2018-08-01

Model Predictive Linear Control with Successive Linearization

Jesse Robert Friedbaum
Brigham Young University

Model Predictive Control with Successive Linear Approximation on Robotic Systems

Jesse Robert Friedbaum

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mathematics

Brigham Young University

ABSTRACT

Model Predictive Control with Successive Linear Approximation on Robotic Systems

Jesse Robert Friedbaum
Department of Mathematics, BYU
Master of Science

Robots have been a revolutionizing force in manufacturing in the 20^{th} and 21^{st} centuries but have proven too dangerous around humans to be used in many other fields including medicine. We describe a new control algorithm for robots developed by the Brigham Young University Robotics and Dynamics and Robotics Laboratory that has shown potential to make robots less dangerous to humans and suitable to work in more applications. We analyze the computational complexity of this algorithm and find that it could be a feasible control for even the most complicated robots. We also show conditions for a system which guarantee local stability for this control algorithm.

Keywords: control theory, robotics, numerical analysis, model predictive control, mpc

Acknowledgments

I would like to express my deepest gratitude to all of my advisory committee members and especially my advisor Ben Webb for their guidance and more importantly their patience with me. I would also like to thank my office mate Shane McQuarrie for always having the answers to my constant questions about $\LaTeX$ and for slogging through the equations of motion for the two link planar robot found later in this thesis. Finally, I would like to thank Marc Killpack, Phillip Hyat, Jon Terry and Levi Rupert of the BYU Robotics and Dynamics Laboratory for patiently explaining and re-explaining their work.

Contents

Contents **iv**

List of Figures **vi**

1 Introduction **1**

2 Robotics **2**

2.1 Hard Robots 2

2.2 Soft Robots 6

3 Control Theory **8**

3.1 Technical Details of Control Theory 9

3.2 PD control 11

3.3 Model Predictive Control 12

3.4 New Control Method 13

4 Introduction of Linearization Techniques **15**

4.1 Fixed State 16

4.2 Coupling Torque 17

4.3 Proofs of Linear Approximations 18

5 Comparing Linearization Technique Performance **20**

5.1 Introdution of Genetic Algorithms 21

5.2 Comparing Computational Complexity with Genetic Algorithms 23

6 Proof of New Control Method **27**

6.1 Global Instability 28

6.2 Continuity Results 29

6.3 Local Stability 38

7 Conclusion **40**

7.1 Future Work 40

Bibliography **41**

List of Figures

1.1 Industrial robot kept in a cage to prevent injuries from humans. 2

2.1 The *Baxter Robot* from Rethink Robotics with 2 joints labeled on one arm. Used with permission of BYU RAD lab 3

2.2 Diagram of a 2-joint robot 4

2.3 The *King Louie* soft robot from the BYU RAD Lab used with permission of the RAD lab 6

6.1 Phase plot of equation (6.1) 28

Chapter 1. Introduction

Robots have been a revolutionizing force in many industries in the 21$^{\text{st}}$ century, with the number of industrial robots projected to exceed three million by the year 2020, which represents a doubling of the stock within a 7-year period [1]. Unfortunately, because current robot technologies make them very dangerous to human bystanders, robots have primarily only been able to work on tasks that include very little human interaction, often being placed in cages in order to avoid contact with humans, see figure 1.1. Indeed, over 70% of robot related accidents reported by the United States Department of Labor were fatal [2]. Regardless, there is an increasing need for robots in areas which require close human interaction particularly in medical and nursing care due to the aging populace in the United States and many other parts of the world [3].

There are two principle ways to make robots more safe for human interaction. The first approach is software based and involves developing new control algorithms so that traditional "hard robots" with rigid parts can move around humans in such a way that they are not likely to cause serious injury. The second approach is hardware based and involves the creation of new "soft robots" such as the King Louis robot in the BYU Robotics and Dynamics (RAD) Laboratory shown in Figure 2.3. King Louis is built to be compliant so that it may be pushed off of its course, making it less likely to injure a human even in the event of an unplanned collision. Regrettably, the dynamics dictating the movement of such soft robots are far more complicated than their hard counter parts and traditional control algorithms have proven ineffective on these new robots [4]. Hence both methods for making robots safe around humans will require the development of improved control algorithms. Here we will examine mathematically a novel control algorithm developed by the BYU RAD Lab that has already proven effective in controlling soft robots [4]. We give sufficient conditions on the robotic system that guarantee the local stability of this new algorithm and examine its computational advantages over traditional techniques.

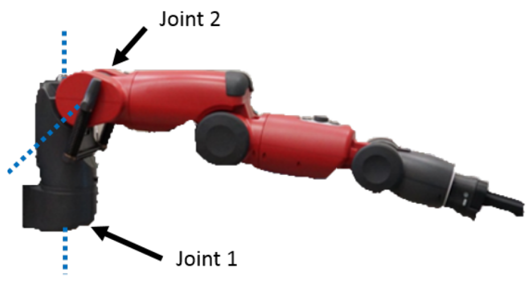
Figure 1.1: Industrial robot kept in a cage to prevent injuries from humans.

Chapter 2. Robotics

2.1 Hard Robots

We will begin by giving a brief explanation of robotic systems and the equations that govern their evolution in time. We define a robot as a series of rigid elements called *links* connected together end to end. We will call the connection between links a *joint*. Each joint has a single axis around which the two links attached to the joint may rotate relative to each other. We will call the angle at which the two links are rotated relative to each other the *bend* in the joint. Figure 2.1 shows the *Baxter Robot* from Rethink Robotics with two of its joints labeled. Throughout this thesis we will also assume one end of this robot is fixed to the reference frame i.e. the base of the robot is fixed in space. For traditional robots (as opposed to soft robots), we assume that there is an actuator motor at each joint capable of applying torque to that joint. Instead of describing the position of a robot by the position of its joints and links in space we will describe the position of a robot by the angle at which each of its joints are bent. We call this description *configuration space* and note that it has

(a) Full Robot



(b) Joints labeled

Figure 2.1: The *Baxter Robot* from Rethink Robotics with 2 joints labeled on one arm. Used with permission of BYU RAD lab

the advantage of being describable with a simple vector $\mathbf{q}(t) \in \mathbb{R}^n$ which contains the angle at which each of the joints on an n jointed robot is bent at time t. It is a simple matter to find the locations of the links and joints of a robot in space when we know the configuration space description of the robot.

Newtonian physics gives us a model of our robot's movement:

$$D(\mathbf{q})\ddot{\mathbf{q}} = C(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + F(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) + \tau \tag{2.1}$$

where $D(\mathbf{q}) \in \mathbb{R}^{n\times n}$ is a generalized mass matrix which represents the resistance of the robot to acceleration and depends solely on the joint angles $\mathbf{q}$. In this equation $C(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^{n\times n}$ is the matrix of Christoffel symbols. The vector $C(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}$ gives the Coriolis forces acting on each joint which are a naturally occurring phenomenon in rotating systems. We note that this vector not only depends on the position of the robot but also its velocity. Next, $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^n$ is the torque which gravity applies to each of the joints of the robot which again depends on the robot's current position. Each of the terms explained above will typically contain long sums, products and compositions of various trigonometric functions which grow much more complex with the addition of each joint to the robot. To illustrate this, we will

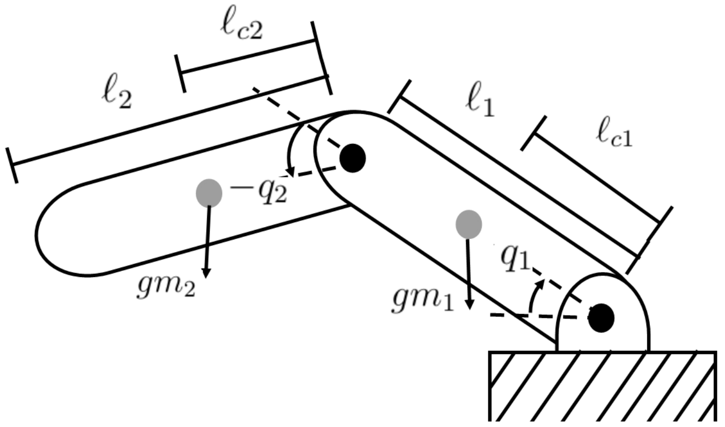


Figure 2.2: Diagram of a 2-joint robot

examine the equations of motion for the simple two joint robot in figure 2.2. We will leave out $F(\mathbf{q}, \dot{\mathbf{q}})$ in these equations as we have not yet discussed its meaning. The following table defines all the terms needed for these equations of motion.

Variable	Meaning
q_1	Bend in joint 1
τ_1	Torque at link 1
m_1	Mass of link 1
ℓ_1	Length of link 1
ℓ_{c1}	Distance from joint 1 to the center of mass of link 1
I_{z1}	Moment of inertia of link 1 about its center of mass with respect to the z axis
q_2	Bend in joint 2
τ_2	Torque at link 2
m_2	Mass of link 2
ℓ_2	Length of link 2
ℓ_{c2}	Distance from joint 1 to the center of mass of link 1
I_{z1}	Moment of inertia of link 2 about its center of mass with respect to the z axis

The equation of motion for the robot consisting of only joint one in the figure is simply

$$\begin{bmatrix} m_1 l_{c1}^2 + I_{z1} \end{bmatrix} \cdot \begin{bmatrix} \ddot{q}_1 \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix} \cdot \begin{bmatrix} \dot{q}_1 \end{bmatrix} + \begin{bmatrix} m_1 l_{c1} g \cos(q_1) \end{bmatrix} + \begin{bmatrix} \tau_1 \end{bmatrix}.$$

However, when considering the entire robot depicted in the figure, the equations of motion become

$$\begin{aligned} &\begin{bmatrix} m_1 l_{c1}^2 + m_2(l_1^2 + l_{c2}^2 + 2l_1 l_{c2}\cos(q_2)) + I_{z1} + I_{z2} & m_2(l_{c2}^2 + l_1 l_{c2}\cos(q_2)) + I_{z2} \\ m_2(l_{c2}^2 + l_1 l_{c2}\cos(q_2)) + I_{z2} & m_2 l_{c2}^2 + I_{z2} \end{bmatrix} \cdot \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} \\ &= \begin{bmatrix} -m_2 l_1 l_{c2}\sin(q_2)\dot{q}_2 & -m_2 l_1 l_{c2}\sin(q_2)(\dot{q}_1 + \dot{q}_2) \\ m_2 l_1 l_{c2}\sin(q_2)\dot{q}_1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \\ &+ \begin{bmatrix} (m_1 l_{c1} + m_2 l_1) g\cos(q_1) + m_2 l_{c2} g\cos(q_1 + q_2) \\ m_2 l_{c2} g\cos(q_1 + q_2) \end{bmatrix} + \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}. \end{aligned} \tag{2.2}$$

