APPLICATION OF GENETIC PROGRAMMING TO UNDERACTUATED MECHANICAL SYSTEMS

by

WILLIAM J. YOUNG

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

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

2002

Oakland University
Rochester, Michigan

APPROVED BY:

_____
Chair

_____
Adviser

_____
Adviser

To my parents:
For your love, support, and encouragement.

To my wife Carolyn:
For your love, patience, and valuable advice. I couldn't have done this without you.

ACKNOWLEDGMENTS

I wish to thank my Thesis committee for allowing me to pursue a topic of my choosing and for their excellent instruction. Special thanks goes to Dr. Ka C. Cheok for his patience, advice, and humor.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

1.1 Genetic Programming Background

Genetic Programming (GP) is an optimization method based on evolutionary concepts. It belongs to a family of algorithms, which include Genetic Algorithms [1] and Evolutionary Strategies. GP and related strategies are typically applied to problems for which optimal solutions are unknown or unobtainable in reasonable time (e.g. NP-hard, NP-complete problems) [13].

Each of these evolutionary methods involves evaluating a "population" of potential solutions, carrying the best ones over to a new "generation" of solutions. The copy process intentionally introduces random changes into some of the solutions, which may improve or worsen their performance. After many generations, the solutions often, but not always, approach an optimal solution.

Although evolutionary methods can solve difficult problems, they are not a panacea and do not displace analytic methods. In particular, GP does not guarantee any satisfactory solution will be found. The solutions GP does find may not be simple or straightforward or even humanly understandable [13]. GP solutions tend not to be *optimal,* but "good enough". Since an optimal solution is unknown in many problem domains explored with evolutionary methods, non-optimality is often not an issue. Despite these limitations, evolutionary methods are proving themselves useful in a growing number of areas.

1.2 Motivations

Few publications exist which directly address the creation of controllers for mechanical systems via Genetic Programming.

In [2], GP is employed to develop a controller for a two-link planar manipulator with revolute joints. The control task is to move a payload from Point A to Point B and back again in minimal time. Controllers were successfully evolved to accomplish this task for manipulators of varying physical parameters.

In [3], GP is used to find a control algorithm for satellite attitude control. The control problem considered is to change a space satellite's attitude, starting from rest and ending at rest. An inertia wheel actuates each axis of the satellite. By building the problem's symmetry into the Genetic Programming formulation, the authors were able to discover a result that performs nearly as well as known numerical solutions.

In [4], the authors applied Genetic Programming to the problem of controlling a 3-dimensional multibody system. The system considered was Luxo, the articulated lamp in the introduction to the movie "Toy Story". The control problem was to cause Luxo to hop from a starting point to an end point in minimal time, without falling over. The authors achieved good results, well suited to their animation objectives, i.e. the control appeared "natural" and a bit imperfect, which matched Luxo's desired character.

The plant in each of these works is a fully actuated mechanical system. (Actually, Luxo is underactuated once the base begins to leave the ground. The base is fairly heavy and large, which makes landing reasonably stable. A large amount of the control occurs in the fully actuated mode, with the base solidly on the ground.) These

works together provide solid evidence that controllers evolved via Genetic Programming are capable of controlling fully actuated systems.

It is not clear whether the successes for fully actuated systems extend to underactuated systems, which are fundamentally more difficult to control. Underactuated systems are successfully evolved in [5], where the objective is to evolve both the bodies and control systems for imaginary creatures to swim through a medium. Although the swimming problem is inherently underactuated, the author employs a variety of techniques to tailor the genetic algorithm very specifically to the research objective, and so does not compare directly against the Genetic Programming approach.

1.3 Project Goals

The goal of this work is to explore whether Genetic Programming can be successfully applied to the control of an underactuated mechanical system. The following questions will be investigated:

- Which analytical control methods are successful for controlling underactuated mechanical systems?
- Can GP be used to develop controllers for underactuated mechanical systems?
- If solutions exist, how do they perform with respect to the known analytical methods?
- If solutions exist, what are they useful for?

CHAPTER 2. THE DISCBOT

The system under consideration is a variant of the two-dimensional inverted pendulum [6], called the DiscBot. In the DiscBot, the cart normally seen in the inverted pendulum is replaced with a disc, as shown in Figure 2.1.

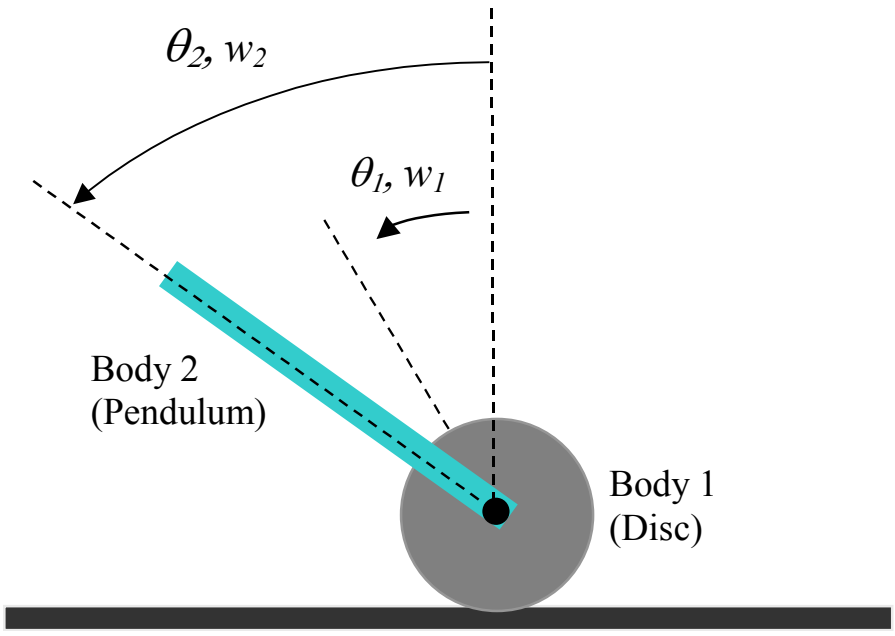

Figure 2.1 The DiscBot

Table 2.1 summarizes DiscBot parameters used in simulations described in this thesis.

Table 2.1 DiscBot Parameters

Parameter	Value	Units	Note
M_1	1.00	kg	Disc mass
R	0.20	m	Disc radius
J_1	0.02	kg*m^2	Disc rotational inertia
M_2	2.00	kg	Pendulum mass
L	1.00	m	Pendulum length
W	0.10	m	Pendulum width
J_2	0.168	kg*m^2	Pendulum rotational inertia
G	9.81	m/s^2	Gravitational acceleration

2.1 Rigid Body Dynamics

The Disc is assumed to be in slip-free contact with the ground at all times. The joint between the Disc and the Pendulum is frictionless, and the system is in Earth's gravitational field.

Using the Euler-Lagrange method, the system dynamics are given by [7]

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial L}{\partial \mathbf{q}} = \mathbf{u}\,. \tag{2.1}$$

where

$L = K - P$

K = total system kinetic energy

P = total system potential energy

$\mathbf{q} = \begin{bmatrix}\theta_1 & \theta_2\end{bmatrix}^T$ = joint displacement vector

$\dot{\mathbf{q}} = \begin{bmatrix}\omega_1 & \omega_2\end{bmatrix}^T$ = joint velocity vector

$\mathbf{u}$ = general torque vector applied at joints.

Substitution of K and P yields

$$\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{\mathbf{q}}}-\frac{\partial P}{\partial \dot{\mathbf{q}}}\right)-\frac{\partial K}{\partial \mathbf{q}}+\frac{\partial P}{\partial \mathbf{q}}=\mathbf{u}\,. \tag{2.2}$$

Since P is a function of joint position only, (2.2) becomes

$$\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{\mathbf{q}}}\right)-\frac{\partial K}{\partial \mathbf{q}}+\frac{\partial P}{\partial \mathbf{q}}=\mathbf{u}\,. \tag{2.3}$$

For many mechanical systems, (2.3) can be represented by the compact notation

$$\begin{aligned}&\frac{d}{dt}\left(\frac{\partial}{\partial \dot{\mathbf{q}}}\left(\frac{1}{2}\dot{\mathbf{q}}^{t}\mathbf{W}\dot{\mathbf{q}}\right)\right)-\frac{\partial K}{\partial \mathbf{q}}+\frac{\partial P}{\partial \mathbf{q}}=\mathbf{u}\\ &\text{or}\\ &\mathbf{W}\ddot{\mathbf{q}}+\dot{\mathbf{W}}\dot{\mathbf{q}}-\frac{\partial K}{\partial \mathbf{q}}+\frac{\partial P}{\partial \mathbf{q}}=\mathbf{u}\\ &\mathbf{W}\ddot{\mathbf{q}}+\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}+\boldsymbol{\tau}_{g}(\mathbf{q})=\mathbf{u}\\ &\text{where}\\ &\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}=\dot{\mathbf{W}}\dot{\mathbf{q}}-\frac{\partial K}{\partial \mathbf{q}}\quad\text{and}\quad\boldsymbol{\tau}_{g}(\mathbf{q})=\frac{\partial P}{\partial \mathbf{q}}\ .\end{aligned} \tag{2.4}$$

$\boldsymbol{W}$ is a symmetric positive definite matrix known as the *Inertia Matrix*. For each link, $\boldsymbol{C}$ contains Coriolis and centripetal forces, and $\boldsymbol{\tau_g}$ contains gravitational forces.

2.1.1 DiscBot Dynamics

For the DiscBot, the potential energy is given by

$$P = m_1 g r + m_2 g\left(r + L\cos q_2\right). \tag{2.5}$$

Differentiating (2.5) with respect to $\mathbf{q}$ yields $\boldsymbol{\tau_g}$,

$$\boldsymbol{\tau}_g = \frac{\partial P}{\partial \mathbf{q}} = \begin{bmatrix} 0 \\ -m_2 g L \sin q_2 \end{bmatrix}. \tag{2.6}$$

The kinetic energy K is given by

$$\begin{aligned} & K = K_1 + K_2 \\ & K_1 = \frac{1}{2}\left(m_1 r^2 + J_1\right)\dot{q}_1^2 \\ & K_2 = \frac{1}{2}\left(m_2 r^2 \dot{q}_1^2 + \left(m_2 L^2 + J_2\right)\dot{q}_2^2 + 2 m_2 r L \cos(q_2)\dot{q}_1 \dot{q}_2\right) \\ & \text{or} \\ & K = \frac{1}{2}\dot{\mathbf{q}}^t \mathbf{W} \dot{\mathbf{q}} \\ & \mathbf{W} = \begin{bmatrix} \left((m_1 + m_2) r^2 + J_1\right) & m_2 r L \cos q_2 \\ m_2 r L \cos q_2 & m_2 L^2 + J_2 \end{bmatrix}. \end{aligned} \tag{2.7}$$

From (2.4), **C** can now be expressed as

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = \dot{\mathbf{W}}\dot{\mathbf{q}} - \frac{\partial K}{\partial \mathbf{q}} = \begin{bmatrix} -m_2 r L \sin q_2 \dot{q}_2^2 \\ 0 \end{bmatrix}.$$

The applied torque vector **u** is composed of equal and opposite torque components applied to the Disc and the Pendulum. This accounts for the actuator connecting them. The vector **u** is given by

$$\mathbf{u} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \tau = \mathbf{M}\,\tau .$$

where τ is the joint torque. Note that choosing q_2 defined with respect to q_1 rather than the vertical axis, as in Figure 2.1, would lead to the more familiar $\boldsymbol{M} = [0\ 1]^T$. As will become apparent in later sections, the DiscBot is much easier to visualize with the chosen absolute body coordinates.

