular velocity

• $q_{mc}$ - Corresponding motorside capstan angle

• $v_{qmc}$ - Corresponding motorside capstan angular velocity

• $k_e$ - Corresponding cable stiffness

• $b_e$ - Corresponding cable damping

• $t_r$ - Transmission ratio of corresponding gearbox. Sign chosen depending on the presence of kinematically inverting elements in motor and gearbox

• $r_m$ - Motor side capstan radius

• $r_l$ - Corresponding joint side capstan radius

• $\Gamma_i$ - Corresponding joint torque

Figure 3.1 shows an illustration of the power transmission path from motor to the robot joint. The torque applied on the motor is transmitted through a gearbox to the motor side capstan. The power from the motor capstan to the joint capstan is transmitted through the deformation occurring in the cable.
The deformation of the gears in the gear box is neglected. Since the gearbox is used to magnify the torques, the torques acting on the motor side capstan are large compared to inertial and frictional force. So, the effect of inertia and friction on the motor side capstan are neglected. The second RAVEN joint has a two segmented cable run. But, due to the short length and thick cable of one of the runs, it's effect was neglected.

3.1.2 Inverse Dynamics

The inverse dynamics of a rigid robot can be used to compute the joint torques to produce a desired joint acceleration. The inverse dynamics is computed using Newton Euler algorithm as described in [13, 15]. The inverse dynamics is used to compute the forward dynamics as described in the subsequent section. The algorithm for the inverse dynamics computed in the local link coordinate system from [13] is reproduced here:
The forward recursive equations for $j=1,...,n$

\[ \omega_{j-1}^j = A_{j-1}^j \omega_{j-1}^{j-1} \]  
(3.14)

\[ \omega_j^j = \omega_{j-1}^j + \tilde{\sigma}_j \dot{q}_j a_j^j \]  
(3.15)

\[ \dot{\omega}_j^j = A_{j-1}^j \dot{\omega}_{j-1}^{j-1} + \tilde{\sigma}_j (\ddot{q}_j a_j^j + \omega_j^{j-1} \times \dot{q}_j a_j^j) \]  
(3.16)

\[ V_j^j = A_{j-1}^j (V_{j-1}^{j-1} + U_{j-1}^{j-1} P_{j-1}^{j-1}) + \]  
\[ \sigma_j (\ddot{q}_j a_j^j + 2 \omega_j^{j-1} \times \dot{q}_j a_j^j) \]  
(3.17)

\[ F_j^j = M_j \dot{V}_j^j + U_j^j MS_j^j \]  
(3.18)

\[ M_j^j = J_j^j \dot{\omega}_j^j + \omega_j^j \times (J_j^j \omega_j^j) + MS_j^j \times V_j^j \]  
(3.19)

\[ U_j^j = \dot{\omega}_j^j + \omega_j^j \dot{\omega}_j^j \]  
(3.20)

The backward recursive equations, for $j=n,...,1$

\[ f_j^j = F_j^j + f_{j+1}^j + f_{ej}^j \]  
(3.21)

\[ f_{j-1}^j = A_{j-1}^{j-1} f_j^j \]  
(3.22)

\[ m_j^j = M_j^j + A_{j+1}^j m_{j+1}^{j+1} + P_{j+1}^j \times f_{j+1}^j + m_{ej}^j \]  
(3.23)

\[ \Gamma_j = (\sigma_j f_j^j + \tilde{\sigma}_j m_j^j)^T a_j^j + \]  
\[ + F_{sj} sign(\dot{q}_j) + F_{ej} \dot{q}_j + I a_j \dot{q}_j \]  
(3.24)

where,

- $\omega_{j-1}^j$ - Angular velocity of link $j - 1$ with respect to frame $R_j$

- $\omega_j^j$ - Angular velocity of link $j$ in the local coordinate system

- $A_{j-1}^j$ - Rotation component of transformation representing frame $R_{j-1}$ in frame $R_j$

- $P_{j-1}^j$ - Translation component of transformation representing frame $R_{j-1}$ in frame $R_j$

- $\dot{\omega}_j^j$ - Angular velocity of link $j$ in local coordinate system

- $V_j^j$ - Linear velocity of origin $O_j$ of frame $R_j$. 
• $MS_j^l$ - First moment of link $j$ with respect to frame $R_j$. It is mass times the centre of mass in local coordinates.