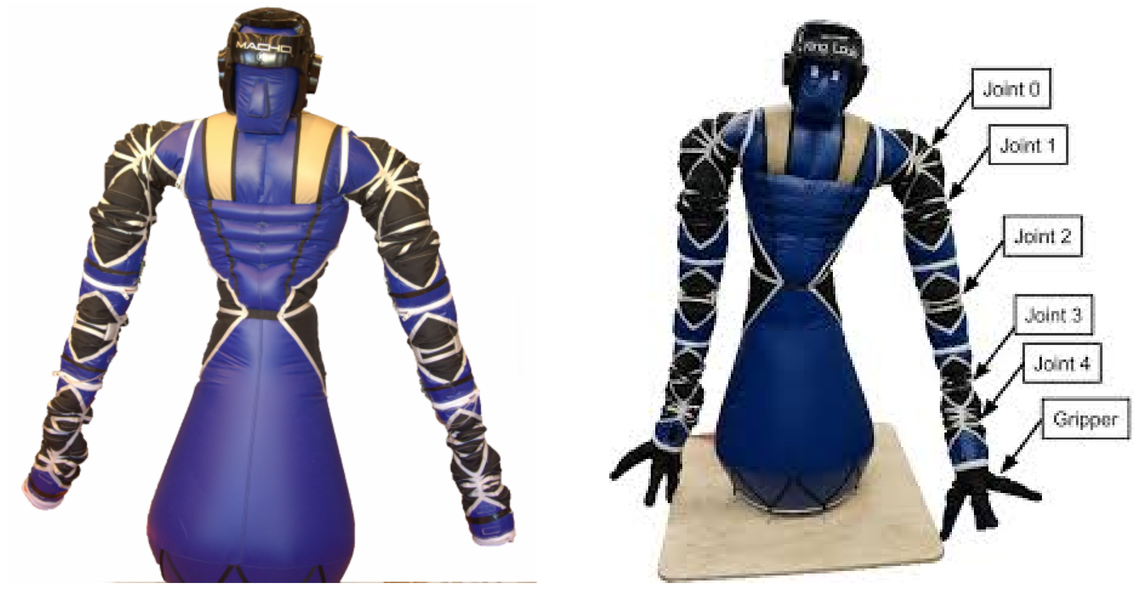
A similar jump in complexity can be expected for the inclusion of each additional joint.

The vector $F(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^n$, which we previously ignored, represents the frictional forces on our robot which result from complicated interactions between the materials inside the joints of the robot. This friction could be modeled as Coulomb friction, viscous friction, aerodynamic friction, or any combination of the preceding. In order to keep our system tractable we will model this friction only as viscous friction. That is, we set

$$F(\mathbf{q}, \dot{\mathbf{q}}) = F_v \dot{\mathbf{q}}$$

where $F_v \in \mathbb{R}^{n \times n}$ is a constant diagonal matrix with nonnegative entries representing the frictional coefficient for each joint.

Finally, the vector $\tau(t) \in \mathbb{R}^n$ represents the torques applied by the actuator motors at each joint at time t. We will assume we can choose the value of this quantity at any time or "control this quantity directly". This is the only part of the robot we may control directly.



(a) Full Robot (b) Joints labeled

Figure 2.3: The *King Louie* soft robot from the BYU RAD Lab used with permission of the RAD lab

It is useful to reformulate (2.1) as a first-order differential-equation. To do this we define the vector $\mathbf{x} \in \mathbb{R}^{2n}$ by

$$\mathbf{x} = \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix}$$

in which case (2.1) can be rewritten as

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{0} & -D^{-1}(\mathbf{q})(C(\mathbf{q}, \dot{\mathbf{q}}) + F_v) \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{x} + \begin{bmatrix} D^{-1}(\mathbf{q}) \\ \mathbf{0} \end{bmatrix} \mathbf{u} \tag{2.3}$$

where $\mathbf{u} = \mathbf{g}(\mathbf{q}) + \tau$. Note that any mass matrix $D(\mathbf{q})$ coming from a real physical system will be positive definite and therefore D^{-1} will exist.

2.2 Soft Robots

The principle difference between soft and hard robots is that, instead of using actuator motors to move the joints, soft robots contain two inflatable bladders for each joint. As they inflate, one bladder applies a torque to the joint in the positive direction and the other applies a torque in the negative direction. Because air is compressible this allows the robot's joints to be forced backwards if they come in contact with a person or other unexpected object. The drawback to this design is that we can no longer pick the torques to apply to each joint directly. Instead the torque at the joints is given by the equation

$$\tau = S\mathbf{q} + \Gamma^{+}\mathbf{p}^{+} - \Gamma^{-}\mathbf{p}^{-} \tag{2.4}$$

where $S \in \mathbb{R}^{n\times n}$ is a stiffness matrix that represents the tendency of the robot to snap back to its resting position. The vectors $\mathbf{p}^{+}$ and $\mathbf{p}^{-} \in \mathbb{R}^{n}$ are the pressures in the two air bladders that apply torques to the joints in the positive and negative direction respectively. Both $\Gamma^{+}, \Gamma^{-} \in \mathbb{R}^{n\times n}$ are constant diagonal matrices which represent how strongly the air pressure in each bladder affects the joint to which it is attached.

Unfortunately, we are not even able to choose the values of $\mathbf{p}^{+}$ and $\mathbf{p}^{-}$ directly. Instead we may only directly control the pressure of the air in the tube that fills and drains each bladder. We will refer to this pressure as $\mathbf{p}^{+}_{in}$ and $\mathbf{p}^{-}_{in}$. This effects $\mathbf{p}^{+}$ and $\mathbf{p}^{-}$ through the following differential equation

$$\dot{\mathbf{p}}^{+} = \boldsymbol{\alpha}_{+}\mathbf{p}^{+} + \boldsymbol{\beta}_{+}\mathbf{p}^{+}_{in} \tag{2.5}$$

$$\dot{\mathbf{p}}^{-} = \boldsymbol{\alpha}_{-}\mathbf{p}^{-} + \boldsymbol{\beta}_{-}\mathbf{p}^{-}_{in} \tag{2.6}$$

where $\boldsymbol{\alpha}_{+}, \boldsymbol{\alpha}_{-}, \boldsymbol{\beta}_{+}$ and $\boldsymbol{\beta}_{-}$ are $n \times n$ constant diagonal matrices.

The dynamics of a soft robot is then described by the system of ODE's consisting of (2.5), (2.6) and

$$D(\mathbf{q})\ddot{\mathbf{q}} = C(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + F_v\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + S\mathbf{q} + \Gamma^{+}\mathbf{p}^{+} - \Gamma^{-}\mathbf{p}^{-}. \tag{2.7}$$

We can rewrite this as a first order ODE by defining

$$\mathbf{y} = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{q} \\ \mathbf{p}^+ \\ \mathbf{p}^- \end{bmatrix}, \mathbf{u} = \begin{bmatrix} \mathbf{p}^+_{in} \\ \mathbf{p}^-_{in} \end{bmatrix}.$$

We then arrive at the system

$$\dot{\mathbf{y}} = \begin{bmatrix} -D^{-1}(\mathbf{q})S & -D^{-1}(\mathbf{q})(C(\mathbf{q},\dot{\mathbf{q}}) + F_v) & D^{-1}(\mathbf{q})\Gamma^+ & D^{-1}(\mathbf{q})\Gamma^- \\ \mathbf{0} & I & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\alpha}_+ & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\alpha}_- \end{bmatrix} \mathbf{y}$$
$$+ \begin{bmatrix} D^{-1}(\mathbf{q})\mathbf{g}(\mathbf{q}) \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \boldsymbol{\beta}_+ & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\beta}_- \end{bmatrix} \mathbf{u}. \tag{2.8}$$

Now that we have described the systems with which we will be working we are now prepared to discuss the tasks we wish to perform on these systems.

Chapter 3. Control Theory

We now give a brief introduction to control theory and discuss the specific difficulties of controlling robotic systems. Principally control theory asks: If we have a dynamical system (such as the movement of a robotic arm) and we control one aspect of the system (such as the torque produced by the actuator motors of that robot), is it possible to choose the aspect of the system we can control so that the entire system moves to some desired state? This is usually possible with robotic systems, but finding the right choice for the part of the system we can control can be very challenging, particularly for soft robots. We will discuss several well known control methods and one recently developed method from the BYU RAD Lab that is uniquely effective with soft robots.

3.1 Technical Details of Control Theory

Suppose we have two distinct points in $\mathbb{R}^n$, $\mathbf{x}_0$ and $\mathbf{x}_{goal}$. (Although the principles of control theory need not be limited to $\mathbb{R}^n$, we will limit ourselves to $\mathbb{R}^n$ because this is the space most applicable to robotics.) We also have either a continuous-time first order differential-equation

$$\dot{\mathbf{x}} = f(t, \mathbf{x}, \mathbf{u}) \tag{3.1}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0 \tag{3.2}$$

or a discrete-time dynamical system

$$\mathbf{x}_{n+1} = f(n, \mathbf{x}_n, \mathbf{u}). \tag{3.3}$$

In (3.1) and (3.3) $\mathbf{u}$ is a function from $\mathbb{R}^m$ to $\mathbb{R}^k$ in which $m = n$ or $m = 1$. The function $\mathbf{u}$ is called the *control* for our system and we will assume we have the freedom to decide what this function is. With f, $\mathbf{x}_0$, and $\mathbf{x}_{goal}$ known, we wish to choose $\mathbf{u}$ so that $\mathbf{x}$ is driven to $\mathbf{x}_{goal}$. If we pick $\mathbf{u}$ to be solely a function of t or a function of n, for the continuous and discrete cases respectively, our control is known as an *open loop control.* If we pick $\mathbf{u}$ to be a function of $\mathbf{x}$, our control is known as a *closed loop control.* Generally speaking, closed loop controllers are preferable to open loop controllers because they tend to have a property know as *robustness.* A control is robust if it is effective at not only moving the system for which it was designed to the desired location, but is also effective for slightly perturbed versions of that system. Open loop controllers cannot adapt to perturbed systems. On the other hand, if a closed loop controller is applied to a perturbed system and the perturbance causes the system to move in an unexpected way, then the new location will be fed back into the controller giving it a chance to modify its output to better fit the new system.

If $\mathbf{x}_{goal}$ is a stable fixed point for $\dot{\mathbf{x}} = f(t, \mathbf{x}, \tilde{\mathbf{u}}(t))$ or $\mathbf{x}_{n+1} = f(n, \mathbf{x}_n, \tilde{\mathbf{u}}(n))$ for some control $\tilde{\mathbf{u}}(t)$ or $\tilde{\mathbf{u}}(n)$ respectively, we call the control $\tilde{\mathbf{u}}$ *stable*. Similarly, if $\mathbf{x}_{goal}$ is an asymptotically stable fixed point, we call the control *asymptotically stable*.