Putting it all together, the DiscBot dynamics are

$$\begin{bmatrix} \left((m_1+m_2)r^2+J_1\right) & m_2 rL\cos q_2 \\ m_2 rL\cos q_2 & m_2L^2+J_2 \end{bmatrix}\ddot{\mathbf{q}} + \begin{bmatrix} -m_2 rL\sin(q_2)\dot{q}_2^2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -m_2 gL\sin q_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}\tau \ . \tag{2.8}$$

2.2 Control System Formulation

The work described here compares some traditional nonlinear control methods to controllers derived via Genetic Programming. For the comparison to be meaningful, all considered controllers must fit into the same architecture.

2.2.1 Architecture

A system described by (2.4) can be expressed in *standard form*, which is more useful for control design [8]. A system in standard form is written

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \sum_{i=1}^{r} g_i(x)\tau_i \ . \tag{2.9}$$

The DiscBot dynamics (2.4) can be written in standard form as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{G}(\mathbf{x})\boldsymbol{\tau}$$
$$\mathbf{y} = \mathbf{h}(\mathbf{x})$$
where
$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \dot{\mathbf{q}} \\ -\mathbf{W}^{-1}\left(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \boldsymbol{\tau}_g(\mathbf{q})\right) \end{bmatrix}, \quad \mathbf{G}(\mathbf{x}) = \begin{bmatrix} \mathbf{0}_{2x2} \\ \mathbf{W}^{-1}\mathbf{M} \end{bmatrix} \tag{2.10}$$
$$\mathbf{y} = \mathbf{h}(\mathbf{x})$$
and
$$\mathbf{x} = \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix}.$$

The variable **x** is the system state vector of dimension *n*, **u** is the input vector of dimension *m*, and **y** is the output vector of dimension *p*. A stabilizing controller, which seeks to bring ***x*** to zero, can be written as

$$\boldsymbol{\tau} = \mathbf{p}(\mathbf{x}). \tag{2.11}$$

Figure 2.2 graphically depicts the system formed by (2.10) and (2.11).

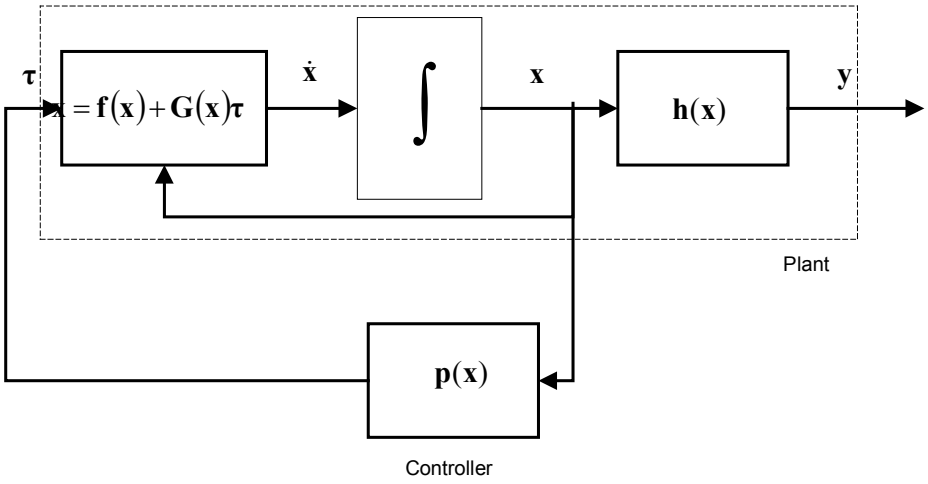

Figure 2.2 Controller Architecture

CHAPTER 3. CONVENTIONAL CONTROL

3.1 Linear Approximation with LQR Control

Local stabilization can often be achieved by constructing a linear controller based on a linearized approximation of the plant dynamics [8]. Such linear controllers are appropriate in applications not requiring global stabilization.

For the many mechanical systems described by (2.4), repeated in (3.1), a linear approximation about the origin can be calculated from the first two terms of the Taylor Series representing **f**(**x**) and **G**(**x**), taken about the origin [8]. Doing so produces an approximation of the nonlinear system in standard linear form, as in (3.2).

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{G}(\mathbf{x})\boldsymbol{\tau}$$

where

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \dot{\mathbf{q}} \\ -\mathbf{W}^{-1}\left(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \boldsymbol{\tau}_g(\mathbf{q})\right) \end{bmatrix}, \quad \mathbf{G}(\mathbf{x}) = \begin{bmatrix} \mathbf{0} \\ \mathbf{W}^{-1}\mathbf{M} \end{bmatrix} \tag{3.1}$$

and

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

$$\dot{\mathbf{x}} \approx \mathbf{A}\mathbf{x} + \mathbf{B}\boldsymbol{\tau}$$

where

$$\mathbf{A} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial \mathbf{q}} & \frac{\partial \mathbf{f}}{\partial \dot{\mathbf{q}}} \end{bmatrix}, \text{ and } \mathbf{B} = \begin{bmatrix} \frac{\partial \mathbf{G}}{\partial \boldsymbol{\tau}} \end{bmatrix} \tag{3.2}$$

Symbolic representations of $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{G}}{\partial \mathbf{u}}$ tend to be quite lengthy, even for relatively low order systems. It is very helpful to use symbolic manipulation software when computing the **A** and **B** matrices, or $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{G}}{\partial \mathbf{u}}$ may be computed numerically.

Using the linear approximation, the entire array of linear control design tools is available to create stabilizing controllers. One choice is the LQR (Linear Quadratic Regulator), an optimal linear regulator based on a quadratic performance index [9].

Given the linear time varying system

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\boldsymbol{\tau}(t) \tag{3.3}$$

and the quadratic performance index

$$J = \int_{t_0}^{\infty} \frac{1}{2}\left[\mathbf{x}^T(t)\mathbf{Q}\mathbf{x}(t) + \boldsymbol{\tau}^T(t)\mathbf{R}\boldsymbol{\tau}(t)\right]dt\,, \tag{3.4}$$

where **Q** is a real, symmetric positive semi-definite matrix and **R** is a real, symmetric positive definite matrix, there exists a linear, time-invariant state feedback controller of the form [9]:

$$\begin{aligned} \boldsymbol{\tau}(\mathbf{x}) &= -\mathbf{R}^{-1}\mathbf{B}^t\mathbf{K}\mathbf{x}(t) \\ \boldsymbol{\tau}(\mathbf{x}) &= -\mathbf{F}\mathbf{x}(t) \end{aligned} \tag{3.5}$$

where **K** is the solution to the algebraic Riccati equation

$$\mathbf{0} = -\mathbf{K}\mathbf{A} - \mathbf{A}^T\mathbf{K} - \mathbf{Q} + \mathbf{K}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{K}\,. \tag{3.6}$$

3.2 Input-Output Linearization

Feedback linearization strategies seek to “undo” the nonlinear elements contained in the plant, yielding a system that, viewed externally, appears linear. The

Input-Output Linearization technique accomplishes this by using output feedback. As depicted in Figure 3.1, the system's state is fed into a *linearization block* and new input, **v**, is introduced. The system, when viewed from the outside, appears to be a linear system with **v** as input and **y** as output.

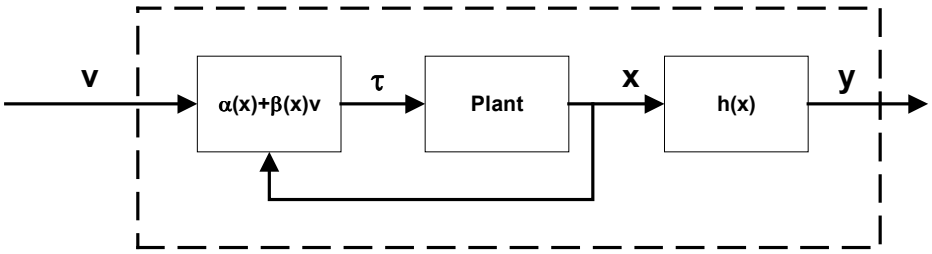

Figure 3.1 Input-Output Linearization

The goal of linearization is to discover an *algebraic* relationship between system input and output. Doing so allows the calculation of the system input required to achieve a desired output. This arrangement effectively removes nonlinearities between system input and output.

The system output function **h**(**x**) is used to find the direct input-output relationship. However, a system's output does not usually explicitly include input terms. Since **h**(**x**) is a function of the system state and the system state derivative is influenced directly by the system input, it makes sense to differentiate the output function **h**(**x**) until the input appears. For each output, the number of differentiations

required to recover the input is called its *relative degree.* For MIMO systems, the *total relative degree* is defined as the sum of all output channels' relative degrees [10].

3.3 Relative Degree and Invertibility

The relative degree for each output channel can be found by differentiating each output channel until the input appears. In general, each channel must be evaluated individually, but if one assumes the same relative degree for each output channel (equal-RD assumption), the input-output relationship can be determined with a compact matrix notation. The equal-RD assumption is valid for many mechanical systems with force inputs and displacement outputs.

Recall the standard form system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{G}(\mathbf{x})\boldsymbol{\tau}$$
$$\mathbf{y} = \mathbf{h}(\mathbf{x})$$

where

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \dot{\mathbf{q}} \\ -\mathbf{W}^{-1}\left(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \boldsymbol{\tau}_g(\mathbf{q})\right) \end{bmatrix}, \quad \mathbf{G}(\mathbf{x}) = \begin{bmatrix} \mathbf{0} \\ \mathbf{W}^{-1}\mathbf{M} \end{bmatrix} \tag{3.7}$$

and

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

where $\mathbf{h}(\mathbf{x})$ is assumed to be a function of displacement states only,

$$\mathbf{y} = \mathbf{h}(\mathbf{q}) .$$

The first derivative is

$$\dot{\mathbf{y}} = \frac{\partial \mathbf{h}(\mathbf{q})}{\partial \mathbf{x}}\dot{\mathbf{x}} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x}) + \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\mathbf{G}(\mathbf{x})\boldsymbol{\tau} = L_f\mathbf{h}(\mathbf{x}) + (L_G\mathbf{h}(\mathbf{q}))\boldsymbol{\tau} . \tag{3.8}$$

(Note the use of Lie derivatives. Lie algebra proves very useful in feedback linearization theory. For more detail, refer to [10]. The theoretical utility of Lie algebra will not be relied upon heavily here, however.)