• $\hat{\omega}_j^l$ - Skew symmetric form of the vector $\dot{\omega}$ which is defined as

$$\hat{u} = \begin{pmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{pmatrix}$$

where $u$ is a vector with components $u = [u_x, u_y, u_z]^T$

• $\hat{\omega}_j^l$ - Skew symmetric form of $\omega_j^l$

• $q$ - Joint angle (rotary) or displacement (prismatic)

• $\dot{q}_j$ - Joint velocity (angular velocity for rotary joints and linear velocity for prismatic joints)

• $\ddot{q}_j$ - Joint accelerations

• $a_j^l$ - Unit vector along z axis in local coordinates. $[0,0,1]^T$.

• $f_{e_j}^l$ - Force exerted by link $j$ on environment

• $m_{e_j}^l$ - Moment about origin $O_j$ exerted by link $j$ on environment

• $\Gamma_j$ - Torque calculated by inverse dynamics that produces the desired accelerations

• $F_{s_j}, F_{v_j}$ - Coloumb friction ($F_{s_j}$) and viscous friction ($F_{v_j}$) components

• $Ia_j$ - Moment of inertia of the rotor on link $j$.

### 3.1.3 Forward Dynamics

The forward dynamics is used to calculate the accelerations produced on the joints for an applied torque. The forward dynamics is computed using the inverse dynamics by the method described in [13, 26]. When, the dynamics are expressed by (3.1)
Coriolis, Centrifugal, Gravity components

The coriolis, centrifugal and gravity components are the torque computed by the inverse dynamics when $\ddot{q} = 0$

$$H(q, v_q) = \Gamma_{\ddot{q}=0}$$ (3.25)

Inertia Matrix

The column $i$ of the inertia matrix $A$, $A_i$, is the torque calculated by inverse dynamics when the desired acceleration is a unit basis vector $e_i$ with 1 in the $i$th row and zeros everywhere else, with $H(q, v_q) = 0$, i.e., $\ddot{q} = 0$, the acceleration due to gravity $g = 0$, $Fc = 0$, $f_{ej} = 0$, $m_{ej} = 0$ for $j=1,\ldots,n$. 

$$Ae_i = A_i = \Gamma_{\ddot{q}=e_i,H(q,v_q)=0}$$ (3.26)

The inertia matrix thus calculated is used with the forward dynamic equation 3.1 as described in the next subsection.

Joint acceleration

The joint accelerations are obtained by solving 3.1. The inertia matrix $A$ and $H(q, v_q)$ are obtained from earlier steps. The solution to equation 3.1 is obtained by either closed form solution for small sizes or by a numerically stable method like the LU decomposition algorithm.

The joint torques used for calculating the forward dynamics are the torques produced as a result of the cable deformation. The reaction torque component of the cable deformation force acts on the motor. Similar to [20] an exponential spring was used to model the elasticity of the cable.
Chapter 4
UKF ON RAVEN

4.1 UKF Implementation

4.1.1 Square root UKF

The square root form of the UKF was used for its favorable numerical properties [25]. The continuous time differential equation representing the system is discretized using the ERK method when used with the UKF. The measurement function for the robot is the motor positions which are a part of the states. i.e., If the continuous time dynamics in the state space representation are described by

\[ \dot{x} = f_c(x(t), u(t)) \] (4.1)

The discretized form is

\[ x_{k+1} = f(x_k, u_k) \] (4.2)

with

\[ f(x_k, u_k) = ERK4(f_c(x(t), u(t))) \] (4.3)

where \( k \) denotes the \( kth \) time step.

4.1.2 Robbins Monro correction

Stochastic approximation methods are iterative algorithms to find the roots of functions which may not be possible to compute directly but can be estimated through noisy observations. Given a function such that

\[ \mathbb{E}[Q(x, e(t))] = f(x) = 0 \] (4.19)

where \( Q(x, e(t)) \) is the noisy observation of \( f(x) \) for a known \( x \) and \( e(t) \) is a random disturbance from some unknown distribution, a convergent solution to (4.19) can be obtained using the Robins-Monro algorithm as [14, 23]

\[ \hat{x}(t) = \hat{x}(t-1) + \gamma(t)Q(\hat{x}(t-1), e(t)) \] (4.20)
where \( \gamma(t) \) is a sequence of positive scalars tending to zero and \( \sum_{t=0}^{\infty} \gamma(t) = \infty, \sum_{t=0}^{\infty} \gamma(t)^2 < \infty \)

The convergence of the UKF is affected by the process noise information provided to it. A Robbins-Monro stochastic approximation scheme \([6, 14, 23]\) was used to approximate the process noise.

The system is given by

\[
x_{k+1} = f(x_k, u_k) + r_k
\]  (4.21)

where \( x_k \) and \( u_k \) is the state and input at time step \( k \) and \( x_{k+1} \) is the state at time step \( k + 1 \), and \( r_k \) is the process noise.