It is often useful to have a deterministic way to compare the performance of different possible controls. We do this by creating a *cost functional* $J(\mathbf{x}, \mathbf{u})$ which assigns a score to any possible control. The typical forms of a cost functional are

$$J(\mathbf{x}, \mathbf{u}) = \int_{t_0}^{\infty} (\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u}) dt \tag{3.4}$$

$$J(\mathbf{x}, \mathbf{u}) = \int_{t_0}^{t_{final}} (\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u}) dt + \mathbf{x}(t_{final})^T Q \mathbf{x}(t_{final}) \tag{3.5}$$

where Q and R are positive-definite matrices. Equation (3.4) is known as an *infinite time horizon* cost functional and (3.5) is an *finite time horizon* cost functional. With a cost functional it is often possible to find a unique *optimal control* $\mathbf{u}^*(t)$, which minimizes the value of the cost functional. This also applies to discrete-time control problems with the discrete-time cost functionals

$$J(\mathbf{x}, \mathbf{u}) = \sum_{i=0}^{\infty} (\mathbf{x}_i^T Q \mathbf{x}_i + \mathbf{u}_i^T R \mathbf{u}_i) \tag{3.6}$$

$$J(\mathbf{x}, \mathbf{u}) = \sum_{i=0}^{N} (\mathbf{x}_i^T Q \mathbf{x}_i + \mathbf{u}_i^T R \mathbf{u}_i) dt + \mathbf{x}_N^T Q \mathbf{x}_N. \tag{3.7}$$

Unfortunately, the optimal $\mathbf{u}^*(t)$ is generally an open loop control and potentially not very robust. However, in practice it is sometimes possible to write that optimal control as a close loop control such as in the case of a linear quadratic regulator (LQR) where the cost functional is of the form of (3.4), (3.5), (3.6) or (3.7) and the system's dynamics are linear in both $\mathbf{x}$ and $\mathbf{u}$. In practice it is common to test and make adjustments to Q and R in a trial and error process until the controller functions as desired. This process is known as *parameter tuning*. This is usually necessary because it is very difficult to design a cost function that prioritizes the desired performance without trial and error.

3.2 PD control

In order to understand the new control method developed by the RAD lab that we will consider in this thesis, we will first review two commonly used controllers.

Perhaps the most widely used control algorithm in any application is the Proportional Derivative or PD controller. For a single dimensional system, the PD controller is defined as

$$u = k_p(x_{goal} - x) + k_d(\dot{x}_{goal} - \dot{x}),$$

where k_p and k_d are positive numbers called *gains*. These values are usually adjusted through trial and error (tuned) to perform well for the specific system being controlled. The first term is known as the proportional term because it applies a force proportional to the distance from the current state of the system to the desired state. If x represents physical displacement, this term is identical to the the force that would be applied if an Ideal Newtonian Spring with stiffness coefficient k_p and natural length zero was attached to x and x_{goal}. This term is responsible for pushing the system to the desired location. The second term is known as the derivative term because it is proportional to the difference in derivatives between our actual and desired system state. This can be thought of as a physical damper and reduces system overshoot and oscillations. In practice it is common to ignore $\dot{x}_{goal}$ because we usually wish our system to be stationary (have a zero derivative) at $\mathbf{x}_{goal}$. The control then becomes

$$u = k_p(x_{goal} - x) - k_d\dot{x}.$$

This technique can be applied to higher dimensional systems by simply applying a different PD controller to each individual element of the state, e.g. each joint of the robot. Our control is then given by

$$\mathbf{u} = K_p(\mathbf{x}_{goal} - \mathbf{x}) + K_d(\dot{\mathbf{x}}_{goal} - \dot{\mathbf{x}})$$

or

$$\mathbf{u} = K_p(\mathbf{x}_{goal} - \mathbf{x}) - K_d\dot{\mathbf{x}}$$

where K_p and K_d are now constant diagonal $n \times n$ matrices containing the gains for each individual element of $\mathbf{x}$. This control system is effective in a wide variety of situations and provably stable when applied to robotic systems [5]. Unfortunately, in order to achieve quick and precise movements in robotic systems the gains must typically be set to very large numbers. This is equivalent to mounting extremely stiff springs and dampers between the robot and the desired location of the robot, which gives the robot a propensity to move through any obstacles with great force and makes the robot inherently dangerous to anyone or anything in its vicinity. Moreover, this technique has also proven to be ineffective at controlling more complicated inflatable "soft" robots, generally leading to wild oscillations in the robot's movement [4]. For this reason, more sophisticated controls such as the control introduced in section 3.4 are needed.

3.3 Model Predictive Control

Model Predictive Control (MPC) or Receding Time Horizon Control is a more sophisticated and computationally expensive control technique when compared to PD control. In order to implement this control a time horizon T and a time step Δt must be chosen such that $T >> \Delta t$. The time horizon T will be used as a time by which we want to reach $\mathbf{x}_{goal}$ when calculating our control. The time step Δt will be used as the amount of time we wait between calculations of a new control. We will also require a finite time horizon cost functional that may be tuned in order to improve performance. The MPC algorithm has the following steps:

(i) Calculate the optimal control $\mathbf{u}^*(t)$ with finite time horizon T.

(ii) Apply the optimal control for Δt amount of time.

(iii) Return to step (i).

Despite calculating the optimal control $\mathbf{u}^*(t)$, MPC is suboptimal because the control problem changes with each cycle through the algorithm as the time at which our system is supposed to arrive at its goal keeps getting pushed backwards (the receding time horizon). These constant recalculations do, however, make this algorithm more robust to unexpected disturbances, such as errors in our physical model or collisions with unknown objects. This is because $\mathbf{u}$ is now calculated using $\mathbf{x}$ at each step and the controller has essentially become a closed loop controller. The principle drawback of MPC is that it is very computationally expensive to calculate and recalculate the optimal control after each time step Δt. For this reason, it is not commonly applied to robotic systems, however the method introduced in the following section modifies MPC in such a way that it is practical for robotic systems.

3.4 New Control Method

We now explore a new control method developed by the BYU RAD lab to make traditional robots less dangerous to use around humans and to effectively control soft robots for which traditional control methods fail to work effectively.

We first implement PD controllers on each individual joint. We intentionally choose very small gains that would typically lead to wild oscillations of the system. These small gains do, on the other hand, make the robot less dangerous. (Recalling that small gains can be seen as moving the joints with very weak springs and dampers should make these safety benefits obvious.) In order to counteract the tendency of the controller to oscillate we replace $\mathbf{x}_{goal}$ with $\mathbf{x}_{command}$. We then apply an MPC controller to choose $\mathbf{x}_{command}$ in order to move the system to $\mathbf{x}_{goal}$ as quickly as possible. In essence, we switch our control from τ to $\mathbf{x}_{command}$. If we are attempting to control a hard robot, we may rewrite the equations of motion as:

$$\dot{\mathbf{x}} = \begin{bmatrix} -D^{-1}(\mathbf{q})K_p & -D^{-1}(\mathbf{q})(C(\mathbf{q},\dot{\mathbf{q}}) + F_v + K_d) \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{x} + \begin{bmatrix} D^{-1}(\mathbf{q})K_p \\ \mathbf{0} \end{bmatrix} \mathbf{u}$$

$$\mathbf{u} = \mathbf{x}_{command} + D^{-1}\mathbf{g}(\mathbf{q}).$$

To make this control scheme more understandable, we again try to visualize this process through the use of springs and dampers. This new control system amounts to attaching very weak springs and dampers to the robot's joints. Then, instead of attaching the other end of the spring directly to the desired location, we intelligently pull on the other end of the spring to move the whole robot to $\mathbf{x}_{goal}$ as quickly as possible.

Unfortunately, this control method is every bit as computationally expensive as regular MPC control and far too complex to be applied to any useful robotic systems. In order to make this problem computationally tractable we make linear approximations of the system at each time step Δt and calculate the optimal control for these greatly simplified linear systems. Our method may now be described as:

(i) Create a linear approximation of the system around the current state of the system $\mathbf{x}(t)$.

(ii) Calculate the optimal control on the linearized approximate system with finite time horizon T.

(iii) Apply the optimal control for Δt amount of time.

(iv) Return to step (i).

Because the optimal control for a simple linear system is much easier to compute than the optimal control for a complicated system such as (2.2) (or for most industrial robots which are considerably more complicated) this control technique can be used to control robotic systems in real time. There is considerable freedom in this method to choose how we make the linear approximation to our system in step (i). The choice of which linearization technique we use significantly affects the practicality of this method and will be the topic of the next section.

Chapter 4. Introduction of Linearization Techniques

We now discuss several different methods to create the linear approximations of our system required by the algorithm in section 3.4. We introduce some additional notation which will aid us in discussing different linearization techniques. The linearization techniques we put forward are designed for systems of the form

$$\dot{\mathbf{x}} = A(\mathbf{x})\mathbf{x} + B(\mathbf{x})\mathbf{u}. \tag{4.1}$$

The equations of motion for a hard robot may be written this way by setting

$$A(\mathbf{x}) = \begin{bmatrix} -D^{-1}(\mathbf{q})K_p & -D^{-1}(\mathbf{q})(C(\mathbf{q},\dot{\mathbf{q}}) + F_v + K_d) \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

and

$$B(\mathbf{x}) = \begin{bmatrix} D^{-1}(\mathbf{q})K_p \\ \mathbf{0} \end{bmatrix}.$$

The equations of motion for a soft robot may be written this way by setting

$$A(\mathbf{x}) = \begin{bmatrix} -D^{-1}(\mathbf{q})S & -D^{-1}(\mathbf{q})(C(\mathbf{q},\dot{\mathbf{q}}) + F_v) & D^{-1}(\mathbf{q})\Gamma^{+} & D^{-1}(\mathbf{q})\Gamma^{-} \\ \mathbf{0} & I & \mathbf{0} & \mathbf{0} \\ -\boldsymbol{\beta}_{+}K_d & -\boldsymbol{\beta}_{+}K_p & \boldsymbol{\alpha}_{+} & \mathbf{0} \\ \boldsymbol{\beta}_{-}K_d & \boldsymbol{\beta}_{-}K_p & \mathbf{0} & \boldsymbol{\alpha}_{-} \end{bmatrix}$$

and

$$B(\mathbf{x}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \boldsymbol{\beta}_+ & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\beta}_- \end{bmatrix}.$$

We define $\mathbf{x}_{start}$ to be the state of our system at the beginning of step (i) of our control algorithm.

We now explore two linearizations techniques: the commonly used *Fixed State* technique and the new *Coupled Torque* method developed by the BYU RAD Lab.

4.1 Fixed State

Our first linearization technique comes from [6]. This technique simply assumes that the matrices A and B remain fixed as $\mathbf{x}$ changes. This gives us the equation

$$\dot{\mathbf{x}} = A(\mathbf{x}_{start})\mathbf{x} + B(\mathbf{x}_{start})\mathbf{u}. \tag{4.2}$$

We may then consider $A = A(\mathbf{x}_{start})$ and $B = B(\mathbf{x}_{start})$ to be constant matrices. This system has the advantage of possessing the easily computable analytical solution

$$\mathbf{x}(t) = e^{A(\mathbf{x}_{start})t}\mathbf{x}_{start} + (e^{A(\mathbf{x}_{start})t} - I)A^{-1}(\mathbf{x}_{start})B(\mathbf{x}_{start})\mathbf{u}. \tag{4.3}$$

We refer to this system as the Fixed State Approximation of (4.1) because we assume, for the sake of calculating A and B, that our state $\mathbf{x}$ remains fixed (at least for Δt amount of time). The optimal control for this linear system is far easier to compute than the optimal control for the full non-linear system and makes our control computationally practical in the sence that it can be implemented on certain robotic systems [4]. Unfortunately, even using this simplification, it is still too computationally expensive to implement this control on soft robots. For this reason, we introduce an even more simplified model.