To determine if the input appears, only the second term in (3.8) must be calculated:

$$L_G h(\mathbf{q}) = \frac{\partial h(\mathbf{q})}{\partial \mathbf{x}} \mathbf{G}(\mathbf{x}) = \begin{bmatrix} \mathbf{J} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{W}^{-1}\mathbf{M} \end{bmatrix} = \mathbf{0}.$$

Since the input has not appeared, the second derivative must be found,

$$\ddot{\mathbf{y}} = \frac{\partial \dot{\mathbf{y}}}{\partial \mathbf{x}} \dot{\mathbf{x}} = L_f^2 \mathbf{h}(\mathbf{x}) + (L_G L_f \mathbf{h}(\mathbf{x}))\boldsymbol{\tau}. \tag{3.9}$$

Again, it is only necessary to calculate the second term to determine the appearance of the input.

$$L_f \mathbf{h}(\mathbf{q}) = \frac{\partial \mathbf{h}(\mathbf{q})}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}) = \begin{bmatrix} \mathbf{J} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}} \\ -\mathbf{W}^{-1}\left(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \tau_{\mathrm{g}}\right) \end{bmatrix} = \mathbf{J}\dot{\mathbf{q}}$$

$$L_G L_f \mathbf{h}(\mathbf{q}) = \frac{\partial \mathbf{J}\dot{\mathbf{q}}}{\partial \mathbf{x}} \mathbf{G}(\mathbf{x}) = \begin{bmatrix} \frac{\partial \mathbf{J}\dot{\mathbf{q}}}{\partial \mathbf{q}} & \frac{\partial \mathbf{J}\dot{\mathbf{q}}}{\partial \dot{\mathbf{q}}} \end{bmatrix} \mathbf{G}(\mathbf{x}) = \begin{bmatrix} \frac{\partial \mathbf{J}\dot{\mathbf{q}}}{\partial \mathbf{q}} & \mathbf{J} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{W}^{-1}\mathbf{M} \end{bmatrix} = \mathbf{J}\mathbf{W}^{-1}\mathbf{M}$$

The quantity $\mathbf{J}\mathbf{W}^{-1}\mathbf{M}$ is zero only when $\mathbf{J}$ is singular, since $\mathbf{W}$ is always positive definite. Therefore, the input has appeared after two differentiations, and the relative degree for each output channel is two. The total relative degree is the sum of the individual relative degrees, four.

The relation (3.9) connects the second derivative of the output to the input. If this relationship can be inverted to solve for $\boldsymbol{\tau}$ as a function of $\ddot{\mathbf{y}}$, then the system can be at least partially linearized. Upon inspection, it is apparent that the system must be

square (have same number of inputs and outputs) in order to invert (3.9). Otherwise, an additional constraint on either input or output must be imposed in order to invert the system. Techniques exist for inverting non-square systems with more inputs than outputs [11], but are not considered here.

3.3.1 Linearization

To design a feedback linearization for the system, we must first define a new system input

$$\mathbf{v} = \begin{bmatrix} y_1^{(r)} \\ \vdots \\ y_p^{(r)} \end{bmatrix} = \mathbf{y}^{(r)}$$

where r is the relative degree of each output channel.

By calculating the derivatives of $\mathbf{y}$, one can see that

$$\begin{aligned} &\mathbf{v} = \mathbf{y}^{(2)} = \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}\ddot{\mathbf{q}} \\ &\Downarrow \\ &\ddot{\mathbf{q}} = \mathbf{J}^{-1}\left(\ddot{\mathbf{y}} - \dot{\mathbf{J}}\dot{\mathbf{q}}\right). \end{aligned} \tag{3.10}$$

When (3.10) is substituted into the system dynamics (3.1), we find the algebraic relationship between $\mathbf{u}$ and $\mathbf{v}$:

$$\begin{aligned} &\mathbf{W}\ddot{\mathbf{q}} + \mathbf{C} + \boldsymbol{\tau}_g = \mathbf{M}\boldsymbol{\tau} \\ &\ddot{\mathbf{q}} = -\mathbf{W}^{-1}\left(\mathbf{C} + \boldsymbol{\tau}_g\right) + \mathbf{W}^{-1}\mathbf{M}\boldsymbol{\tau} \\ &\mathbf{J}\ddot{\mathbf{q}} = -\mathbf{J}\mathbf{W}^{-1}\left(\mathbf{C} + \boldsymbol{\tau}_g\right) + \mathbf{J}\mathbf{W}^{-1}\mathbf{M}\boldsymbol{\tau} \\ &\dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}\ddot{\mathbf{q}} = \dot{\mathbf{J}}\dot{\mathbf{q}} - \mathbf{J}\mathbf{W}^{-1}\left(\mathbf{C} + \boldsymbol{\tau}_g\right) + \mathbf{J}\mathbf{W}^{-1}\mathbf{M}\boldsymbol{\tau} \\ &\mathbf{v} = \dot{\mathbf{J}}\dot{\mathbf{q}} - \mathbf{J}\mathbf{W}^{-1}\left(\mathbf{C} + \boldsymbol{\tau}_g\right) + \mathbf{J}\mathbf{W}^{-1}\mathbf{M}\boldsymbol{\tau}\ . \end{aligned} \tag{3.11}$$

If the system is square and $\mathbf{JW^{-1}M}$ is invertible, then (3.11) can be solved for $\boldsymbol{\tau}$:

$$\begin{aligned}
&\boldsymbol{\tau} = \boldsymbol{\alpha}(\mathbf{x}) + \boldsymbol{\beta}(\mathbf{x})\mathbf{v} \\
&\text{where} \\
&\boldsymbol{\alpha}(\mathbf{x}) = \left(\mathbf{JW^{-1}M}\right)^{-1}\left(\mathbf{JW^{-1}}\left(\mathbf{C} + \mathbf{T_g}\right) - \dot{\mathbf{J}}\dot{\mathbf{q}}\right) \\
&\boldsymbol{\beta}(\mathbf{x}) = \left(\mathbf{JW^{-1}M}\right)^{-1} .
\end{aligned} \tag{3.12}$$

This expression is the feedback linearization for the system. The overall linearized system can be described by a linear state-space equation, with state variables determined by the output **y**. Specifically, for each $v_i = y_i^{(r)}$, there are *r* states consisting of y_i and its first *(r-1)* derivatives, as in (3.13).

$$\begin{aligned}
z_1 &= y_1 & \dot{z}_1 &= z_{p+1} \\
&\vdots & &\vdots \\
z_p &= y_p & \dot{z}_p &= z_{2p} \\
z_{p+1} &= \dot{y}_1 \quad \Rightarrow & \dot{z}_{p+1} &= v_1 \\
&\vdots & &\vdots \\
z_{2p} &= \dot{y}_p & \dot{z}_{2p} &= v_p
\end{aligned} \tag{3.13}$$

The new states can be expressed as the linear system

$$\begin{aligned}
&\dot{\mathbf{z}} = \mathbf{Az} + \mathbf{Bv} \\
&\mathbf{A} = \begin{bmatrix} \mathbf{0_{p\times p}} & \mathbf{I_{p\times p}} \\ \mathbf{0_{p\times p}} & \mathbf{0_{p\times p}} \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} \mathbf{0_{p\times p}} \\ \mathbf{I_{p\times p}} \end{bmatrix}.
\end{aligned} \tag{3.14}$$

3.3.2 Internal Dynamics

The Input-Output Linearization procedure is only well developed for square plants (number of inputs equals number of outputs) [10]. For plants with more outputs than inputs, the control designer must choose which outputs may be controlled, as the

excess of outputs over inputs cannot be linearized. The states related to the nonlinearized outputs will become the *internal dynamics*. If the internal dynamics are unstable, the entire system is unstable, even though the linearized states may be controllable.

3.4 Passivity Methods

Controllers may also be designed by consideration of the *passivity* property found in many systems. A Passivity-Based Controller (PBC) can stabilize a system without a linearization procedure, and exploit the system's inherent structure.

A system is considered passive if it contains no internal sources of energy. That is, the system's total energy is the sum of its initial energy and the energy supplied via the inputs. Euler-Lagrange systems can be shown to be passive by differentiating the total system energy to reveal the power flow.

For Euler-Lagrange systems, creating a controller utilizing the passivity property is relatively straightforward. The following closely matches the derivation in [12].

In an Euler-Lagrange system the total energy is

$$E = \tfrac{1}{2}\dot{\mathbf{q}}^{\mathbf{T}}\mathbf{W}\dot{\mathbf{q}} + P(\mathbf{q})$$

The power flowing into the system is therefore

$$\dot{E} = \dot{\mathbf{q}}^{\mathbf{T}}\mathbf{W}\ddot{\mathbf{q}} + \tfrac{1}{2}\dot{\mathbf{q}}^{\mathbf{T}}\dot{\mathbf{W}}\dot{\mathbf{q}} + \dot{\mathbf{q}}^{T}\frac{\partial \mathbf{P}}{\partial \mathbf{q}}. \tag{3.15}$$

Recalling that $\frac{\partial P}{\partial \mathbf{q}} = \boldsymbol{\tau}_g$ and substituting the system dynamics into (3.15), the power becomes

$$\begin{aligned}
\dot{E} &= \dot{\mathbf{q}}^{\mathbf{T}}\mathbf{W}\left(-\mathbf{W}^{-1}\mathbf{C}\dot{\mathbf{q}} - \mathbf{W}^{-1}\boldsymbol{\tau}_g + \mathbf{W}^{-1}\mathbf{M}\boldsymbol{\tau}\right) + \tfrac{1}{2}\dot{\mathbf{q}}^{\mathbf{T}}\dot{\mathbf{W}}\dot{\mathbf{q}} + \dot{\mathbf{q}}^{\mathbf{T}}\boldsymbol{\tau}_g \\
\dot{E} &= -\dot{\mathbf{q}}^{\mathbf{T}}\mathbf{C}\dot{\mathbf{q}} - \dot{\mathbf{q}}^{\mathbf{T}}\boldsymbol{\tau}_g + \dot{\mathbf{q}}^{\mathbf{T}}\mathbf{M}\boldsymbol{\tau} + \tfrac{1}{2}\dot{\mathbf{q}}^{\mathbf{T}}\dot{\mathbf{W}}\dot{\mathbf{q}} + \dot{\mathbf{q}}^{\mathbf{T}}\boldsymbol{\tau}_g \qquad (3.16) \\
\dot{E} &= \dot{\mathbf{q}}^{\mathbf{T}}\left(\tfrac{1}{2}\dot{\mathbf{W}} - \mathbf{C}\right)\dot{\mathbf{q}} + \dot{\mathbf{q}}^{\mathbf{T}}\mathbf{M}\boldsymbol{\tau} \ .
\end{aligned}$$

Now, using the fact that $\left(\dot{\mathbf{W}} - \mathbf{2C}\right)$ is skew-symmetric (for proof see [17]), and for skew-symmetric matrices

$$\mathbf{x}^{\mathbf{T}}\mathbf{A}\mathbf{x} = 0 \quad \forall \mathbf{x} \ ,$$

(3.16) becomes

$$\dot{E} = \dot{\mathbf{q}}^{\mathbf{T}}\mathbf{M}\boldsymbol{\tau} = \dot{\tilde{\mathbf{q}}}^{\mathbf{T}}\boldsymbol{\tau}$$

where

$$\dot{\tilde{\mathbf{q}}} = \mathbf{M}^{\mathbf{T}}\dot{\mathbf{q}} \ .$$

The power flowing into the system is therefore the sum of the products of each joint's velocity relative to its attached links and the torque acting on that joint.