The update step in the UKF is given by

\[
\hat{x}_k = \hat{x}_{k|k-1} + K_k(y_k - y_{k|k-1})
\]  (4.22)

where \( \hat{x}_{k|k-1} \) and \( y_{k|k-1} \) is the predicted state and predicted output, \( y_k \) is the measured output at step \( k \), \( \hat{x}_k \) is the updated state at step \( k \), \( K_k \) is the kalman gain.

The process noise is approximated using

\[
R_k^v = (1 - \alpha_k)R_{k-1}^v + \alpha_k K_k(y_k - y_{k|k-1})(y_k - y_{k|k-1})^T K_k^T
\]  (4.23)

where \( R^v \) is the process noise covariance matrix. The process noise covariance is constrained to be a diagonal matrix \([6]\) since we are assuming additive and uncorrelated noise. The gain sequence is

\[
\alpha_k = \frac{a}{\alpha_{\text{min}} + k}
\]  (4.24)

where \( a \) and \( \alpha_{\text{min}} \) are constants that affects the initial value of the gain sequence and rate of decrease of magnitudes of gain values. \( a \) typically set to \( a = 1 \). \( \alpha_k \) is constrained subject to \( \alpha_k \leq \alpha_{\text{max}} \) so that the gain sequence does not go to zero but towards a small value that enables the filter to track drifts in process noise \([14]\) resulting from changes in operating conditions.

4.1.3 Estimation incorporating joint limits

The formulation of the forward dynamics and the UKF does not incorporate the physical limits imposed on the movement of the joints of the robot. Providing this information to the UKF would help in improving the convergence properties and stability of the filter. This is implemented based on
the method outlined for state estimation with constraints in \cite{12}, \cite{12} suggests projecting the sigma points of the prediction and update stages and the estimate onto the constrained space. This enables the constraint information to be incorporated into the apriori covariance and predicted estimates. An additional benefit is that this procedure is deterministic and hence suitable for real-time operation unlike some other types of constrained UKF which involve an optimization problem.
Algorithm 1 Algorithm for square root UKF from [25]

Initialize with:

\[
\hat{x}_0 = \mathbb{E}[x_0] \quad S_0 = \text{chol}\{\mathbb{E}[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]\} \quad (4.4)
\]

For \( k \in \{1, \ldots, \infty\} \),

Sigma point calculation and time update:

\[
\mathcal{F}_{k|k-1} = [\hat{x}_{k-1} \quad \hat{x}_{k-1} + \gamma S_k \quad \hat{x}_{k-1} - \gamma S_k]
\]  
\[
\mathcal{F}_{k|k-1}^+ = \text{F} [\mathcal{F}_{k|k-1}, u_{k-1}]
\]

\[
\hat{x}_k^- = \sum_{i=0}^{2L} W_t^{(m)} \mathcal{F}_{i,k|k-1}^- \quad (4.5)
\]

\[
S_k^- = \text{qr}\{[\sqrt{W_1^{(c)}} (\mathcal{F}_{k|k-1}^+ - \hat{x}_k^-) \quad \sqrt{R}]\}
\]  
\[
S_k^- = \text{cholupdate}\{S_k^-, \mathcal{F}_{0,k}^+ - \hat{x}_k^- , W_0^{(c)}\} \quad (4.8)
\]

\[
\mathcal{F}_{k|k-1} = [\hat{x}_k^- \quad \hat{x}_k^- + \gamma S_k^- \quad \hat{x}_k^- - \gamma S_k^-]
\]  
\[
H[\mathcal{F}_{k|k-1}^-] \quad (4.10)
\]

\[
\hat{y}_k^- = \sum_{i=0}^{2L} W_t^{(m)} \mathcal{Y}_{i,k|k-1}^- \quad (4.12)
\]

Measurement update equations:

\[
S_{y_k} = \text{qr}\{[\sqrt{W_1^{(c)}}(\mathcal{Y}_{k|k}\hat{y}_k) \quad \sqrt{R}]\}\quad (4.13)
\]

\[
S_{y_k} = \text{cholupdate}\{S_{y_k}, \mathcal{Y}_{0,k} - \hat{y}_k, W_0^{(c)}\} \quad (4.14)
\]

\[
P_{kjyk} = \sum_{i=0}^{2L} W_t^{(c)} [\mathcal{X}_{i,k|k-1} - \hat{x}_k^-][\mathcal{Y}_{i,k|k-1} - \hat{y}_k^-] \quad (4.15)
\]

\[
\mathcal{K}_k = \left(P_{kjyk}/S_{y_k}\right)/S_{y_k} \quad (4.16)
\]

\[
U = \mathcal{K}_k S_{y_k} \quad (4.17)
\]

\[
S_k = \text{cholupdate}\{S_k^- , U , -1\} \quad (4.18)
\]

where \( R^p \) is process noise cov., \( R^m \) is measurement noise cov.
Algorithm 2 Algorithm for UKF with constrained states from [12]

For $k = 1, 2, \ldots, \infty$:

1. Calculate $2N + 1$ sigma points based on the present state covariance using (4.5) and project the sigma points which are outside the feasible region to the boundary to obtain the constrained sigma points,

$$\mathscr{X}_{i,k-1}^C = P(\mathscr{X}_{i,k-1})$$

where $P$ is the projection.