4.2 Coupling Torque

In this method we not only assume that A and B are constant but that most of the terms in the vectors $A\mathbf{x}$ and $B\mathbf{u}$ are also constant.

To do this we first create the coupling matrices

$$C_A = \begin{bmatrix} I & I \\ I & I \end{bmatrix}, C_B = \begin{bmatrix} I \\ I \end{bmatrix}.$$

where I is the $n \times n$ identity matrix. Next we define

$$\tilde{C}_A = \mathbf{1}_{2n\times 2n} - C_A, \tilde{C}_B = \mathbf{1}_{2n\times n} - C_B$$

where $\mathbf{1}_{m\times n}$ is an $m \times n$ matrix of ones. That is $\tilde{C}_A$ has ones where C_A has zeros and visa versa. We will use $\circ$ to represent the Hadamard Product, which is pointwise multiplication of two matrices of the same dimensions. We now define

$$A_{diag} = C_A \circ A(\mathbf{x}_{start}), \quad B_{diag} = C_B \circ B(\mathbf{x}_{start}).$$

Note that these matrices contain the values of $A(\mathbf{x}_{start})$ and $B(\mathbf{x}_{start})$ on the main diagonal and the nth super and sub-diagonals and contains zeros elsewhere. We also define

$$A_{nondiag} = \tilde{C}_A \circ A(\mathbf{x}_{start}), \quad B_{nondiag} = \tilde{C}_B \circ B(\mathbf{x}_{start}).$$

These matrices are identical to $A(\mathbf{x}_{start})$ and $B(\mathbf{x}_{start})$ except with the elements of the main diagonal and the nth super and sub-diagonals replaced with zeros.

We then use the linearized system

$$\dot{\mathbf{x}} = A_{diag}\mathbf{x} + B_{diag}\mathbf{u} + A_{nondiag}\mathbf{x}_{start} + B_{nondiag}\mathbf{u}_{start}. \tag{4.4}$$

Note that the last two terms of (4.4) are constant. These terms represent the torque applied on one joint from the movement of other joints, which are the terms that couple the dynamics of each joint to the other joints. The Coupling Torque method assumes these coupling terms can be represented by as constant torque, or at least may be approximated by a constant torque over Δt amount of time. We note that there is also an easily calculable analytic solution to the linearized Coupled Torque system in (4.4) given by

$$\mathbf{x}(t) = e^{A_{diag}}\mathbf{x}_{start} + (e^{A_{diag}} - I)A_{diag}^{-1}(B_{diag}\mathbf{u} + A_{nondiag}\mathbf{x}_{start} + B_{nondiag}\mathbf{u}). \tag{4.5}$$

This solution will typically be even easier to solve for than (4.3), because it will likely be easier to take the inverse and matrix exponential of A_{diag} than the denser matrix $A(\mathbf{x}_{start})$.

The primary advantage of this method is that the resulting system consists of n decoupled equations (one for each joint) and the optimal control for each of these equations may therefore be calculated independently of the other joints. This opens the possibility of solving all of these control problems simultaneously on separate processors, increasing the speed of the algorithm and the complexity of the systems to which it may be applied. This method is currently the only method that has proven capable of controlling real soft robots in the RAD lab [4].

4.3 Proofs of Linear Approximations

Here we prove that both of these linear approximation become arbitrarily close to the nonlinear system they approximate when looking at a sufficiently small region about the point at which the approximations are taken. Note that $||\cdot||$ will refer to the Euclidean norm in $\mathbb{R}^n$.

Theorem 4.1. *Let $f : \mathbb{R}^{2n} \times \mathbb{R}^n \to \mathbb{R}^{2n}$ be a continuous function of the form*

$$f(\mathbf{x}, \mathbf{u}) = A(\mathbf{x})\mathbf{x} + B(\mathbf{x})\mathbf{u}.$$

Let $f_{FS} : \mathbb{R}^{2n} \times \mathbb{R}^n \to \mathbb{R}^{2n}$ be the Fixed State approximation given by (4.2) and $f_{CT} : \mathbb{R}^{2n} \times \mathbb{R}^n \to \mathbb{R}^{2n}$ be the Coupled Torque approximation given by (4.4) at some $\mathbf{x}_0 \in \mathbb{R}^{2n}$.

For every $\epsilon > 0$ there exists a $\delta_{FS} > 0$ such that $\mathbf{x} \in B(\mathbf{x}_0, \delta_{FS})$ implies

$$||f(\mathbf{x}, \mathbf{u}) - f_{FS}(\mathbf{x}, \mathbf{u})|| < \epsilon,$$

and there also exists $\delta_{CT} > 0$ such that $\mathbf{x} \in B(\mathbf{x}_0, \delta_{CT})$ implies

$$||f(\mathbf{x}, \mathbf{u}) - f_{CT}(\mathbf{x}, \mathbf{u})|| < \epsilon$$

for all $\mathbf{u} \in \mathbb{R}^n$.

Proof. Fix $\epsilon > 0$ and $\mathbf{u} \in \mathbb{R}^n$

Note

$$\begin{aligned} f_{FS}(\mathbf{x}_0, \mathbf{u}) &= A(\mathbf{x}_0)\mathbf{x}_0 + B(\mathbf{x}_0)\mathbf{u} \\ &= f(\mathbf{x}_0, \mathbf{u}) \end{aligned}$$

and

$$\begin{aligned} f_{CT}(\mathbf{x}_0, \mathbf{u}) &= A_{diag}\mathbf{x}_0 + B_{diag}\mathbf{u} + A_{nondiag}\mathbf{x}_0 + B_{nondiag}\mathbf{u} \\ &= A(\mathbf{x}_0)\mathbf{x}_0 + B(\mathbf{x}_0)\mathbf{u} \\ &= f(\mathbf{x}_0, \mathbf{u}) \end{aligned}$$

Recall f is continuous and f_{FS} and f_{CT} are linear and affine respectively and therefore continuous as well. Then there exists δ, δ_1 and δ_2 such that

$$||\mathbf{x} - \mathbf{x}_0|| < \delta \Rightarrow ||f(\mathbf{x}, \mathbf{u}) - f(\mathbf{x}_0, \mathbf{u})|| < \frac{\epsilon}{2}$$

and

$$||\mathbf{x} - \mathbf{x}_0|| < \delta_1 \Rightarrow ||f_{FS}(\mathbf{x}, \mathbf{u}) - f_{FS}(\mathbf{x}_0, \mathbf{u})|| < \frac{\epsilon}{2}$$

and

$$||\mathbf{x} - \mathbf{x}_0|| < \delta_2 \Rightarrow ||f_{CT}(\mathbf{x}, \mathbf{u}) - f_{CT}(\mathbf{x}_0, \mathbf{u})|| < \frac{\epsilon}{2}.$$

We now define $\delta_{FS} = \min\{\delta, \delta_1\}$ and $\delta_{CT} = \min\{\delta, \delta_2\}$. $||\mathbf{x} - \mathbf{x}_0|| < \delta_{FS}$ then implies

$$\begin{aligned}
||f(\mathbf{x}, \mathbf{u}) - f_{FS}(\mathbf{x}, \mathbf{u})|| &= ||f(\mathbf{x}, \mathbf{u}) - f(\mathbf{x}_0, \mathbf{u}) + f(\mathbf{x}_0, \mathbf{u}) - f_{FS}(\mathbf{x}, \mathbf{u})|| \\
&\leq ||f(\mathbf{x}, \mathbf{u}) - f(\mathbf{x}_0, \mathbf{u})|| + ||f(\mathbf{x}_0, \mathbf{u}) - f_{FS}(\mathbf{x}, \mathbf{u})|| \\
&= ||f(\mathbf{x}, \mathbf{u}) - f(\mathbf{x}_0, \mathbf{u})|| + ||f_{FS}(\mathbf{x}, \mathbf{u}) - f_{FS}(\mathbf{x}_0, \mathbf{u})|| \\
&< \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon,
\end{aligned}$$

and $||\mathbf{x} - \mathbf{x}_0|| < \delta_{CT}$ implies

$$\begin{aligned}
||f(\mathbf{x}, \mathbf{u}) - f_{CT}(\mathbf{x}, \mathbf{u})|| &= ||f(\mathbf{x}, \mathbf{u}) - f(\mathbf{x}_0, \mathbf{u}) + f(\mathbf{x}_0, \mathbf{u}) - f_{CT}(\mathbf{x}, \mathbf{u})|| \\
&\leq ||f(\mathbf{x}, \mathbf{u}) - f(\mathbf{x}_0, \mathbf{u})|| + ||f(\mathbf{x}_0, \mathbf{u}) - f_{CT}(\mathbf{x}, \mathbf{u})|| \\
&= ||f(\mathbf{x}, \mathbf{u}) - f(\mathbf{x}_0, \mathbf{u})|| + ||f_{CT}(\mathbf{x}, \mathbf{u}) - f_{CT}(\mathbf{x}_0, \mathbf{u})|| \\
&< \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon \qquad \square
\end{aligned}$$

Another way to state this result would be that as $\mathbf{x} \to \mathbf{x}_0$ our linear approximations of $f(\mathbf{x}, \mathbf{u})$ approaches the real system.

Chapter 5. Comparing Linearization Technique Performance

5.1 Introdution of Genetic Algorithms

In order to compare the performance of the linearization techniques put forward in the previous chapter, we must first understand how the optimal control from part (ii) of the algorithm put forward in section 3.4 is calculated in the "real world." The BYU RAD lab, where this control method was developed, has found the most effective algorithm for calculating the optimal control to be a genetic algorithm introduced in [7].

Genetic algorithms get their name from the fact that they were designed to simulate the evolution of genetic code in living organisms. In order to perform a genetic optimization algorithm on a certain problem, there must be a way to combine a certain number of potential solutions to create new and distinct potential solutions. This process is known as *mating* and is meant to mimic how living organisms mate to create distinct organisms that to some extent possess a mixture of the attributes of its parents. Genetic algorithms also require a process that makes random changes to a potential solution. This process is known as *mutation* and represents the random changes that occasionally occur in the DNA of living things.

All genetic algorithms follow the same basic steps:

(a) Start with a population of candidate solutions P.

(b) Calculate the performance of each element of P. (In the case of optimal control this means to find the value of the cost functional that would result from applying each control.)

(c) Select a subset of the elements in P and mate them together to get a new population C of the same cardinality as P.

(d) Apply mutations to some or all of the elements of C.

(e) Return to step (a) replacing P with C.

Each new population of candidate solutions P is called a generation. After a certain stopping requirement is met the algorithm is carried out until step (ii) is reached and the highest performing candidate is returned. Although a variety of stopping criterion could be used, in the case of using a genetic algorithm inside of the algorithm introduced in section 3.4, the natural choice for stopping criterion is to simply go through the loop a set number of times. This is because there is a set time, Δt, within which the calculation must be completed. Other stopping criterion run the risk of not finishing the calculation before a result is needed.