A candidate Lyapunov function for an Euler-Lagrange system is

$$V(q,\dot{q}) = \tfrac{1}{2}K_E\left(E - E_d\right)^2 + \tfrac{1}{2}K_v\left|\dot{\tilde{\mathbf{q}}}\right|^2 + \tfrac{1}{2}K_p\left|\tilde{\mathbf{q}} - \tilde{\mathbf{q}}_d\right|^2 .$$

Its time derivative is

$$\begin{aligned}
\dot{V}(q,\dot{q}) &= K_E\left(E - E_d\right)\dot{E} + \dot{\tilde{\mathbf{q}}}^{\mathbf{T}}K_v\ddot{\tilde{\mathbf{q}}} + \dot{\tilde{\mathbf{q}}}^{\mathbf{T}}K_p\left(\tilde{\mathbf{q}} - \tilde{\mathbf{q}}_{\mathbf{d}}\right) \\
\dot{V}(q,\dot{q}) &= \dot{\tilde{\mathbf{q}}}^{\mathbf{T}}\left[K_E\left(E - E_d\right)\boldsymbol{\tau} + K_v\ddot{\tilde{\mathbf{q}}} + K_p\left(\tilde{\mathbf{q}} - \tilde{\mathbf{q}}_{\mathbf{d}}\right)\right] .
\end{aligned}$$

Substituting the system dynamics and regrouping yields

$$\dot{V}(q,\dot{q}) = \dot{\tilde{\mathbf{q}}}^{\mathbf{T}}\left[K_E\left(E-E_d\right)\boldsymbol{\tau} + K_v\left[-\mathbf{M}^T\mathbf{W}^{-1}\left(\mathbf{C}+\boldsymbol{\tau}_g\right)+\mathbf{M}^T\mathbf{W}^{-1}\mathbf{M}\boldsymbol{\tau}\right]+K_p\left(\tilde{\mathbf{q}}-\right.\right.$$
$$\dot{V}(q,\dot{q}) = \dot{\tilde{\mathbf{q}}}^{\mathbf{T}}\left[\left[K_E\left(E-E_d\right)+K_v\mathbf{M}^T\mathbf{W}^{-1}\mathbf{M}\right]\boldsymbol{\tau} - K_v\mathbf{M}^T\mathbf{W}^{-1}\left(\mathbf{C}+\boldsymbol{\tau}_g\right)+K_p\left(\tilde{\mathbf{q}}-\right.\right.$$
$$\dot{V}(q,\dot{q}) = \dot{\tilde{\mathbf{q}}}^{\mathbf{T}}\left[\left[K_E\left(E-E_d\right)+K_v\mathbf{M}^T\mathbf{W}^{-1}\mathbf{M}\right]\boldsymbol{\tau}+\mathbf{F}\right] \tag{3.17}$$

where

$$\mathbf{F} = -K_v\mathbf{M}^T\mathbf{W}^{-1}\left(\mathbf{C}+\boldsymbol{\tau}_g\right)+K_p\left(\tilde{\mathbf{q}}-\tilde{\mathbf{q}}_{\mathbf{d}}\right).$$

If (3.17) is defined to be non-increasing by the assertion

$$\dot{V}(\mathbf{q},\dot{\mathbf{q}}) = -K_\partial\dot{\tilde{\mathbf{q}}}^{\mathbf{T}}\dot{\tilde{\mathbf{q}}}$$

then the control law becomes

$$\begin{aligned}\left[K_E\left(E-E_d\right)+K_v\mathbf{M}^T\mathbf{W}^{-1}\mathbf{M}\right]\boldsymbol{\tau}+\mathbf{F} &= -K_\partial\dot{\tilde{\mathbf{q}}}\\ \boldsymbol{\tau} &= \left[K_E\left(E-E_d\right)+K_v\mathbf{M}^T\mathbf{W}^{-1}\mathbf{M}\right]^{-1}\left(-K_\partial\dot{\tilde{\mathbf{q}}}-F\right)\end{aligned} \tag{3.18}$$

so long as

$$K_E\left(E-E_d\right)+K_v\mathbf{M}^T\mathbf{W}^{-1}\mathbf{M} \tag{3.19}$$

is nonsingular. To find the system's stability properties, one must use LaSalle's theorem, which must be applied on a case-by-case basis.

The control law (3.18) is stable, but typically converges to a homoclinic orbit, which includes the desired fixed point. The control must therefore be switched to a suitable linear controller after the system enters its basin of attraction.

CHAPTER 4. GENETIC PROGRAMMING METHOD

4.1 Overview

Genetic Programming was inspired by the better-known Genetic Algorithm [1]. Both techniques maintain large populations of potential solutions, each evaluated in turn, using a *fitness function*. Some individuals will inevitably perform better than others with respect to the fitness function. A new generation of solutions is generated from the current one based on its fitness distribution. Selection methods vary, but more fit individuals will always be more likely to be selected than their less fit neighbors. Copying into the new generation is performed *with errors* to introduce variation into the next generation. The new generation is then evaluated like the first and the process repeats. As the process progresses, solutions tend to satisfy the fitness function better and better. A "run" is considered complete when either the fitness function is satisfied or a predetermined maximum number of generations have elapsed.

4.1.1 Tree Representation of Computer Programs

Genetic Programming's unique representation of solutions distinguishes it from Genetic Algorithms. In Genetic Algorithms, solutions are typically represented as bit strings, or genotypes, with predefined regions of the bit string mapped to specific phenotypic traits. In Genetic Programming, solutions are represented directly as *computer programs*, written in some suitable domain-specific language [13]. The

programs are represented as n-ary trees, similar to an intermediate representation used by many computer language compilers. An example tree is shown in Figure 4.1.

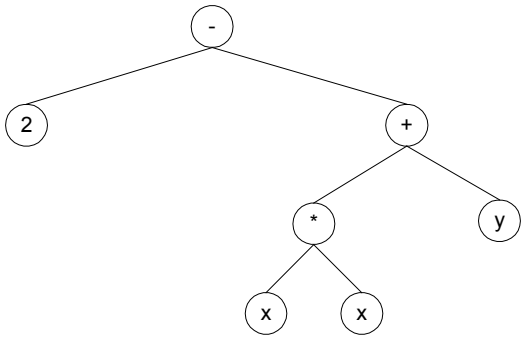

Figure 4.1 Sample Tree

A tree, when traversed in prefix order, represents an expression or program. For example, the tree in Figure 4.1 is equivalent to the mathematical expression

$$2-\left(x^2+y\right).$$

The tree representation offers two advantages:

- The solutions' structures are not predetermined. The trees are free to evolve into whatever program is required to solve the problem.
- Problem domains include any area a solution can be represented as a computer program. Examples include mathematical equations, logical relations, etc.

To represent solutions in a particular domain, two sets of nodes must be defined: the *function set* and the *terminal set*. The function and terminal sets together form the language in which problem domain solutions are expressed.

The function set consists of operators that are necessary or may be helpful in the desired solution, and are always internal nodes. Examples include addition, subtraction, and multiplication. Each function's output must be defined for *all* combinations of its inputs, because trees are constructed and manipulated in random fashion, and no guarantee can be provided as to the range of a given function's inputs. Therefore, modified versions of some familiar functions must be used in a GP system. For example, the division operator cannot be used directly in a GP system, as its result is undefined when the divisor is zero. However, a modified version of the division operator, referred to as *protected division*, may be used. The protected division operator behaves exactly as division, except for when the divisor is suitably close to zero ("suitably close" is defined in a problem-specific manner). When the divisor is near zero, the protected division operator returns a constant value of large magnitude (again, problem-specific) with the sign determined by the signs of the dividend and divisor. The sign of zero is considered positive. Other definitions of protected division may be suitable for certain problem domains [13].

Terminals represent program inputs, and are always exterior nodes (leaves) in the tree. Some examples include numeric constants, system outputs (y_i), and system states (x_i).

A specialized terminal, known as the *Ephemeral Random Constant*, is a terminal with a constant numeric value. The value is assigned randomly from a designated range when the terminal is created, such as when a random initial tree is created. The

Ephemeral Random Constant is useful for introducing parameter variation into the population.

4.1.2 Genetic Programming

Once suitable function and terminal sets are defined for a problem, the Genetic Programming algorithm may be applied. A Genetic Programming session proceeds according to the flowchart in Figure 4.2.

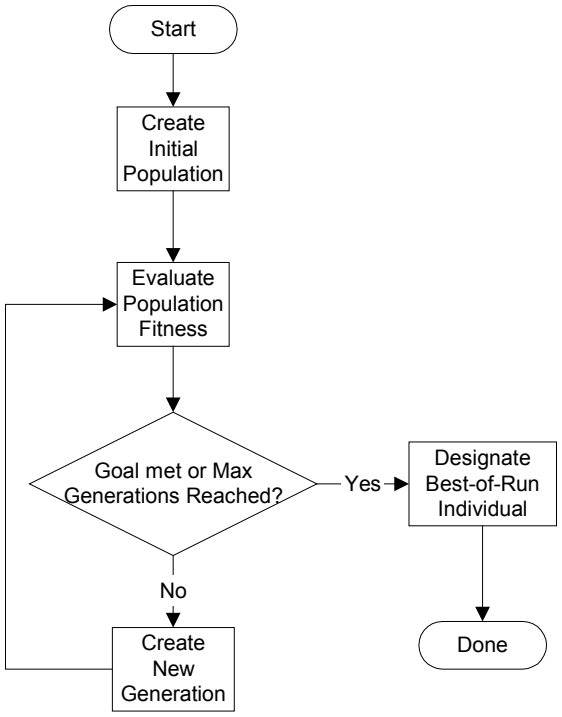

Figure 4.2 Genetic Programming Flowchart

Initial Random Population

First, a population of randomly built trees, or candidates, is created using the chosen function and terminal sets. Typically, limits are placed on the minimum and maximum depth of randomly generated trees.