2. Time update equations:

Transform the constrained sigma points through the state update functions (4.6) and apply the constraint on these transformed points.

$$\mathscr{X}_{k|k-1}^* = F[\mathscr{X}_{k|k-1}^C, u_{k-1}]$$

$$\mathscr{X}_{i,k|k-1}^* = P(\mathscr{X}_{i,k|k-1}^*)$$

Calculate the apriori state estimate (4.7) and apriori covariance (4.8, 4.9) using the constrained transformed sigma points $\mathscr{X}_{i,k|k-1}^* C$.

3. Measurement-update equations:

Transform the constrained sigma points through measurement function (4.11) and calculate the UKF state estimate from the measurement update by the same procedure as in Figure: 1) but with the constrained points. If the estimate violates the constraints, the same projection can be applied on it.
Chapter 5

STATE ESTIMATION AND CONTROL

5.0.4 Simulated robot

We tested the constrained UKF in a real-time simulation environment. The simulation used a FUSE (File system in user space)/CUSE (Character device in user space) device file based application running in real-time, mimicking the I/O of the real robot. The FUSE/CUSE [7, 24] framework running on Linux enables a user to create character device files in user space. The simulation application [Ch 2] captures input from the control software, simulates the forward dynamics and returns the system output. The integration of the dynamics was performed using CVODE/Sundials [8] solver. Both the control and simulation ran on the same machine (PREEMPT_RT patched Ubuntu 10.04 Linux (kernel 2.6.33.7.2-rt30) running on an Intel Core2Quad Q8300 2.5 Ghz 4Gb ram. The integrator had issues running in real-time during braking events on the robot. The data during these times were unused. Since the primary aim of this work was to study the feasibility of implementing the UKF for the robot, the parameters were chosen empirically. For example, the inertia parameters were obtained from the CAD model of the robot. The stiffness was chosen by scaling the stiffness value for a cable of known dimension. Although the parameters are not measured directly from the RAVEN, this choice of parameters should be a reasonable approximation for some nominal operating condition of the robot.

The full state information available from the simulation was compared against the estimates produced by UKF.

5.0.5 State estimation

The UKF was used to estimate the states of the simulated robot moving in free motion. The measurements were the motor angle and the inputs were the torques applied on the motor. The state estimates were compared against the true states available from the simulation. The process noise covariance and initial covariance were both initialised to $10^{-10}I$ and the measurement noise covari-
ance was set at $10^{-5} I$. The relative rms error was used as a metric to measure the accuracy of the UKF estimates. The relative rms error is calculated as

$$\text{RMS\%} = 100 \ \frac{\text{RMS(estimated error)}}{\text{RMS(actual value)}}$$

The state estimation performance was tested across different frequencies and amplitudes to study the change in performance at different frequencies. To this end, the desired trajectory was a pure sine function which was fed into a PD controller controlling the motor position. The same trajectory, with the amplitude normalized against the range of motion, was used on all the joints. However, since the dynamics of the robot change nonlinearly with joint configuration, the response of each joint differs even though the trajectories are similar. Due to the tracking error in the PD controller and the cable deformation, the actual joint ranges achieved would be different from the desired position. Additionally, a trajectory consisting of a sum of sines was used to evaluate the performance with a composite trajectory. Finally, the stiffness of the cables in the model used by the UKF is perturbed by 10% from the true value to study the effect of parameter perturbations on the performance.

5.0.6 Utilising estimates for control

The state estimates were used with a controller to study the performance of the UKF under closed loop control of the simulated robot. In this case, the control input to the motor is calculated based on the joint positions obtained from the state estimate instead of from the simulated motor positions directly. The desired trajectory was a signal consisting of a sum of sines.