Although all genetic algorithms follow this same basic outline there still exists a considerable amount of freedom in how they are implemented, particularly in the way the subset of P from step (iii) is chosen and how its elements are grouped for mating. That subset is usually chosen to contain primarily well performing candidates to represent the principle of "survival of the fittest." There is also a substantial amount of freedom in how many mutations are applied in step (iv).

When applying genetic algorithms to our particular control problem we note that it is impossible to apply any control varying continuously in time on actual hardware. For this reason when actually calculating controls we divide the time horizon, T, into $S > 0$ equally sized subintervals and assume that our control is constant on each of these intervals. This is equivalent to making a step function approximation of a control that is continuous in time. With this assumption the space of possible controls becomes a subset of $\mathbb{R}^{S\times N}$ where N is the dimension of the control (equal to the number of joints in a hard robot and twice the number of joints in a soft robot). If we define $u_{i,j}$ to be the j^{th} component of the control vector in the i^{th} time interval of some potential control u, we can then define a mating algorithm. If $\{u^k\}_{k=1}^m$ is a set of controls, we can mate them to make a new control $\tilde{u}$ where $\tilde{u}_{i,j} = u^k_{i,j}$ for some randomly chosen k. We can apply mutations to a control u by adding some randomly generated number to $u_{i,j}$ for randomly chosen i and j. Now that we have described a mating and mutation algorithm we can compare the computational complexity for a genetic algorithm using both Fixed State and Coupled Torque linear approximations.

5.2 Comparing Computational Complexity with Genetic Algorithms

There are two concepts to keep in mind when calculating the computational complexity of a genetic algorithm. First, it is important to keep track of the number of operations that must be performed in order to carry out each step in order to estimate how long it will take to complete a single loop through the algorithm. Second, we must have some estimate on how quickly the algorithm will converge, or how many generations we must pass through before we arrive at an acceptable solution. Precise calculations of the convergence rates of genetic algorithms are often very difficult to derive and, although we touch on this briefly, we will spend the majority of this section examining the cost per loop of the algorithm.

We will first examine the cost of each step of the Genetic Algorithm presented in section 5.1. We will present the steps in increasing order of complexity.

Step (e) has no significant cost.

Step (a) only has significant complexity cost on the first pass through the loop when we must create our first population of candidates. Because we anticipate numerous passes through the loop we will therefore view its cost as negligible.

We will calculate the complexity of step (c) using the mating algorithm put forward in section 5.1. This process involves looking up a random number and replacing a value for every element of every candidate control at every time step and should then run in $\mathcal{O}(NS|P|)$ time for each iteration of the loop; where N is the dimension of the control vector, S is the number of time steps our horizon is broken into and $|P|$ is the number of candidate solutions considered in each generation. This step will most likely also involve sorting the elements of P by their performance (calculated in step (b)) in order to find the highest performing candidates to mate together. A good sorting algorithm should run in $\mathcal{O}(\ln |P|)$ time and is, therefore, negligible compared to the cost of the rest of this step.

We will use the mutation algorithm in section 5.1 to calculate the complexity of step (d). It is only necessary to look up three random numbers and replace one value for each mutation using this algorithm. This step will then run in $\mathcal{O}(M)$ time where M is the number of mutations applied. If the number of mutations applied is proportional to the number of values that could potentially change, then our cost becomes $\mathcal{O}(NS|P|)$.

Finally we will now show that step (b) has considerably larger complexity than the other steps, but that its complexity is significantly lower for Coupling Torque linearziations than Fixed State linearziations. This step requires that we calculate the trajectory of the system for every candidate control. This may be approximated in several ways. We will use Euler's method here because it is what current RAD Lab control code uses, but using a Runga-Kutta method or calculating the analytic solutions found in (4.3) and (4.5) should yield similar results. We now count the number of floating point operations (flops) necessary to perform step (b).

For Fixed State at every time step except the last time step and every control we must:

(i) Calculate this steps contribution to the cost function which has two parts:

- Calculate $\mathbf{u}_i^T R \mathbf{u}_i$ ($(2N-1)(N+1)$ flops)
- Calculate $\mathbf{x}_i^T Q \mathbf{x}_i$ ($(4N-1)(2N+1)$ flops)

(ii) Calculate the position at the next time step as follows:

- Calculate $\mathbf{x}_i + A(\mathbf{x}_0)\mathbf{x}_i + B(\mathbf{x}_0)\mathbf{u}_i$ ($(6N-1)2N$ flops)

In the last time step we only calculate the contribution to the cost function. Adding all of these costs together and multiplying by the number of time steps and size of a generation we arrive at the result:

Theorem 5.1. *The number of flops required to perform step (b) of a genetic algorithm for a control problem that has been linearized with the Fixed State method grows at a rate of $\mathcal{O}(N^2T|P|)$ with leading term behavior $22N^2T|P|$.*

For Coupled Torque we may look at every element of the control vector separately. We will define $\mathbf{x}_{i,j}$ to be the j^{th} and $j+n^{th}$ elements of $\mathbf{x}_i$. We also define $\widehat{M_j}$ to be the submatrix of a matrix M containing the elements found j^{th} or $j+n^{th}$ row and the j^{th} or $j+n^{th}$ column. Let $r_{j,j}$ be the j^{th} element of the main diagonal of R. Then at every time step except the last time step and every element of each control we must:

(i) Calculate this steps contribution to the cost function

- Calculate $\mathbf{u}_{i,j}^T r_{j,j} \mathbf{u}_{i,j}$ (2 flops)
- Calculate $\mathbf{x}_{i,j}^T \widehat{Q}_j \mathbf{x}_{i,j}$ (9 flops)

(ii) Calculate the position at the next time step

- Calculate $\mathbf{x}_{i,j} + \widehat{A_{diag}}_j \mathbf{x}_{i,j} + \widehat{B_{diag}}_j \mathbf{u_0} + \widehat{A_{nondiag}}_j \mathbf{x}_{0i,j} + \widehat{B_{nondiag}}_j \mathbf{u}_{0i,j}$ (16 flops)

Here we assume that the values of $\widehat{A_{nondiag}}_j \mathbf{x}_{0i,j} + \widehat{B_{nondiag}}_j \mathbf{u}_{0i,j}$ were already calculated when we constructed the linearization. As before we need only calculate the contribution to the cost function in the last time step. Adding the cost of each piece together and multiplying by the number of time steps, size of a generation and dimension of a control vector we arrive at the result:

Theorem 5.2. *The number of flops required to perform step (b) of a genetic algorithm for a control problem that has been linearized with the Coupled Torque method grows at a rate of $\mathcal{O}(NT|P|)$ with leading term behavior $27NT|P|$.*

Comparing theorems 5.1 and 5.2 shows a significant improvement in computational complexity when using Coupled Torque compared to Fixed State, but in a real world implementation the advantages of Coupled Torque would likely be even more pronounced. There are a couple reasons for this. First, we have not yet considered the possibility of solving the decoupled optimization problems coming from the coupled torque approximation in parallel. Using Coupled Torque, we could divide these calculation evenly over multiple processors up to N processors. This would lead to greatly improved performance:

Theorem 5.3. *The temporal complexity of step (b) of a genetic algorithm for a control problem that has been linearized with the Coupled Torque parallelized over $n \leq N$ processors method grows at a rate of $\mathcal{O}(NT|P|)$ with leading term behavior $27\left\lceil \frac{N}{n} \right\rceil T|P|$.*

It seems reasonable, however, to assume that one might design the hardware to solve this control problem specific to the robot being controlled. In which case we would ensure there were at least N processors. In this case our performance would be even better.

Corollary 5.4. *The temporal complexity of step (b) of a genetic algorithm for a control problem that has been linearized with the Coupled Torque parallelized over N processors method grows at a rate of $\mathcal{O}(T|P|)$ with leading term behavior $27T|P|$.*

Secondly, in our previous calculations we assumed that the number of candidate solutions in a generation for each decoupled optimization problem in the Coupled Torque problem would be the same as the cardinality of each generation for the single optimization problem that must be solved for the Fixed State approximation. The space of potential candidate solutions for these simplified problems would be much smaller and would most likely require a smaller generation size in order to solve the problem effectively. [8] presents a rigorously calculated bound for convergence of a genetic algorithm. This bound, however, requires that the set of all possible solutions to the problem, which we will call E, to be finite. This would necessarily be the case with a computer implementation of a genetic algorithm, but for our problem it may be expedient to limit the number of candidate solutions even more by picking a finite number of values, E_i, that the i^{th} component of the control vector can take. If we were to then look at our decoupled control problem for just one element of the control we would see the possible number of controls for this problem would be significantly smaller than the number of possible controls for the problem of calculating the entire control vector at once. [8] give the bound

$$||p^k - p^\infty|| \leq \{1 - [|E|^{2|P|} - (|E|^2 - |E|)^{|P|}](p_m p_s)^{2|P|}\}^{\lfloor \frac{k}{2} \rfloor} \tag{5.1}$$

where k represents the generation number and $||p^k - p^\infty||$ is the difference between the distribution of solutions given in generation k and the final distribution of solutions the algorithm converges to. The value p_m represents the minimum probability of changing from one potential solution to another potential solution via a mutation and p_s is another probability related to the system. From this inequality we see that if we have a smaller number of possible solutions we may then be able to get similar convergence while using a smaller population size. This suggests that we could get similar convergence rates for solving a Coupled Torque linearization as a Fixed State Linearization while using smaller generation sizes. This would improve the computational complexity of solving a Coupled Torque Problem even more when compared to a Fixed State problem.

Chapter 6. Proof of New Control Method

Several proofs of the global stability of the MPC control algorithm exist [9], and the modified control introduced in Section 3.4 has shown itself to be effective on a variety of robots [4], but the rigorous stability results for regular MPC do not apply to this new method. We will show that the usual control theory assumptions of Lipschitz continuous system dynamics and a compact set of possible controls are insufficient to show global stability for the control method introduced in 3.4. We will also prove in this chapter that this algorithm is stable inside of a small region of the goal $\mathbf{x}_{goal}$ when both the regular control theory assumptions and a few additional assumptions are made.