Fitness

Next, each candidate is evaluated, which determines its *fitness*. A candidate's fitness is a measure of how well it solves the problem, and fitness is determined by a metric known as the *fitness function*. Evaluation typically involves executing the candidate in a predetermined number of conditions, known as *fitness cases*, which are analogous to the training sets used in neural network training. For example, each candidate is executed as the controller of the simulated plant, once for each fitness case. Each fitness case is assigned a number representing the candidate's performance or *raw fitness*. The sum of the raw fitness over all fitness cases is the candidate's raw fitness. Raw fitness may be assigned in any manner natural and useful in the problem domain, e.g. in some domains higher raw fitness corresponds to better performance, and in other domains lower raw fitness corresponds to better performance.

Once all candidates are evaluated, each one's raw fitness is converted to *normalized fitness* via a three-step process:

1. Calculate *standard fitness*. Standard fitness is simply the raw fitness expressed such that all fitness values are positive, and lower fitness values correspond to

better performance. The best possible fitness, if known, corresponds to zero. For many problems, standard fitness is identical to raw fitness.

2. Calculate *adjusted fitness*: Adjusted fitness is defined as $a(i)=\frac{1}{1+s(i)}$

   where *s(i)* is the standardized fitness for individual i. For adjusted fitness, larger values denote better performance. Adjusted fitness exaggerates the difference between relatively very fit individuals (i.e. possess low standard fitnesses), easing the discernment of good candidates from very good candidates.

3. Calculate *normalized fitness*: Normalized fitness is defined as $n(i)=\frac{a(i)}{\sum_{k=1}^{M} a(k)}$

   where *a(i)* is the adjusted fitness of individual *i* and *M* is the population size. For normalized fitness, larger values denote better performance. The normalized fitness rescales the adjusted fitness values so the sum of all fitnesses is one, which is useful for fitness proportionate selection.

Selection

Candidates for the next generation are selected based on the current generation's normalized fitness distribution. *Fitness proportionate* selection is used in this work, although other selection methods exist [13].

The new generation's individuals are selected from the current generation by three methods: reproduction, crossover, and mutation. Figure 4.3 illustrates the GP

selection process. Following the GP flowchart, production of the next generation proceeds as follows:

1. The propagation operator is randomly selected (reproduction, crossover, or mutation). In a given GP run, each operator is assigned a fixed probability.
2. An appropriate number of individuals (one for reproduction and mutation, two for crossover) are selected from the current generation, based on their normalized fitness values. Selection is performed with replacement.
3. The selected operator is applied and the resulting individuals are placed in the new generation (1 for reproduction and mutation, 2 for crossover).

Typically, the best individual is also explicitly copied into the new generation in order to preserve the best result so far.

Selection Operators

Three selection operators are used to propagate individuals from one generation to the next: reproduction, crossover, and mutation. Each operator is an analog of phenomena observed in natural selection [13]. In Genetic Programming, each operator manipulates one or more candidate's tree representation directly.

Reproduction

Reproduction is the wholesale copying of a tree from one generation to the next. No modifications whatsoever are made to the tree. Reproduction is analogous to

"survival of the fittest" in the natural world.

Crossover

The crossover operation is analogous to sexual reproduction in the real world. Sexual reproduction mixes traits from two individuals, the parents, to create offspring that possess some traits from each parent.

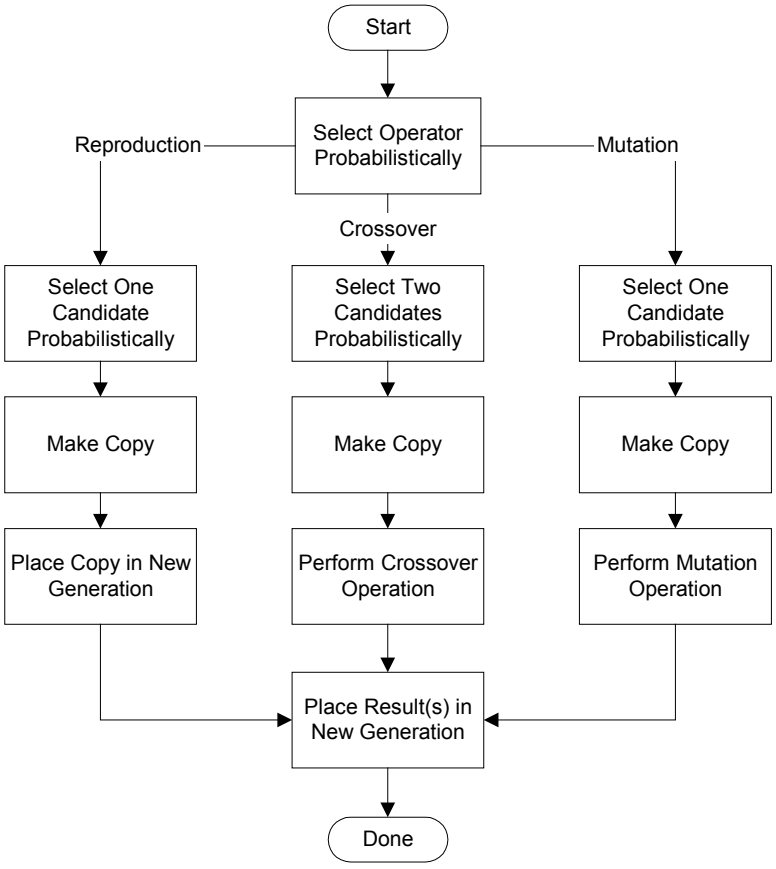

Figure 4.3 Selection Flowchart

In the context of Genetic Programming, Crossover produces offspring by first copying each parent tree, and then swapping randomly chosen subtrees in the offspring's trees. Crossover is illustrated in Figure 4.4.

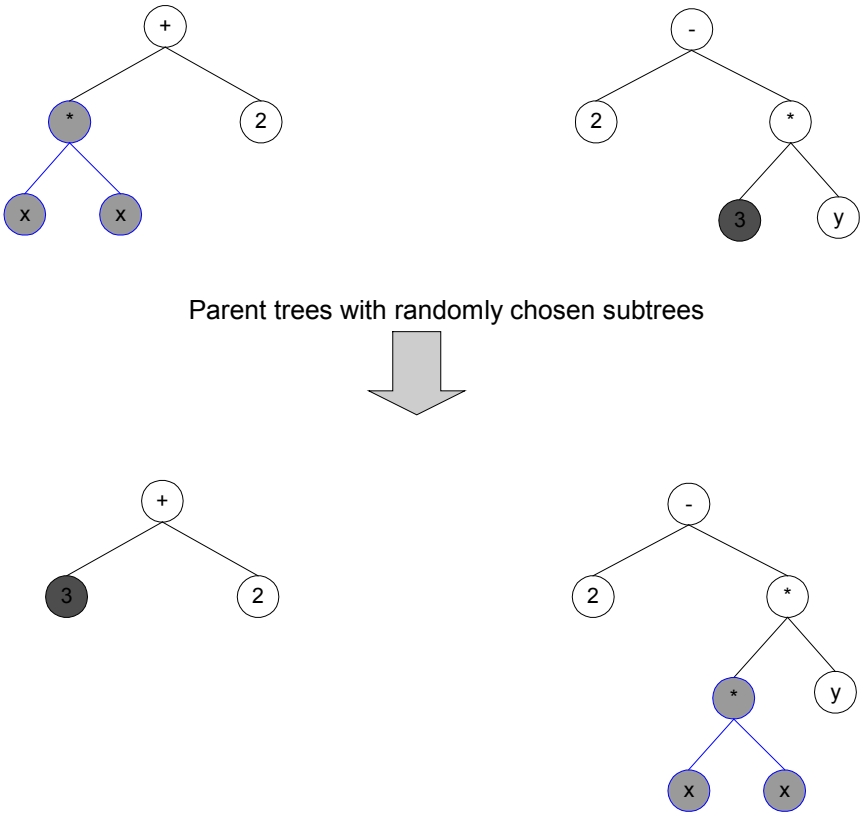

Figure 4.4 Crossover Operation

Mutation

In the real world, genetic mutation occurs when DNA nucleotide sequences change due to copying errors or damage from external forces (e.g. radiation). Mutation introduces truly random changes, whereas crossover merely recombines existing genetic

sequences. Mutation is important in Genetic Programming because it is the only way to introduce novelty into the population. During a run, GP may settle in a local minimum, which cannot be escaped by reshuffling existing individuals. In these situations, random mutations provide the path out of the minimum.

Mutation is achieved by randomly selecting a subtree in a parent tree and replacing it with a randomly generated subtree, while preserving a maximum tree-depth limitation. Figure 4.5 illustrates the mutation operation.

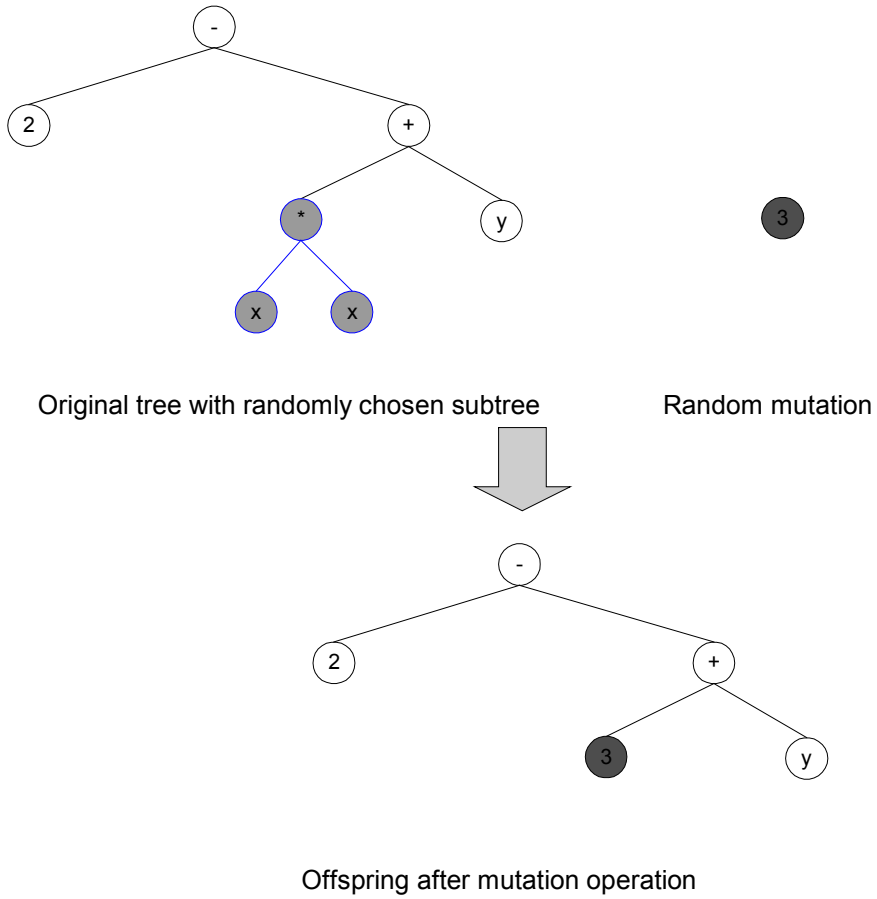

Figure 4.5 Mutation Operation

Termination

The GP algorithm terminates when either the desired fitness is achieved by at least one candidate or the maximum number of generations have elapsed. Otherwise, a new generation is created, using the three propagation operators, from the current generation and the process is repeated.