PD control on Joints

This controller considers the system with the motor torques as the input and the joint positions obtained from the state estimates as the output. The controller is designed as if each motor and joint combination is an independent system, though in reality they are dynamically coupled. The gains for the controller are chosen by trial and error to obtain a combination that is stable and with low tracking error.
Approximate feedback linearization

In this case, we perform feedback linearization on the robot joints neglecting the deformation of the cable and assuming that the power is transmitted to the robot joints purely with gearing. This would give us three decoupled second order systems, one corresponding to each joint. The decoupling is achieved by choosing inputs as follows [13]:

For a rigid robot with torques $\Gamma$ acting on the joints,

\[
\ddot{q} = A(q)^{-1}[\Gamma - H(q, \dot{q})] \tag{5.1}
\]

\[
\Gamma = A(q)u + H(q, \dot{q}) \tag{5.2}
\]

\[
\ddot{q} = u \tag{5.3}
\]

$u$ is the input to the second order system $\ddot{q} = u$, where $q$ is a vector of joint positions and $u$ is a vector of inputs to the linearized system.

We close the loop around each of this linearized system with a PD controller with the gains chosen by trial and error. i.e.,

\[
\ddot{q} = u = kp(q_d - q) + kd(\dot{q}_d - \dot{q}) \tag{5.4}
\]

where, $q_d$ is the vector of desired joint positions and $\dot{q}_d$ is the vector of desired joint velocities. $kp$ and $kd$ are diagonal matrices of proportional and derivative gains of the PD controller respectively. The neglect of the cable deformation can be considered as a modelling error on the system. The PD controller should provide some robustness against this modelling error especially at lower and medium frequencies. However, at higher frequencies and higher amplitudes, the degree of freedom introduced by the cable becomes more pronounced and may eventually lead to poor performance due to the changes in magnitude and phase of the torque that is acting on the joint. Furthermore, the performance is also affected due to the fact that this decoupling input considers a continuous time system, whereas the inputs applied to the system are in discrete time steps.
Figure 5.1: Illustration of PD control on joint. $q_l$ and $q_{ld}$ are the joint positions and desired joint positions respectively. $\dot{q}_l$ and $\dot{q}_{ld}$ are the joint velocities and desired joint velocities respectively. $\tau_m$ is the torque applied on the motor. $q_m$ is the measured motor position. $\dot{q}_m$ is the motor velocity.
Figure 5.2: Illustration of PD control on joint with approximate feedback linearization. $q_l$ and $q_{ld}$ are the joint positions and desired joint positions respectively. $\dot{q}_l$ and $\dot{q}_{ld}$ are the joint velocities and desired joint velocities respectively. $\tau_m$ is the torque applied on the motor. $q_m$ is the motor position. $\dot{q}_m$ is the motor velocity. $A$ is the inertia matrix of the robot. $H$ consists of centrifugal, Coriolis, frictional and gravitational components. $tr$ is the transmission ratio of the gearbox from the motor to the joint capstan.
Chapter 6

RESULTS

6.1 Results-Simulation

To study the performance of the UKF at various frequencies, pure sine reference signals were used. The frequencies of the reference signal were related such that the product of amplitude and frequency squared was constant (proportional to second derivative and hence torque). Frequency was 5 Hz at 90% maximum possible amplitude and 0.125 Hz at 10% maximum amplitude for each joint. The reference signal was used with a PD controller on the motor so that the controller acts only on truly known states and the estimation was independent of control. As a consequence of this choice, the controller neglects cable deformation and so, the response of the the joint states would be different at different frequencies of the reference signal.

The estimation performance of the UKF was also studied with two controllers using the UKF estimates as described in the previous section. The desired trajectory for the controllers was a sum of sines and were kept same for both the controllers. These controllers were engaged after performing the test at multiple frequencies as described earlier. This had the benefit that controllers used the state estimates that had converged. The performance across different controllers were not compared since such a comparison would require tuning all the controllers for optimal performance.

In each case, i.e. estimation at various frequencies and estimation with controller using the estimates, the stiffness programmed in the system model used by the UKF was perturbed by 10%. The stiffness parameter was chosen for perturbation as it has a significant effect on the dynamics of the robot compared to other parameters. Additionally, the stiffness in real world use tends to change over time due to effects such as plastic deformation, creep, variations over temperature, repeated use, age etc.
Figure 6.1 shows a snapshot of the state estimation. The joint positions obtained from the motor positions kinematically is also shown to illustrate the effect of cable deformation. The UKF is able to provide better estimates of joint angles in the presence of cable deformation. Table 6.1 shows the relative rms error of the estimation at different frequencies with the perturbation in stiffness.

Table 6.2 shows the rms error of the tracking for two different tracking controllers (Approximate decoupling control + PD control, PD control on joint based on UKF estimates). Figs 6.2 shows a snapshot of the desired trajectory and tracking. The table shows the controller performance under perturbations in stiffness. The desired trajectory was a sum of sines. Table 6.3 also shows the performance of the UKF under closed loop with these tracking controllers.