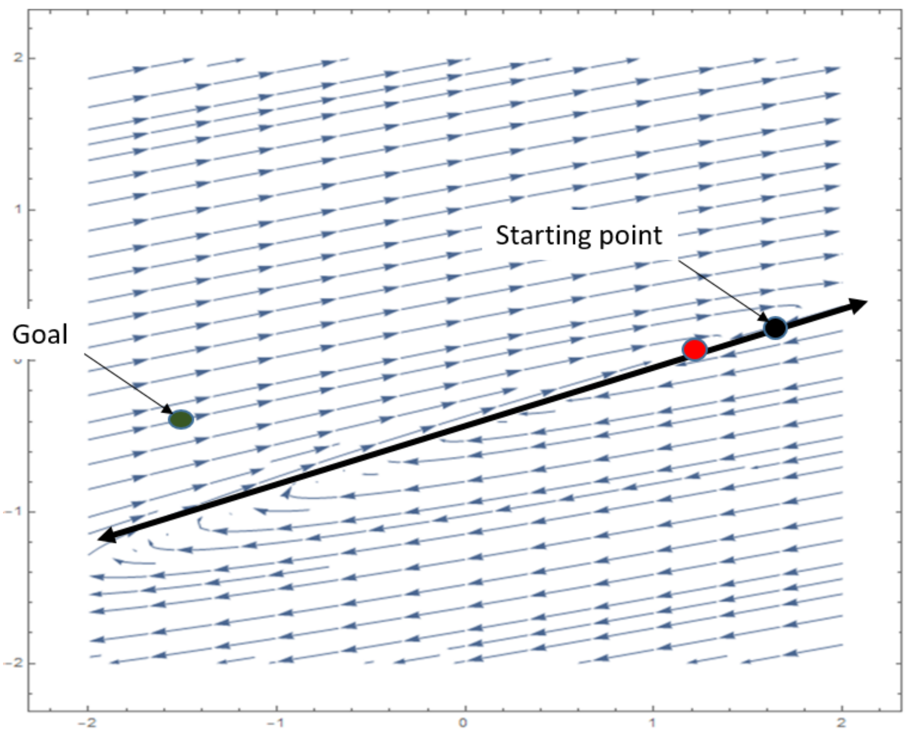


Figure 6.1: Phase plot of equation (6.1)

6.1 Global Instability

In this section we give a counter example to stability. Consider the system

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 3x - y \\ 9x - 4y \end{bmatrix} \mathbf{u}$$

This is of the form (4.1) with

$$A(x,y) = \mathbf{0},\ B(x,y) = \begin{bmatrix} 3x - y \\ 9x - 4y \end{bmatrix}.$$

Figure 6.1 shows a phase plot of

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 3x - y \\ 9x - 4y \end{bmatrix} \tag{6.1}$$

as well as a starting location and a goal location for our control problem and the linearized dynamics at the starting location. By choosing u to be either positive or negative, we may move either forward or backward along phase trajectories respectively. One can clearly see form the picture that choosing $u < 0$ will cause us to move backwards along a trajectory right to the goal, so the control problem does have a solution. However, using the linearized dynamics it will clearly be favorable to pick positive u to move closer to the goal. This, unfortunately, will drive us closer and closer to the fixed point marked in red, from which there will be no escape. In this example we see that the control method from Section 3.4 is not globally stable. That is, for certain systems this control method will utterly fail to move from certain starting conditions, $\mathbf{x}_0$, to a certain goal, $\mathbf{x}_{goal}$.

We now turn our attention to showing local stability, but before that we will prove several preliminary continuity results to help build intuition.

6.2 Continuity Results

We will define the set of *viable controls* as the controls that we may apply to a given system. In terms of robotics this is typically limited by the amount of torque that can be created by the actuator motors in hard robots, and in soft robots this is limited by the amount of pressure that can be produced by the air compressors. In continuation we will assume that the set of viable controls is compact and a subset of the Banach of space $L^\infty[t_0, t_{final}]$ functions. To simplify notation in continuation, unless otherwise specified L^∞ will refer to $L^\infty[t_0, t_{final}]$.

We show that the optimal control of a system is continuously dependent on the system that is being controlled.

Theorem 6.1. *Let* $\mathbf{V}$ *be a subset the space of bounded differentiable functions from* $\mathbb{R}^{2n} \times \mathbb{R}^m$ *to* $\mathbb{R}^{2n}$*,* $\mathbf{W}$ *be the compact subset of* L^∞ *functions from* $\mathbb{R}$ *to* $\mathbb{R}^{2n}$ *which are viable controls* *and* $\mathbf{Z}$ *be the set of cost functionals of the form 3.5. Define* $C : \mathbf{V} \times \mathbf{Z} \times \mathbb{R} \to \mathbf{W}$ *by*

$$C(f(x,u), J(x,u), \mathbf{x}_0) = \mathbf{u}^*$$

where $\mathbf{u}^*$ *is the optimal control for the problem*

$$\begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}, \mathbf{u}) \\ \mathbf{x}(t_0) &= \mathbf{x}_0 \\ J(\mathbf{x}, \mathbf{u}) &= \int_{t_0}^{t_{final}} (\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u}) dt + \mathbf{x}(t_{final})^T Q \mathbf{x}(t_{final}) \end{aligned}$$

Assume that C *is well defined, e.g. there exists a unique optimal control* $\mathbf{u}^*$ *for every combination of* $f \in \mathbf{V}$, $J \in \mathbf{Z}$ *and* $\mathbf{x}_0 \in \mathbb{R}^{2n}$. *Then the output of* C *in the* L^∞ *norm is continuously dependent on the input* $f(x,u)$ *in the* L^∞ *norm.*

In order to prove this, we will first need several lemmas.

Lemma 6.2. *Suppose* $\boldsymbol{\phi}(t)$ *is the unique solution to*

$$\begin{aligned} \dot{\mathbf{x}} &= f(t, \mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{x}(t_0) &= \mathbf{x}_0 \end{aligned}$$

Then $\boldsymbol{\phi}(t)$ *is continuously dependent on* $\mathbf{u}(t)$ *in the* L^∞ *norm.*

Proof. By the *Main Theorem for Ordinary Differential Equations in B-spaces* [10] $\boldsymbol{\phi}(t)$ is continuously dependent on both $\mathbf{x}_0$ and $\mathbf{u}(t)$ where a unique solution exists. □

In the next lemma we will use the function $J_{g,\mathbf{x}_0} : \mathbf{W} \to \mathbb{R}$ defined by

$$J_{g,\mathbf{x}_0}(\mathbf{u}) = \int_{t_0}^{t_{final}} (\boldsymbol{\phi}^T Q \boldsymbol{\phi} + \mathbf{u}^T R \mathbf{u}) dt + \boldsymbol{\phi}(t_{final})^T Q \boldsymbol{\phi}(t_{final}) \tag{6.2}$$

where $\boldsymbol{\phi}$ is the solution to

$$\dot{\mathbf{x}} = g(t, \mathbf{x}, \mathbf{u})$$
$$\mathbf{x}(t_0) = \mathbf{x}_0$$

Lemma 6.3. *The output of the function defined in* (6.2) *is continuously dependent on the input* $\mathbf{u}$ *in the* L^∞ *norm when* g *is differentiable.*

Proof. Fix $\epsilon > 0$. Because $\int_{t_0}^{t_{final}} \mathbf{u}^T R \mathbf{u} dt$ is the composition of a continuous functions and, therefore, continuous itself. Then there must exist $\delta_1 > 0$ such that $||\mathbf{u} - \tilde{\mathbf{u}}||_{L_\infty} < \delta_1$ implies

$$\left| \int_{t_0}^{t_{final}} \mathbf{u}^T R \mathbf{u} dt - \int_{t_0}^{t_{final}} \tilde{\mathbf{u}}^T R \tilde{\mathbf{u}} dt \right| < \frac{\epsilon}{2}$$

Similarly, there must exist $\delta_2 > 0$ such that $||\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}||_{L_\infty} < \delta_2$ implies

$$\left| \left(\int_{t_0}^{t_{final}} \boldsymbol{\phi}^T Q \boldsymbol{\phi} dt + \boldsymbol{\phi}(t_{final})^T Q \boldsymbol{\phi}(t_{final}) \right) - \left(\int_{t_0}^{t_{final}} \tilde{\boldsymbol{\phi}}^T Q \tilde{\boldsymbol{\phi}} dt + \tilde{\boldsymbol{\phi}}(t_{final})^T Q \tilde{\boldsymbol{\phi}}(t_{final}) \right) \right| < \frac{\epsilon}{2}$$

By lemma 6.2 there must exist $\delta_3 > 0$ such that $||\mathbf{u} - \tilde{\mathbf{u}}||_{L^\infty} < \delta_3$ implies $||\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}||_{L^\infty} < \delta_2$. Define $\delta = \min\{\delta_1, \delta_3\}$. Let $\tilde{\boldsymbol{\phi}}$ be the unique solution to

$$\dot{\tilde{\mathbf{x}}} = \tilde{f}(t, \tilde{\mathbf{x}}, \tilde{\mathbf{u}})$$
$$\tilde{\mathbf{x}}(t_0) = \mathbf{x}_0$$

Then $||\mathbf{u} - \tilde{\mathbf{u}}||_{L^\infty} < \delta$ will imply

$$\begin{aligned}
\left|J_{g,\mathbf{x}_0}(\mathbf{u}) - J_{g,\mathbf{x}_0}(\tilde{\mathbf{u}})\right| &= \Bigg|\int_{t_0}^{t_{final}} (\boldsymbol{\phi}^T Q\boldsymbol{\phi} + \mathbf{u}^T R\mathbf{u})dt + \boldsymbol{\phi}(t_{final})^T Q\boldsymbol{\phi}(t_{final}) \\
&\quad - \int_{t_0}^{t_{final}} (\tilde{\boldsymbol{\phi}}^T Q\tilde{\boldsymbol{\phi}} + \tilde{\boldsymbol{\phi}}^T R\tilde{\boldsymbol{\phi}})dt + \tilde{\boldsymbol{\phi}}(t_{final})^T Q\tilde{\boldsymbol{\phi}}(t_{final})\Bigg| \\
&\leq \left|\int_{t_0}^{t_{final}} \mathbf{u}^T R\mathbf{u}dt - \int_{t_0}^{t_{final}} \tilde{\mathbf{u}}^T R\tilde{\mathbf{u}}dt\right| \\
&\quad + \Bigg|\left(\int_{t_0}^{t_{final}} \boldsymbol{\phi}^T Q\boldsymbol{\phi}dt + \boldsymbol{\phi}(t_{final})^T Q\boldsymbol{\phi}(t_{final})\right) \\
&\quad - \left(\int_{t_0}^{t_{final}} \tilde{\boldsymbol{\phi}}^T Q\tilde{\boldsymbol{\phi}}dt + \tilde{\boldsymbol{\phi}}(t_{final})^T Q\tilde{\boldsymbol{\phi}}(t_{final})\right)\Bigg| \\
&< \frac{\epsilon}{2} + \frac{\epsilon}{2} \\
&= \epsilon
\end{aligned}$$

□

For the next and final lemma we will need one additional definition.

Definition 6.4. Let Ω be the set of possible values for $\boldsymbol{\phi}(t)$, the solution to

$$\dot{\mathbf{x}} = f(t, \mathbf{x})$$
$$\mathbf{x}(t_0) = \mathbf{x}_0$$

for $t \in [t_0, t_{final}]$.

Lemma 6.5. *Suppose $\boldsymbol{\phi}$ is the solution to*

$$\dot{\mathbf{x}} = f(t, \mathbf{x}, \mathbf{u})$$
$$\mathbf{x}(t_0) = \mathbf{x}_0$$

where f is differentiable. then $\boldsymbol{\phi}$ is continuously dependent on the function $f(t, \mathbf{x}, \mathbf{u})$ in the $L^\infty([t_0, t_{final}] \times \Omega \times \mathbf{W})$ norm.