Incremental Evolution

Some problems may be too difficult to solve all at once. However, if the problem can be broken up into incremental stages or phases, GP may be able to find solutions for each stage in order, and thus achieve the end goal [14]. The incremental approach is analogous to leaving a trail of breadcrumbs for Genetic Programming to follow, where each "crumb" is a separate evolution stage with its own fitness measure. The ultimate goal may not be visible or obviously achievable from the starting point, but the incremental goals provide basic guidance on the path to the solution.

4.2 Genetic Programming for Control Systems

Recall the block diagram for a state feedback regulator, shown in Figure 4.6.

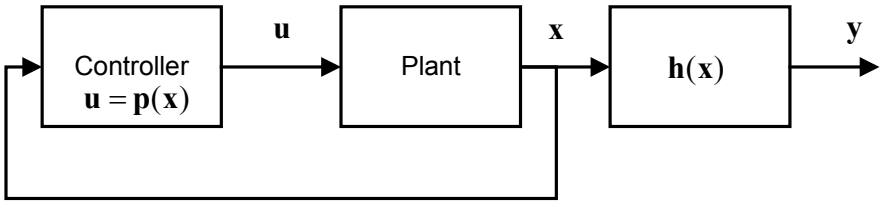

Figure 4.6 Regulator Structure

The block labeled *Controller* is an algebraic function of the plant state **x**:

$$\mathbf{u} = \mathbf{p}(\mathbf{x})$$

Given appropriate function and terminal sets, trees may represent algebraic equations of arbitrary complexity. Genetic Programming may then be used to evolve the trees and hence evolve controllers.

4.2.1 Function and Terminal Sets

Choosing function and terminal sets for a specific problem relies heavily on knowledge of the problem domain. If functions necessary in the solution are excluded, GP will never converge to a satisfactory solution. Likewise, if the function set contains superfluous functions, the convergence rate may become extremely slow, because the search space (all possible trees given the function and terminal set) is much larger than necessary.

The function set chosen for the DiscBot stabilization problem consists of:

- Two-argument multiply (x*y)
- Protected division (x/y)
- Two-argument addition (x+y)

- Two-argument subtraction (x-y)
- Sine
- Cosine

On some runs, an if-then-else function (*if_gt0*) is also used.

The terminal set consists of the plant states (x_1, …, x_n), zero, one, and a few instances of the Ephemeral Random Constant.

- x_1
- x_2
- x_3
- x_4
- 0
- 1
- $\Re$
- $\Re$

4.2.2 Wrapper Function

Each tree's root node is a specialized function called a wrapper. The wrapper node is the interface between the controller that the tree represents and the system it resides in.

In this work, each tree's root node is the "set_x_y" function. The "set_x_y" function possesses as many child trees as there are torque inputs to the mechanical system being interfaced to. For example, the fully actuated DiscBot requires two inputs, and therefore requires a two-argument "set_x_y" function. Likewise, the underactuated DiscBot requires only a one-argument "set_x_y" function. System inputs are numbered left to right. The "set_x_y" function is illustrated in Figure 4.7.

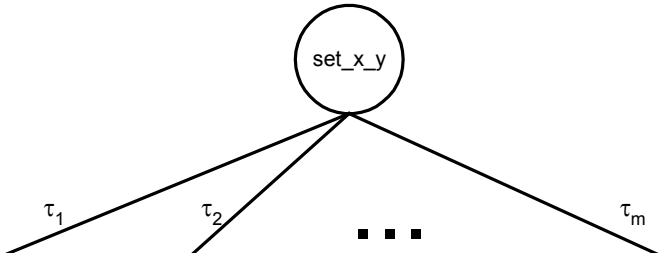

Figure 4.7 "set_x_y" Wrapper Function

4.2.3 Simulation Configuration

To perform a fitness evaluation, each candidate controller is simulated while connected to the DiscBot plant. The system, consisting of a candidate controller, DiscBot plant, output function, and fitness function, is shown in Figure 4.8. A fourth-order Runge-Kutta integration algorithm was used to determine system state.

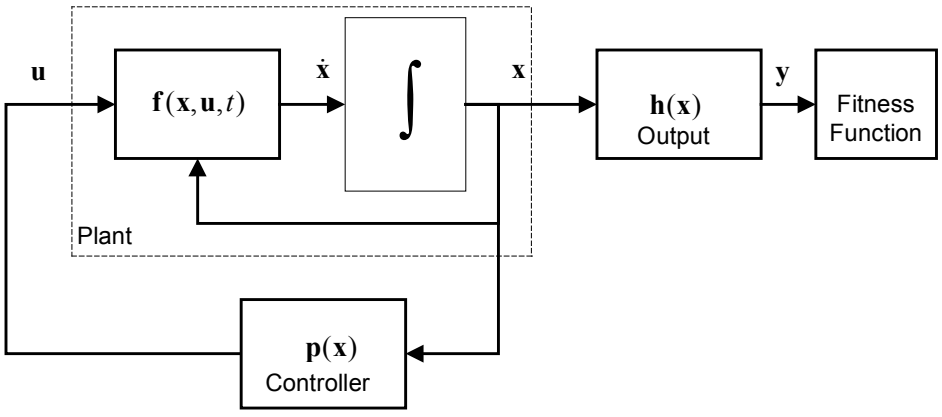

Figure 4.8 Fitness Evaluation Configuration

CHAPTER 5. CONTROLLER PERFORMANCE COMPARISON

Both fully actuated and underactuated versions of the DiscBot are evaluated with multiple control algorithms. The fully actuated DiscBot is considered first. LQR and feedback linearization control schemes are compared to GP-based controllers. Next, the underactuated DiscBot (the normal configuration) is considered. LQR, feedback linearization, and passivity-based control schemes are compared with GP-based controllers evolved via two different performance measures.

The control objective in all cases is to bring the DiscBot state to the origin: the disc at zero displacement, the pendulum standing straight up, and both velocities zero.

5.1 Fully Actuated DiscBot

The fully actuated DiscBot is a DiscBot with an extra actuator added between the disc and ground, as in Figure 5.1. The extra actuator is contrived, and exists only to provide a fully actuated system to evaluate the GP algorithm against.

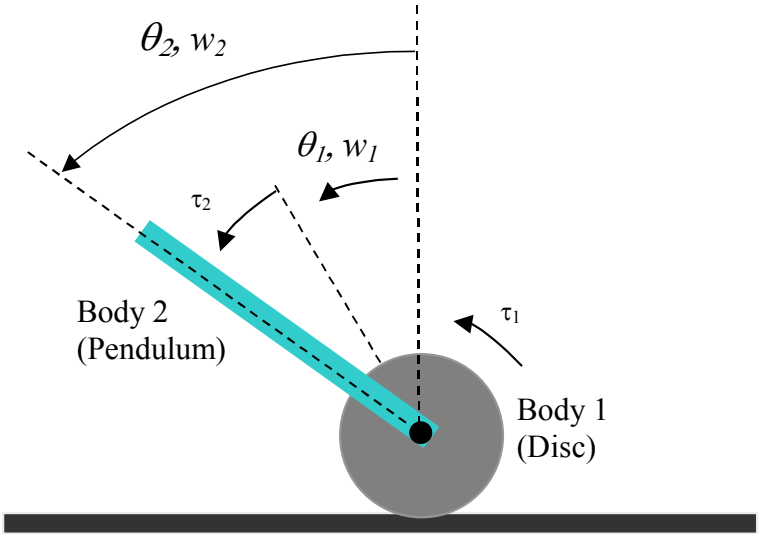

Figure 5.1 Fully Actuated DiscBot

5.1.1 Linear Control

The linear approximation of the fully actuated DiscBot about the origin, given the parameters in Table 2.1, is

$$\dot{\mathbf{x}} \approx \mathbf{A}\mathbf{x} + \mathbf{B}\boldsymbol{\tau}$$

where

$$\mathbf{A} = \begin{bmatrix} 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & -36.6 & 0.0 & 0.0 \\ 0.0 & 25.6 & 0.0 & 0.0 \end{bmatrix}, \text{ and } \mathbf{B} = \begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \\ 12.5 & -16.2 \\ -3.7 & 6.3 \end{bmatrix}. \tag{5.1}$$

The LQR matrices **Q** and **R** are

$$\mathbf{Q} = \begin{bmatrix} 0.2 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.8 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}, \tag{5.2}$$

which lead to the optimal gain

$$F = \begin{bmatrix} 4.23 & 5.08 & 1.13 & 2.10 \\ 1.35 & 25.61 & 1.48 & 6.28 \end{bmatrix}.$$

The LQR controller steers the system state asymptotically to zero, from any initial state within its basin of attraction. Figure 5.2 illustrates LQR controller performance.