The velocity estimates of the joints are shown in Figure 6.3 and Figure 6.4 for correct stiffness parameters and 0.9x the correct stiffness parameters respectively. The figures show that the velocity estimates track the actual joint velocities even with some uncertainty in the model parameter. Figure 6.5 shows the velocity estimation for the motor. The velocity estimation for the motor states were poor. A similar effect was observed on motor velocities corresponding to remaining joints.
<table>
<thead>
<tr>
<th>Ref sine frequency (Hz)</th>
<th>0.125</th>
<th>1.1</th>
<th>2.075</th>
<th>3.05</th>
<th>4.025</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ref Ampl</td>
<td>0.9R</td>
<td>0.46R</td>
<td>0.28R</td>
<td>0.19R</td>
<td>0.13R</td>
<td>0.1R</td>
</tr>
</tbody>
</table>

**Joint**

<table>
<thead>
<tr>
<th>Correct parameters</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Shoulder</td>
<td>0.13</td>
<td>0.71</td>
<td>1.18</td>
<td>2.64</td>
<td>1.24</td>
<td>1.75</td>
</tr>
<tr>
<td>Elbow</td>
<td>0.04</td>
<td>0.11</td>
<td>0.13</td>
<td>0.22</td>
<td>0.18</td>
<td>0.12</td>
</tr>
<tr>
<td>Z axis ins</td>
<td>0.04</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td>0.10</td>
<td>0.08</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1.1x correct stiffness parameters</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Shoulder</td>
<td>0.32</td>
<td>3.92</td>
<td>1.85</td>
<td>2.37</td>
<td>3.05</td>
<td>4.10</td>
</tr>
<tr>
<td>Elbow</td>
<td>0.12</td>
<td>0.53</td>
<td>0.19</td>
<td>0.20</td>
<td>0.39</td>
<td>0.40</td>
</tr>
<tr>
<td>Z axis ins</td>
<td>0.04</td>
<td>0.06</td>
<td>0.05</td>
<td>0.06</td>
<td>0.05</td>
<td>0.05</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0.9x correct stiffness parameters</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Shoulder</td>
<td>0.41</td>
<td>2.73</td>
<td>1.97</td>
<td>2.12</td>
<td>1.84</td>
<td>2.81</td>
</tr>
<tr>
<td>Elbow</td>
<td>0.17</td>
<td>0.35</td>
<td>0.23</td>
<td>0.20</td>
<td>0.20</td>
<td>0.32</td>
</tr>
<tr>
<td>Z axis ins</td>
<td>0.04</td>
<td>0.06</td>
<td>0.06</td>
<td>0.05</td>
<td>0.05</td>
<td>0.04</td>
</tr>
</tbody>
</table>

Table 6.1: Relative RMS Error(%) for different frequencies and amplitudes with perturbations in the cable stiffness parameter of the model. The amplitudes are based on the maximum amplitude possible, R from the midrange point for each joint.
### Joint Shoulders Elbow Z axis ins

<table>
<thead>
<tr>
<th>Controller</th>
<th>Correct parameters</th>
<th>1.1x correct stiffness parameters</th>
<th>0.9x correct stiffness parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approx decoupling+PD</td>
<td>0.03 0.04 0.006</td>
<td>0.02 0.04 0.005</td>
<td>0.03 0.04 0.006</td>
</tr>
<tr>
<td>PD control on joint</td>
<td>0.04 0.05 0.005</td>
<td>0.04 0.05 0.005</td>
<td>0.05 0.05 0.005</td>
</tr>
</tbody>
</table>

Table 6.2: RMS Tracking Error (radians, metres for Z axis ins) of the controllers tracking a sum of sines trajectory using UKF joint estimates under perturbations in the cable stiffness parameter of the model.

### Joint Shoulder Elbow Z axis ins

<table>
<thead>
<tr>
<th>Controller</th>
<th>Correct parameters</th>
<th>1.1x correct stiffness parameters</th>
<th>0.9x correct stiffness parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approx decoupling+PD</td>
<td>0.20 0.04 0.03</td>
<td>0.51 0.07 0.04</td>
<td>1.21 0.21 0.04</td>
</tr>
<tr>
<td>PD control on joint</td>
<td>0.15 0.03 0.04</td>
<td>0.42 0.05 0.05</td>
<td>0.70 0.11 0.05</td>
</tr>
</tbody>
</table>