Proof. Fix $\epsilon > 0$ and define $\tilde{\boldsymbol{\phi}}$ to be the solution to

$$\dot{\tilde{\mathbf{x}}} = \tilde{f}(t, \tilde{\mathbf{x}}, \mathbf{u})$$
$$\tilde{\mathbf{x}}(t_0) = \mathbf{x}_0.$$

Define

$$F(t, \mathbf{y}, \boldsymbol{\lambda}) = f(t, \mathbf{y}, \mathbf{u}) + \boldsymbol{\lambda}$$

where $\boldsymbol{\lambda}$ is a function in $L^\infty([t_0, t_{final}] \times \Omega \times \mathbf{W})$. Note that $\boldsymbol{\phi}$ solves

$$\dot{\mathbf{x}} = F(t, \mathbf{x}, 0)$$
$$\mathbf{x}(t_0) = \mathbf{x}_0.$$

and $\tilde{\boldsymbol{\phi}}$ solves

$$\dot{\tilde{\mathbf{x}}} = F(t, \tilde{f}(t, \tilde{\mathbf{x}}, \mathbf{u}) - f(t, \tilde{\mathbf{x}}, \mathbf{u}))$$
$$\tilde{\mathbf{x}}(t_0) = \mathbf{x}_0,$$

By the *Main Theorem for Ordinary Differential Equations in B-spaces* [10] we know that the solution to

$$\dot{\mathbf{y}} = F(t, \mathbf{y}, \boldsymbol{\lambda})$$
$$\mathbf{y}(t_0) = \mathbf{x}_0.$$

is continuously dependent on $\boldsymbol{\lambda}$. Then there exists $\delta > 0$ such that $||\tilde{f}(t, \mathbf{x}, \mathbf{u}) - f(t, \mathbf{x}, \mathbf{u})||_{L^\infty([t_0, t_{final}] \times \Omega \times \mathbf{W})} < \delta$ implies $||\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}||_{L^\infty} < \epsilon$ $\square$

We are now prepared to prove Theorem 6.1

Proof. Fix $\epsilon > 0$. Define the open ball $B(\mathbf{u}^*, \epsilon) = \{\mathbf{u} \in \mathbf{W} : ||\mathbf{u}^* - \mathbf{u}||_{L^\infty}\}$

Consider

$$\begin{aligned}\mathbf{K} :=& \mathbf{W} - B(\mathbf{u}^*, \epsilon)\\ =& \mathbf{W} \cap B(\mathbf{u}^*, \epsilon)^C\end{aligned}$$

$B(\mathbf{u}^*, \epsilon)^C$ is closed because $B(\mathbf{u}^*, \epsilon)$ is open. $\mathbf{K}$ is then the intersection of two closed sets and also closed. $\mathbf{K}$ is then a closed subset of the compact set $\mathbf{W}$ and compact itself. Then the continuous function (Lemma 6.3) $J_{f,\mathbf{x}_0}(\mathbf{u})$ achieves its minimum in $\mathbf{K}$ by the generalized extreme value theorem [11]. Define

$$\alpha := \min_{\mathbf{u} \in K} J_{f,\mathbf{x}_0}(\mathbf{u})$$

and let

$$\beta := \alpha - J_{f,\mathbf{x}_0}(\mathbf{u}^*).$$

$\beta > 0$ because $\mathbf{u}^*$ is the unique minimum of $J_{f,\mathbf{x}_0}(\mathbf{u})$ in $\mathbf{W}$. Further

$$J_f(\mathbf{u}) - J_f(\mathbf{u}^*) \geq \beta$$

for any $\mathbf{u} \in \mathbf{K}$.

Let

$$c(\boldsymbol{\phi}(t)) := \int_{t_0}^{t_{final}} (\boldsymbol{\phi}^T Q \boldsymbol{\phi}) dt + \boldsymbol{\phi}(t_{final})^T Q \boldsymbol{\phi}(t_{final}).$$

$c(\boldsymbol{\phi}(t))$ is continuous so there exists some $\delta_1 > 0$ such that $||\boldsymbol{\phi}(t) - \tilde{\boldsymbol{\phi}}(t)||_{L^\infty} < \delta_1$ implies $|c(\boldsymbol{\phi}(t)) - c(\tilde{\boldsymbol{\phi}}(t))| < \beta/2$.

Let $\boldsymbol{\phi}$ be the solution to

$$\dot{\mathbf{x}} = f(t, \mathbf{x})$$
$$\mathbf{x}(t_0) = \mathbf{x}_0$$

and $\tilde{\boldsymbol{\phi}}$ be the solution to

$$\dot{\tilde{\mathbf{x}}} = \tilde{f}(t, \tilde{\mathbf{x}})$$
$$\tilde{\mathbf{x}}(t_0) = \mathbf{x}_0$$

By Lemma 6.5 there must exist $\delta > 0$ such that $||\tilde{f} - f||_{L^\infty} < \delta$ we have that $||\boldsymbol{\phi}(t) - \tilde{\boldsymbol{\phi}}(t)||_{L^\infty} < \delta_1$. then $||\tilde{f} - f||_{L^\infty} < \delta$ implies that for any $\mathbf{u} \in \mathbf{W}$

$$\begin{aligned}
||J_f(\mathbf{u}) - J_{\tilde{f}}(\mathbf{u})|| =& \bigg| \int_{t_0}^{t_{final}} (\boldsymbol{\phi}^T Q \boldsymbol{\phi} + \mathbf{u}^T R \mathbf{u}) dt + \boldsymbol{\phi}(t_{final})^T Q \boldsymbol{\phi}(t_{final}) \\
& - \int_{t_0}^{t_{final}} (\tilde{\boldsymbol{\phi}}^T Q \tilde{\boldsymbol{\phi}} + \mathbf{u}^T R \mathbf{u}) dt + \tilde{\boldsymbol{\phi}}(t_{final})^T Q \tilde{\boldsymbol{\phi}}(t_{final}) \bigg| \\
=& \bigg| \int_{t_0}^{t_{final}} \boldsymbol{\phi}^T Q \boldsymbol{\phi} dt + \boldsymbol{\phi}(t_{final})^T Q \boldsymbol{\phi}(t_{final}) \\
& - \int_{t_0}^{t_{final}} \tilde{\boldsymbol{\phi}}^T Q \tilde{\boldsymbol{\phi}} dt + \tilde{\boldsymbol{\phi}}(t_{final})^T Q \tilde{\boldsymbol{\phi}}(t_{final}) \bigg| \\
=& |c(\boldsymbol{\phi}(t)) - c(\tilde{\boldsymbol{\phi}}(t))| \\
<& \frac{\beta}{2}
\end{aligned}$$

which in turn implies

$$J_{\tilde{f}}(\mathbf{u}) - J_f(\mathbf{u}) > -\frac{\beta}{2}$$

and

$$J_f(\mathbf{u}) - J_{\tilde{f}}(\mathbf{u}) > -\frac{\beta}{2}.$$

Then for any $\tilde{f}$ such that $||\tilde{f} - f||_{L_\infty} < \delta_2$ and any $\mathbf{u} \in \mathbf{K}$

$$\begin{aligned}
J_{\tilde{f}}(\mathbf{u}) - J_{\tilde{f}}(\mathbf{u}^*) &= J_{\tilde{f}}(\mathbf{u}) + [-J_f(\mathbf{u}) + J_f(\mathbf{u})] + [-J_f(\mathbf{u}^*) + J_f(\mathbf{u}^*)] - J_{\tilde{f}}(\mathbf{u}^*) \\
&= [J_{\tilde{f}}(\mathbf{u}) - J_f(\mathbf{u})] + [J_f(\mathbf{u}) - J_f(\mathbf{u}^*)] + [J_f(\mathbf{u}^*) - J_{\tilde{f}}(\mathbf{u}^*)] \\
&\geq J_{\tilde{f}}(\mathbf{u}^*) - J_f(\mathbf{u}^*) + \beta + J_f(\mathbf{u}) - J_{\tilde{f}}(\mathbf{u}) \\
&> -\frac{\beta}{2} + \beta - \frac{\beta}{2} \\
&> 0
\end{aligned}$$

This implies

$$J_{\tilde{f}}(\mathbf{u}) > J_{\tilde{f}}(\mathbf{u}^*)$$

and $\mathbf{u}$ cannot be the optimal control for the system

$$\begin{aligned}
\dot{\mathbf{x}} &= \tilde{f}(t, \mathbf{x}, \mathbf{u}) \\
\mathbf{x}(t_0) &= \mathbf{x}_0.
\end{aligned}$$

Then the optimal control must lie within $B(\mathbf{u}^*, \epsilon)$ and C depends continuously on f $\square$

With this result we are prepared to show that we may force the system using the optimal control from the linearized dynamics to follow the system using the optimal control for the full non-linear dynamics arbitrarily closely by requiring that our initial condition $\mathbf{x}_0$ be close to our goal $\mathbf{x}_{goal}$.

Theorem 6.6. *Consider the optimal control for the problem*

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \tag{6.3}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0 \tag{6.4}$$

$$J(\mathbf{x}, \mathbf{u}) = \int_{t_0}^{t_{final}} (\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u}) dt + \mathbf{x}(t_{final})^T Q \mathbf{x}(t_{final}) \tag{6.5}$$

where $f \in \mathbf{V}, J \in \mathbf{Z}$ *and* $\mathbf{u} \in \mathbf{W}$ *as defined in theorem 6.1.*

Let $\tilde{f}(\mathbf{x}, \mathbf{u})$ *be a linearization (either Coupled Torque of Fixed State) of* $f(\mathbf{x}, \mathbf{u})$*. let* $\mathbf{u}^*$*be the optimal control for the problem defined by* (6.3), (6.4) *and* (6.5) *and* $\tilde{\mathbf{u}}^*$ *be the optimal control for the problem defined by*

$$\dot{\mathbf{x}} = \tilde{f}(\mathbf{x}, \mathbf{u}),$$

(6.4) *and* (6.5)*. Define* $\boldsymbol{\phi}^*$ *and* $\tilde{\boldsymbol{\phi}}^*$ *to be the solutions to*

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}^*)$$
$$\mathbf{x}(t_0) = \mathbf{x}_0.$$

and

$$\dot{\mathbf{x}} = f(\mathbf{x}, \tilde{\mathbf{u}}^*)$$
$$\mathbf{x}(t_0) = \mathbf{x}_0.$$

respectively. Let $\Omega^* = \{\boldsymbol{\phi}^*(t) | t \in [t_0, t_{final}]\}$

For every $\epsilon > 0$ *there exists* $\delta > 0$ *such that* $||\boldsymbol{\phi}^* - \tilde{\boldsymbol{\phi}}^*||_{L_\infty} < \epsilon$ *whenever* $\mathbf{x}_0 \in B(\mathbf{x}_{goal}, \delta)$ *and* $||\boldsymbol{\phi}^*(t) - \mathbf{x}_{goal}||$ *is non-increasing in* t*.*

Note that the requirement that $||\boldsymbol{\phi}^*(t) - \mathbf{x}_{goal}||$ is non-increasing in t would have to be true in some small region of $\mathbf{x}_{goal}$ in order for us to be able to effectively control our system.