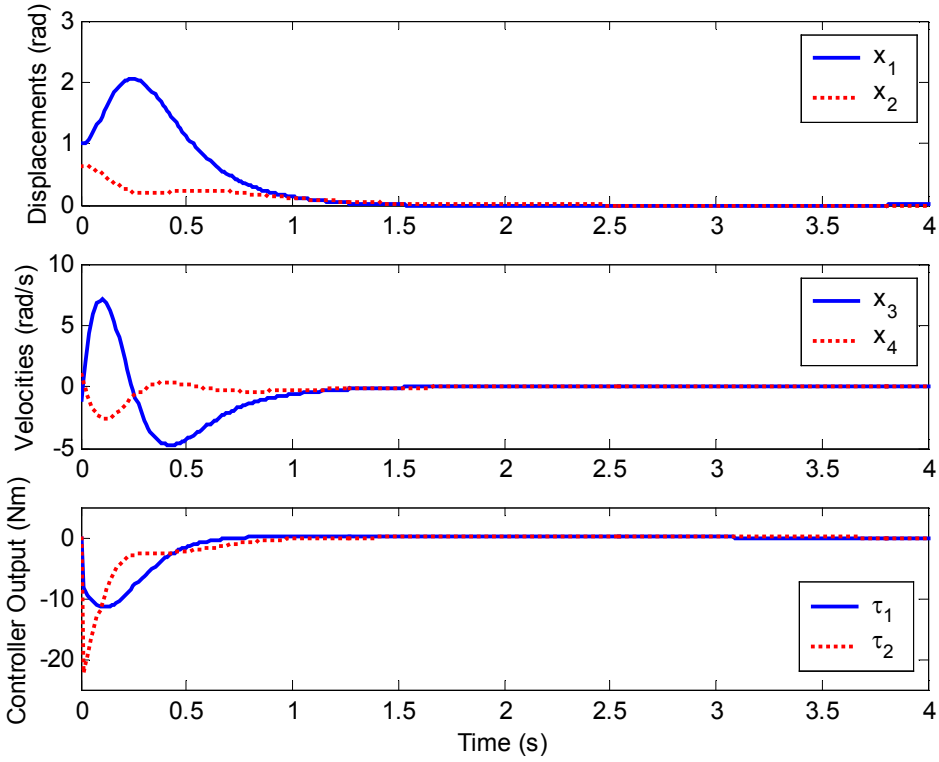

Figure 5.2 LQR Control of Fully Actuated DiscBot, $x_0 = [1\ \pi/5\ -1\ 1]^T$

5.1.2 Feedback Linearization Control

Recall that the input-output linearization is given by

$$\boldsymbol{\tau} = \boldsymbol{\alpha}(\mathbf{x}) + \boldsymbol{\beta}(\mathbf{x})\mathbf{v}$$
where
$$\boldsymbol{\alpha}(\mathbf{x}) = \left(\mathbf{J}\mathbf{W}^{-1}\mathbf{M}\right)^{-1}\left(\mathbf{J}\mathbf{W}^{-1}\left(\mathbf{C} + \mathbf{T}_{\mathbf{g}}\right) - \dot{\mathbf{J}}\dot{\mathbf{q}}\right) \tag{5.3}$$
$$\boldsymbol{\beta}(\mathbf{x}) = \left(\mathbf{J}\mathbf{W}^{-1}\mathbf{M}\right)^{-1} \ .$$

In the fully actuated case the input coupling matrix is

$$\mathbf{M} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}.$$

Taking link displacements q_1 and q_2 as outputs gives

$$\mathbf{h}(\mathbf{x}) = \mathbf{q} \quad \Rightarrow \quad \mathbf{J} = \frac{\partial \mathbf{h}}{\partial \mathbf{q}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Since **M, W,** and **J** are all square and nonsingular, an input-output linearization exists and is given by (5.3). Since there are two displacement outputs, the total relative degree is four. In this case, the total relative degree equals the number of system states, and therefore no internal dynamics exist.

The new states for the linearized system are given by

$$\begin{aligned} z_1 &= y_1 = q_1 \\ z_2 &= y_2 = q_2 \\ z_3 &= \dot{y}_1 = \dot{q}_1 \\ z_4 &= \dot{y}_2 = \dot{q}_2 \\ &\Downarrow \\ \dot{\mathbf{z}} &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\mathbf{z} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}\mathbf{v} = \mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{v} \ . \end{aligned} \tag{5.4}$$

An LQR controller based on the linearized system (5.4) with weighting matrices

$$\mathbf{Q} = \begin{bmatrix} 0.2 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.8 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}, \qquad \mathbf{R} = \begin{bmatrix} 0.01 & 0.00 \\ 0.00 & 0.01 \end{bmatrix},$$

produces the state feedback control law,

$$\mathbf{v} = -\mathbf{F}\mathbf{z}$$

where

$$\mathbf{F} = \begin{bmatrix} 4.47 & 2.99 & 0.00 & 0.00 \\ 0.00 & 0.00 & 89.44 & 13.37 \end{bmatrix},$$

which globally stabilizes the DiscBot system. State trajectories for the fully actuated DiscBot are shown in Figure 5.3 and Figure 5.4 for two sets of initial conditions.

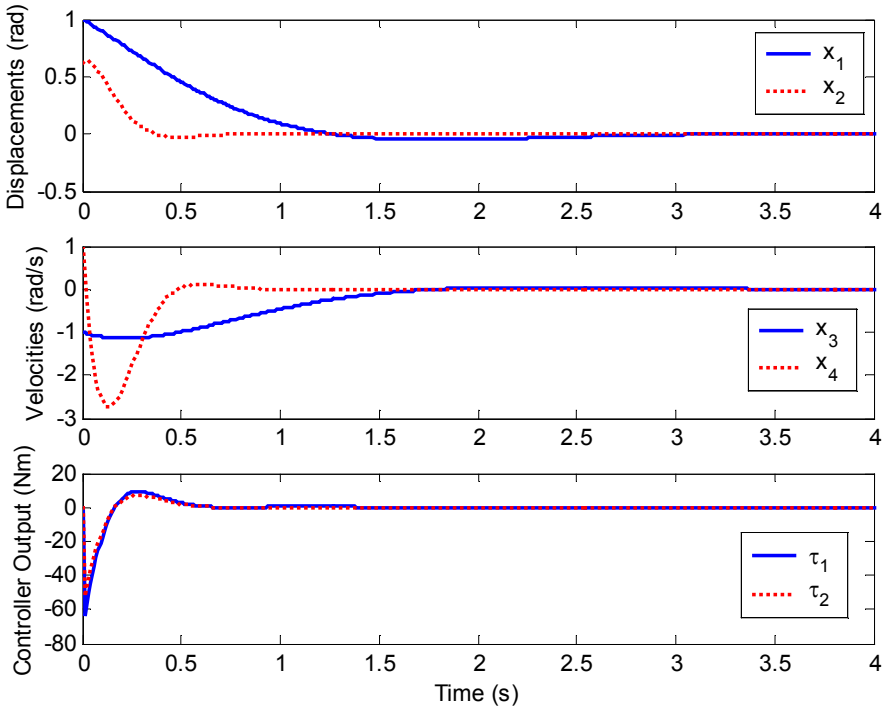

Figure 5.3 FL Control of Fully Actuated DiscBot, $x_0 = [1\ \pi/5\ -1\ 1]^T$

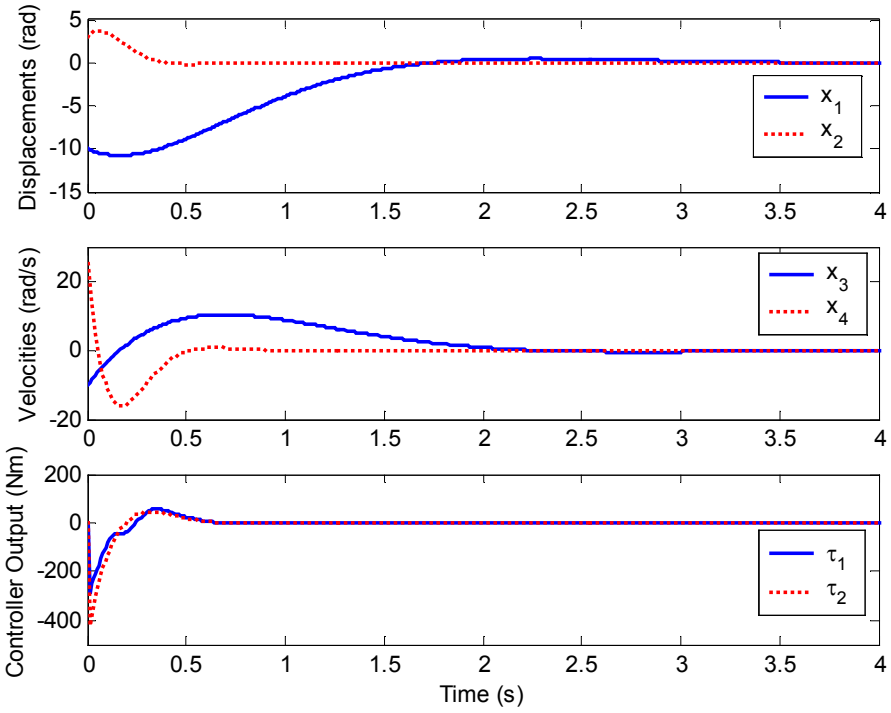

Figure 5.4 FL Control of Fully Actuated DiscBot, $\mathbf{x}_0 = [-10\ \pi\ -10\ 25]^T$

5.1.3 GP-Based Control

A Genetic Programming run was performed on the fully actuated DiscBot, with parameters given in Table 5.1. Each actuator output was also saturated to +/-100 Nm. The fitness function used was

$$J = \int_{t_0}^{t_f} \frac{1}{2}\left[\mathbf{x}^T(t)\mathbf{Q}(t)\mathbf{x}(t) + \boldsymbol{\tau}^T(t)\mathbf{R}\boldsymbol{\tau}(t)\right] dt\,. \tag{5.5}$$

Note the similarity to the familiar LQR performance index, with two exceptions. First, the **R** matrix may be zero, a disallowed condition in the LQR scheme. GP makes no such distinction.

Table 5.1 Fully Actuated DiscBot GP Parameters

Population Size	200
Max. Generations	175
Min. Tree Depth	4
Max. Tree Depth	12
P_r	0.40
P_c	0.30
P_m	0.30
Selection Method	Fitness-Proportionate
Simulation Time	4.0s
Integration Algorithm	Fixed time step 4^{th} order Runge-Kutta
Time Step	10ms

Second, the fitness function is a product of the normal LQR performance index and the simulation time *t*. Extensive evolution attempts revealed that when an LQR performance index is used, the GP system tends to find solutions which are "just stable enough" to minimize the fitness function during the simulation time used in the GP run. These solutions appear to approach a limit cycle. However, when simulated over a larger time scale, it becomes clear that the solution is indeed unstable. Increasing the simulation time in an effort to correct the problem is problematic because excessive computer time is required and the phenomenon is mostly unaffected.

Inclusion of time in the fitness function weighs the integrand greater as simulation time progresses. This causes the GP algorithm to favor individuals that reduce the LQR-like portion of the integrand as time progresses much more than individuals that do not. The longer the simulation time, the more pronounced the effect.

A six-stage incremental approach was used for this GP run. The first four stages weigh pendulum displacement (q_2) very highly, and disc displacement (q_1) very lowly. Initial conditions begin very close to zero to increase at each stage. The fifth and sixth stages shift the weights to put more emphasis on controlling disc displacement. Each stage uses a Raw Fitness threshold to decide when to move on to the next stage, or to finish the run in the case of the last stage. Table 5.2 summarizes the incremental stages used.