Table 6.3: Relative RMS Error (%) of the UKF position estimates under perturbations in the cable stiffness parameter of the model with closed loop control.
Figure 6.1: Plot showing the actual position, position estimates and joint position calculated kinematically by neglecting cable deformation for the shoulder, elbow and z insertion when the reference trajectory is a sine of 3.05 Hz with motor PD control: Correct stiffness parameters
Figure 6.2: Plot showing the tracking using the UKF estimate with correct parameter and approximate feedback linearization and PD control on linearized system.
Figure 6.3: UKF Velocity estimates of the shoulder, elbow and z insertion when the reference trajectory is a sine of 3.05 Hz with motor PD control not using UKF estimates: Correct stiffness parameters
Figure 6.4: UKF Velocity estimates of the shoulder, elbow and z insertion when the reference trajectory is a sine of 3.05 Hz with motor PD control not using UKF estimates: 0.9x Correct stiffness parameters.
Figure 6.5: Velocity estimates of the motor for the shoulder joint when the reference trajectory is a sine of 3.05 Hz with motor PD control: Correct stiffness parameters. The velocity estimates are poor possibly due to the truncation error in the ERK step.
Chapter 7

DISCUSSION

7.1 Simulation

The work in this paper attempts to create a simulation system so that the control software is unaware of the reality of the system it is trying to control. The development of this system also tries to keep as much development as possible in the userspace. The benefits of this arrangement are

- Accessibility to sophisticated debugging tools
- Reduced chance of destabilizing the software system
- Access to libraries that ease the development of complicated algorithms.

Our 1kHz sampling rate gives us about 1ms to do the communication and simulation. It is indeed possible that the robot may end up in certain configurations where it may be not possible to finish the simulation within the sampling time, for example inputs and/or parameter combinations that induce large high frequency transients. However, besides such edge cases, the simulation gives a system designer largely high fidelity simulation environment to test their system.

Another motivating factor in the design of this system is to test new control and estimation algorithms in a safe but reasonably realistic environment. For example, the use of cable driven power transmission in our robots runs the risk of vibrations induced in the robot due to elasticity in the transmission. The simulation system provides a means to study such effects and the control system’s response to such non linear behavior.

In state estimator and controller design, we may want to perform state and/or parameter estimation for monitoring or for use with a controller. Although tools exist to calculate nonlinear observability,
controllability etc, they are often intractable as the system size and complexities increase. Furthermore, analyzing the effects of discretization inherent in digital control systems on the system response is often non trivial. A simulation would be helpful in obtaining an intuitive understanding of the system’s response to different controllers and estimators. Since all the states and parameters of a simulation are known completely, it is possible to evaluate, test and debug estimation and control algorithms (assuming the system model is a reasonable approximation to the real life system).

In contrast to many other popular simulation systems that use a physics engine to model the rigid body dynamics, our implementation uses a device specific (forward dynamic model of the robot) dynamic model. This choice was made for the sake of simplicity, since we could optimize the code for our system model. This choice also potentially reduces any unexpected sources of latency from using software intended for use on generic and often complex systems. A disadvantage of this approach is the absence of many features standard in physics engine such as contact and environment interaction. The architecture of the system however does not prohibit the use of other software to compute the dynamics.

Our implementation uses unix read, write and ioctl system calls for data communication and device control. The custom kernel space driver for the USB I/O board creates a device file corresponding to each USB board connected to the computer. Such a device file based I/O enables the use of the device from different programs and from different programming languages without considering the details of the USB communications protocols. Although this I/O format is suitable for a large number of applications, there might still be some applications where this I/O format may be insufficient, for example on platforms that don’t support device files. In those cases, a different framework to emulate the communication device would be required.

### 7.2 State estimation

Table 6.1 shows the state estimation performance for different frequency signals. The table shows that the estimation error worsens at higher frequencies. One possible cause of this behaviour is that the ERK discretization becomes inaccurate at higher frequencies. The effect of a 10% perturbation
in UKF model parameter on the estimation error is marginal. This means that if the parameters pro-
grammed into the UKF differed from the true values by 10%, the performance is not significantly
affected. We have not attempted to study the effect of parameter perturbation comprehensively at
this point since the estimation error depends also on the frequency content of the signal. Such a
study would need to have a method to decouple the effect of state trajectory from the effect of the
parameter. Additionally, such a study would have to compensate for the inherent sensitivity of the
dynamics of the model to parametric changes.

The accuracy and performance of the UKF depends on good process noise information. The UKF
either diverged or produced incorrect estimates with a fixed pre-programmed process noise far away
from the true values. Guessing process noise values becomes harder as the size of the system grows.
The Robbins Monro approximation method provides good guesses for process noise. However, even
with this correction biased and noisy estimates were observed on the motor velocity estimates.