Proof. Fix $\epsilon > 0$. By lemma 6.2 there must exist $\delta_1 > 0$ such that $||\mathbf{u}^* - \tilde{\mathbf{u}}^*||_{L^\infty} < \delta_1$ implies $||\boldsymbol{\phi}^* - \tilde{\boldsymbol{\phi}}^*||_{L_\infty} < \epsilon$. By theorem 6.1 there must exist $\delta_2 > 0$ such that $||f - \tilde{f}||_{(\Omega^* \times \mathbf{W})} < \delta_2$ implies $||\mathbf{u}^* - \tilde{\mathbf{u}}^*||_{L^\infty} < \delta_1$. By theorem 4.1 there exists $\delta_3 > 0$ such that $\mathbf{x} \in B(x_0, \delta_3)$ implies $||f(\mathbf{x}, \mathbf{u}) - \tilde{f}(\mathbf{x}, \mathbf{u})|| < \delta_2$. We then choose $\delta < \frac{\delta_3}{2}$. Then $\mathbf{x}_0 \in B(\mathbf{x}_{goal}, \delta)$ implies that $B(\mathbf{x}_{goal}, \delta) \subset B(x_0, \delta_3)$. Because $||\boldsymbol{\phi}^*(t) - \mathbf{x}_{goal}||$ is non-increasing $\Omega^* \subset B(\mathbf{x}_{goal}, \delta)$. Then $||f - \tilde{f}||_{(\Omega^* \times \mathbf{W})} < \delta_2$ and $||\boldsymbol{\phi}^* - \tilde{\boldsymbol{\phi}}^*||_{L_\infty} < \epsilon$. $\square$

This thereom tells us that, for all bounded differentiable $f(\mathbf{x}, \mathbf{u})$ for which the optimal control is unique, starting close enough to $\mathbf{x}_{goal}$ will force our control using the linearized system to follow the actual optimal control arbitrarily closely. Because the true optimal control is usually stable this suggests that the control method introduced in section 3.4 may also be stable. Regrettably, this is insufficient for a proof of local stability. This proof also requires a concept of a certain rate of convergence to the optimal control at the starting location $\mathbf{x}_0$ approaches $\mathbf{x}_{goal}$. In the next section we define this concept precisely.

6.3 Local Stability

Definition 6.7. Let $\mathbf{Q}$ be the subset of bounded differentiable functions such that there exists $\gamma > 0$ and $\lambda > 0$ such that $||\boldsymbol{\phi}^*(t) - \mathbf{x}_{goal}|| < ||\mathbf{x}_0 - \mathbf{x}_{goal}||e^{-\lambda t}$ when $\mathbf{x}_0 \in B(\mathbf{x}_{goal}, \gamma)$. Further, there exists $\delta > 0$ such that for all $\delta > \tilde{\delta} > 0$ there exists a value $\tilde{\epsilon} > 0$ such that $\mathbf{x}_0 \in B(\mathbf{x}_{goal}, \tilde{\delta})$ implies $||\boldsymbol{\phi}^*(t) - \tilde{\boldsymbol{\phi}}^*(t)|| < \tilde{\epsilon}(t - t_0)$ and $\tilde{\epsilon} < \lambda\tilde{\delta}$ and $\tilde{\epsilon} < \tilde{\delta}$ where $\boldsymbol{\phi}$ and $\tilde{\boldsymbol{\phi}}^*$ are defined as in Theorem 6.6.

We note that $\mathbf{Q}$ should be non-empty because there exists systems for which the Fixed State and Coupled Torque aproximations are exactly identical to the original system, and one of these systems would lie in $\mathbf{Q}$ if it was possible to find an asymptotically stable control for that systems.

Theorem 6.8. *Suppose $f(\mathbf{x}, \mathbf{u}) \in \mathbf{Q}$. There exists $\alpha > 0$ such that for all $\alpha > \tilde{\alpha} > 0$ there exists Δt such that $||\mathbf{x}_0 - \mathbf{x}_{goal}|| = \tilde{\alpha}$ implies $\tilde{\mathbf{x}}^*(\Delta t) \in B(\mathbf{x}_{goal}, \tilde{\alpha})$.*

Proof. Let $\alpha = \min\{\gamma, \delta\}$. If $\alpha > \tilde{\alpha} > 0$ and $||\mathbf{x}_0 - \mathbf{x}_{goal}|| = \tilde{\alpha}$. This yields

$$\begin{aligned}
||\tilde{\mathbf{x}}^*(\Delta t) - \mathbf{x}_{goal}|| &= ||\tilde{\mathbf{x}}^*(\Delta t) - \mathbf{x}^*(\Delta t) + \mathbf{x}^*(\Delta t) - \mathbf{x}_{goal}|| \\
&\leq ||\tilde{\mathbf{x}}^*(\Delta t) - \mathbf{x}^*(\Delta t)|| + ||\mathbf{x}^*(\Delta t) - \mathbf{x}_{goal}|| \\
&\leq \tilde{\epsilon}\Delta t + ||\mathbf{x}_0 - \mathbf{x}_{goal}||e^{-\lambda\Delta t}
\end{aligned}$$

Note

$$||\tilde{\mathbf{x}}^*(0) - \mathbf{x}_{goal}|| = 0.$$

Taking the derivative with respect to Δt yields

$$\frac{d}{d\Delta t}(\tilde{\epsilon}\Delta t + ||\mathbf{x}_0 - \mathbf{x}_{goal}||e^{-\lambda\Delta t}) = \tilde{\epsilon} - \lambda||\mathbf{x}_0 - \mathbf{x}_{goal}||e^{-\lambda\Delta t}$$

Now if we set $\Delta t = 0$ we find

$$\frac{d}{d\Delta t}(\tilde{\epsilon}\Delta t + ||\mathbf{x}_0 - \mathbf{x}_{goal}||e^{-\lambda\Delta t})\bigg|_{\Delta t=0} = \tilde{\epsilon} - \lambda||\mathbf{x}_0 - \mathbf{x}_{goal}||$$

Because

$$\tilde{\epsilon} < \lambda\tilde{\delta} \leq \lambda||\mathbf{x}_0 - \mathbf{x}_{goal}||$$

we have

$$\frac{d}{d\Delta t}(\tilde{\epsilon}\Delta t + ||\mathbf{x}_0 - \mathbf{x}_{goal}||e^{-\lambda\Delta t})\bigg|_{\Delta t=0} < 0.$$

This means there must exist some sufficiently small value of Δt such that

$$||\tilde{\mathbf{x}}^*(\Delta t) - \mathbf{x}_{goal}|| < ||\tilde{\mathbf{x}}^*(0) - \mathbf{x}_{goal}|| = ||\mathbf{x}_0 - \mathbf{x}_{goal}||$$

This means that $\tilde{\mathbf{x}}^*(\Delta t) \in B(\mathbf{x}_{goal}, \tilde{\alpha})$. □

Because all of the hypothesis of theorem 6.8 are still met if we set $\mathbf{x}_0 = \tilde{\mathbf{x}}^*(\Delta t)$, this theorem can be applied again in the next step of our control. This is significant because it means that if we choose proper Δt we can make a trapping region around $\mathbf{x}_{goal}$ arbitrarily small when using the algorithm presented in section 3.4.

Corollary 6.9. *When using the control algorithm form section 3.4 on a system in* $\mathbf{Q}$*, for any* $\epsilon > 0$ *there exists* Δt *such that* $B(\mathbf{x}_{goal}, \epsilon)$ *is a trapping region.*

This condition is slightly stronger than stability but not as strong as asymptotic stability.

Chapter 7. Conclusion

The new control algorithm developed by the BYU Rad lab has shown itself in practice to be uniquely capable of controlling complicated robotic systems and especially those found in so-called soft robots in both simulations and in the real world [4], but has also shown itself to be particularly difficult to analyze mathematically. We have shown that, when using the Coupled Torque technique for linear approximations, this technique can be computationally feasible for use in even the most complicated robots. Of particular note is the fact that as long as computational hardware is designed specifically for the robot to be controlled we can virtually eliminate the growth in computational complexity as the number of joints, and therefore the dimension of the system, increases.

Via counter-example, we found that global stability for this control method using the standard control theory assumptions of Lipschitz continuous dynamics and a compact space of controls was insufficient to prove global stability (asymptotic or not) of this method. We did, however, show the existence of arbitrarily small trapping regions around the desired location for a specific subset of control problems. This result although slightly stronger than local stability is not as strong as local asymptotic stability.

7.1 Future Work

It is no easy task to look at the dynamics of a specific system and determine whether it is part of the subset of differentiable systems (called $\mathbf{Q}$ in this paper) that where we have proved local stability as it is currently formulated. To make these results more applicable to the real world it would be useful to have sufficient conditions for a system to reside in $\mathbf{Q}$. It would also be a clear improvement to the results of this thesis to find conditions which guarantee the new control method is locally asymptotically stable or guarantee global stability.

Bibliography

[1] IFR Press Releases. *Robots double worldwide by 2020*, 2018. `https://ifr.org/ifr-press-releases/news/robots-double-worldwide-by-2020`.

[2] Occupational Safety and Health Administration. *Accident Search Results*, 2018. `https://www.osha.gov/pls/imis/accidentsearch.search?sic=&sicgroup=&naics=&acc_description=&acc_abstract=&acc_keyword=%22Robot%22&inspnr=&fatal=&officetype=&office=&startmonth=&startday=&startyear=&endmonth=&endday=&endyear=&keyword_list=on&p_start=20&p_finish=39&p_sort=&p_desc=DESC&p_direction=Prev&p_show=40`.

[3] Koji Ikuta Hideki Ishii and Makoto Nokata. Safety evaluation method of design and control for human-care robots. *The International Journal of Robotics Research*, 22(5):281–295, May 2003.

[4] Marc Killpack. Effect of simplified robot dynamic models on model predictive control performance. 2016.

[5] Mark W. Spong, Seth Hutchinson, and M. Vidyasager. *Robot Modeling and Control*. John Wiley and Sons, Inc., 2006.

[6] Katsushiko Ogata. *Discrete-Time Control Systems*. Pearson Education, 2 edition, 1995.

[7] Phil Hyatt and Marc D. Killpack. Real-time evolutionary model predictive control using a graphics processing unit. pages 15–17, November 2017.

[8] Liang Ming, Yuping Wang, and Yiu-Ming cheung. On convergence rate of a class of genetic algorithms. July 2006.

[9] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predicitive control: Stability and optimality. *Automatica*, (36):789–814, November 2000.

[10] Eberhard Zeidler. *Nonlinear Functional Analysis and its Applications*. Springer-Verlag New York Inc., 1986.

[11] Jefferey Humpherys, Tyler Jamison Jarvis, and Emily J. Evans. *Foundations of Applied Mathematics*, volume 1. 2017.