We performed preliminary numerical experiments with a simulation of the pulleyboard, the 1 DOF
simplified model of the RAVEN power transmission. These indicated that the UKF accuracy is sen-
sitive to the truncation error present in the ERK integration. The system has large inertias on one
side (joint, with relatively large links) of the elastic transmission compared to the other (motor). The
ODE describing the system becomes numerically stiff at some configurations due to this difference
in inertia, nonlinear frictional and elastic transmission components etc. The ERK integration seems
to produces inaccurate results (esp motor velocity state) while integrating the sigma points over
time. Further research is required in studying the methods of discretization on the UKF estimation
accuracy.

Since the accuracy of the UKF depends on the programmed process noise, it is possible that the
Robbins-Monro approximation for process noise may converge to a set of values that compensate
the ERK truncation effects, but it is not observed to happen in our case. It is not clear if this ap-
proach can compensate for the truncation error of the ERK applied on elastic power transmission
based systems.
Table 6.2 shows the sum of sines reference tracking performance of the two controllers using state estimates. Table 6.3 shows the estimation performance under closed loop control. It can be seen that the estimation performance is not significantly affected with a controller using the estimates. The control performance was poor when the frequency content of the reference trajectory was increased beyond 1.5 Hz. It should be noted that these controllers were not tuned for maximum performance. Furthermore, the two controllers shown here are examples and the cable deformation is an unaccounted modelling error. It is possible to utilize the state estimates to design better controllers that account for the cable deformation in the calculation of the control input.

7.3 Conclusion

In this work, we have demonstrated state estimation on a simulated surgical robot with flexible cable power transmission using the Unscented Kalman Filter (UKF). Since it can be more practical (due to size, wiring, power, sterilization, complexity etc) to sense motion at the motor rather than at the joint on the other end of the flexible cable transmission, a typical control system will not be aware of motion error due to flexibility between the motor and joint. The UKF provided better estimates of the joint states than those obtained by neglecting the cable deformation and robot dynamics. The process noise information in the UKF affects the convergence and accuracy. As the number of states of the system increases, it is difficult to guess the process noise information and so we have employed the Robbins Monro stochastic approximation scheme to enable the UKF to automatically tune the process noise information. Additionally, we tested the UKF estimation with two different controllers using the UKF estimates to close the control loop. It was also found that the truncation error in the integration method used for discretization affects the estimates and further studies about suitable integration methods are required. We have also implemented a real-time simulation system based on device-file based I/O using the FUSE/CUSE library. This enables the control software to be run with almost no additional modification on either the simulation or the real world system.

We hope that our experience with this work would be useful to control system designers, especially for surgical robots and cable based power transmission mechanisms. The implementation process for the UKF described hopefully provides pointers and pitfalls for the designer who wishes to use
the UKF for state estimation. The simulation system that we have developed should help developers who are looking for an unified way to test a system with simulation and realworld systems.
Chapter 8

FUTURE WORK

8.1 Simulation

One of the limitations of this system is that the computation that can be done within the sampling time is limited. Although techniques such as parallel processing can be used, the requirement of deterministic timing means that the overhead associated with parallel processing may become prohibitive at higher sampling rates. With increasing processing power and technologies such as GPGPU maturing with lower latency access, low latency network communication etc, it may be possible to perform a large number of additional computation within the same sampling time. This opens up the possibility of more realistic physical interaction models. For example, in this case of the surgical robot, we may be able to simulate the interaction of the robot with nonlinear tissues. Another potential application of such a system would be to increase the reliability of the control system running on the robot. Traditional analysis techniques often provide some indication of the performance limits of the system. But, with increased processing power, it would be possible to design many test cases similar to that performed in many other fields to detect edge cases of system failure. For example, in the case of surgical robot, it would be possible to model the deterioration of the cable, change in friction properties of the system etc to detect the cases where the system fails.

8.2 State estimation

Presently, the square root UKF that we implement uses $2N + 1$ sigma points for $N$ number of states. To reduce the computational cost, the simplex unscented transform \[10\] can be used. This can reduce the number of dynamics function evaluations from $2N + 1$ to $N + 1$ thus providing more CPU time for other control computation tasks.

We have noticed that the estimation error is sensitive to the ERK truncation error. Using higher order methods or fixed step implicit integration methods may reduce truncation errors and improve
the estimates further. An evaluation of different methods and their relative computation cost is required to improve the performance of the UKF.

In this work, the estimation is performed with a simulated robot. We plan to incorporate the lessons from this work in the implementation of the UKF with the real robot. In addition to this, the parameter estimation, for monitoring the robot and for adaptive control would be implemented. Another task would be in the online parameter identification of the tissue that the robot interacts with and modifying the behavior of the robot suitably.
BIBLIOGRAPHY