Table 5.2 Fully Actuated DiscBot Incremental GP Parameters

Stage	Initial State x_0 Set	Fitness Function Q, R Matrices	Raw Fitness Threshold
1	(0 π/8 0 0)*0.5 (1 0 0 0) *0.5 (1 -π/8 0 0) *0.5 (0 π/4 0 0) *0.5 (1 π/4 0 0) *0.5 (1 -π/4 0 0) *0.5	$\mathbf{Q} = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.99 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	50
2	(0 π/8 0 0) (1 0 0 0) (1 -π/8 0 0) (0 π/4 0 0) (1 π/4 0 0) (1 -π/4 0 0)	$\mathbf{Q} = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.99 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	50
3	(0 π/8 0 0)*1.5 (1 0 0 0) *1.5 (1 -π/8 0 0) *1.5 (0 π/4 0 0) *1.5 (1 π/4 0 0) *1.5 (1 -π/4 0 0) *1.5	$\mathbf{Q} = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.99 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	50

Stage	Initial State x_0 Set	Fitness Function Q, R Matrices	Raw Fitness Threshold
4	(0 π/8 0 0)*2 (1 0 0 0) *2 (1 -π/8 0 0) *2 (0 π/4 0 0) *2 (1 π/4 0 0) *2 (1 -π/4 0 0) *2	$\mathbf{Q}=\begin{bmatrix}0.01 & 0 & 0 & 0\\ 0 & 0.99 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}, \mathbf{R}=\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$	50
5	(0 π/8 0 0) *2 (1 0 0 0) *2 (1 -π/8 0 0) *2 (0 π/4 0 0) *2 (1 π/4 0 0) *2 (1 -π/4 0 0) *2	$\mathbf{Q}=\begin{bmatrix}0.20 & 0 & 0 & 0\\ 0 & 0.80 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}, \mathbf{R}=\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$	100
6	(0 π/8 0 0)*2.25 (1 0 0 0) *2.25 (1 -π/8 0 0) *2.25 (0 π/4 0 0) *2.25 (1 π/4 0 0) *2.25 (1 -π/4 0 0) *2.25	$\mathbf{Q}=\begin{bmatrix}0.20 & 0 & 0 & 0\\ 0 & 0.80 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}, \mathbf{R}=\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$	1

The progression of Best Raw Fitness is shown in Figure 5.5. Best Raw Fitness decreases monotonically, except in generations where the fitness measure changes.

After 150 generations, one GP run produced the individual in Figure 5.6, with a fitness of 66.8. This individual corresponds to the control law

$$\boldsymbol{\tau}=\begin{pmatrix}-6.866\dot{q}_1\\ -56.495\sin q_2\end{pmatrix} \tag{5.6}$$

This controller brings the DiscBot state nearly to zero, which the exception of disc displacement, for all initial conditions evaluated thus far. Proof of stability has not been attempted. Producing a proof of stability is of little value in the GP realm, because

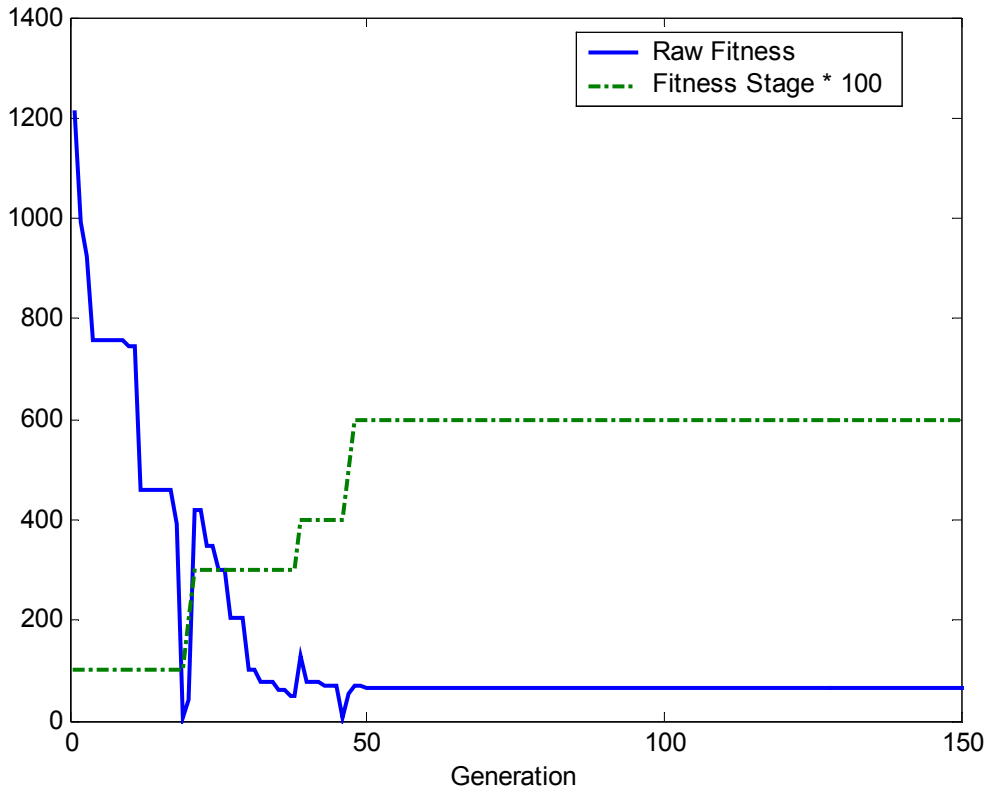

Figure 5.5 Fully Actuated DiscBot Raw Fitness History

the result of any given GP run may in general produce very complex, nonintuitive control laws. Proving stability for such laws is at present an intractable problem. Simply because this particular GP run discovered a simple control law does not mean it is possible to prove stability of GP solutions in general.

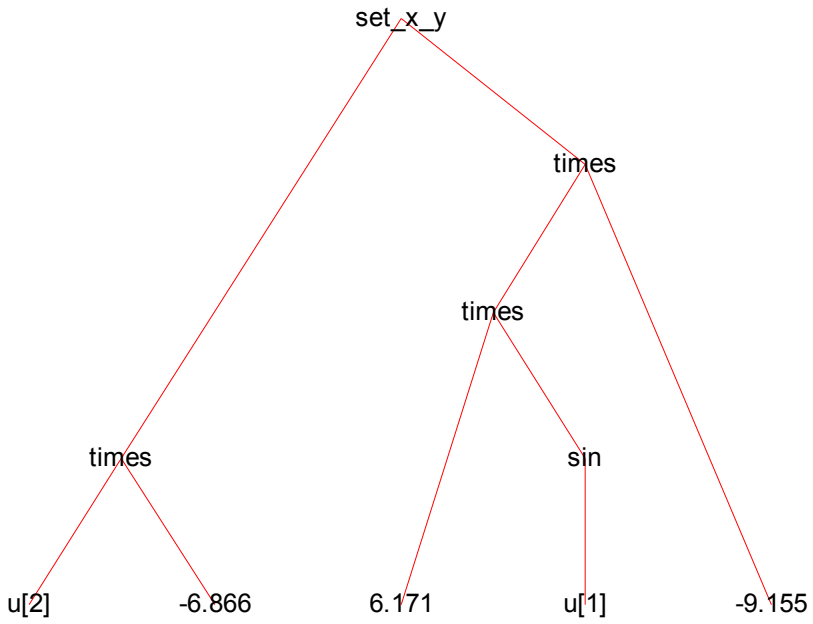

Figure 5.6 Best-of-Run Individual for Fully Actuated DiscBot

The nature of evolutionary methods makes them difficult to analyze. That being said, the control law (5.6) does appear intuitively stable since the control on the disc actuator is a damping term, which we expect to bring disc velocity (but not necessarily displacement) to zero. The pendulum very nearly comes to rest at zero. It does not quite reach zero because, near the origin, the pendulum actuator behaves as a proportional-only controller connected to a linear system. Proportional controllers cannot bring steady-state error to zero [15], although in this particular combination of DiscBot parameters, the pendulum displacement error is very nearly zero.

Controller performances for two different initial conditions are shown in Figure 5.7 and Figure 5.8.

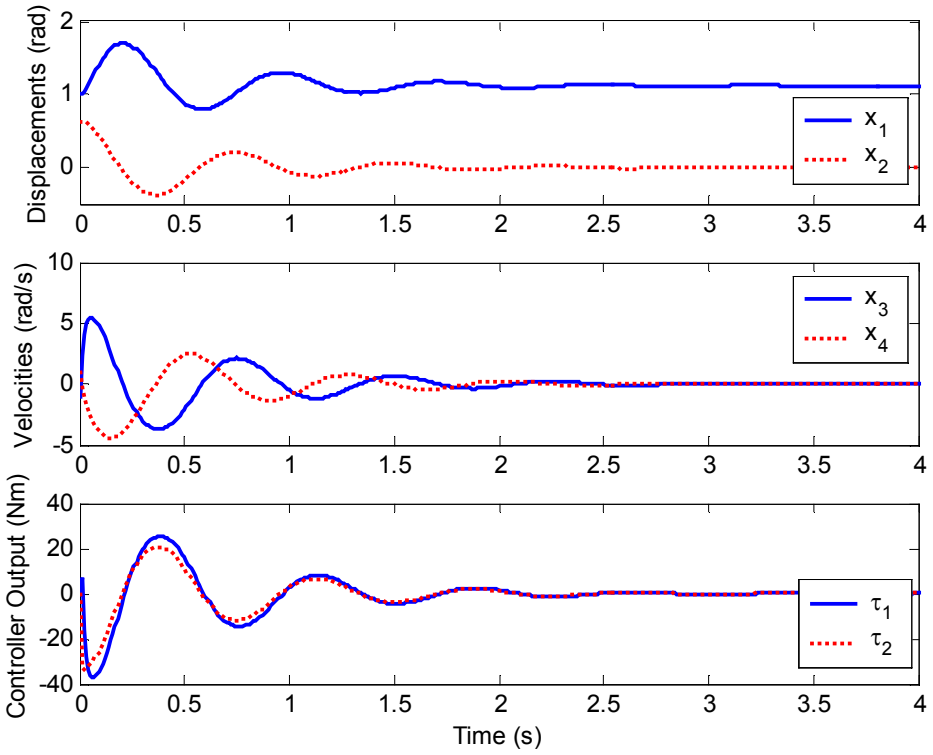

Figure 5.7 GP Control of Fully Actuated DiscBot, $x_0 = [1\ \pi/5\ -1\ 1]^T$

The GP algorithm seems capable of stabilizing a fully actuated DiscBot, which agrees with results for other mechanical systems, found in [2] and [3]. The stabilization is unlikely to be perfect, and some steady-state error will most likely exist. In addition, as typical for Genetic Programming, the solutions may be very difficult to analyze and prove any form of stability, even though the controller appears stable for given sets of initial conditions.

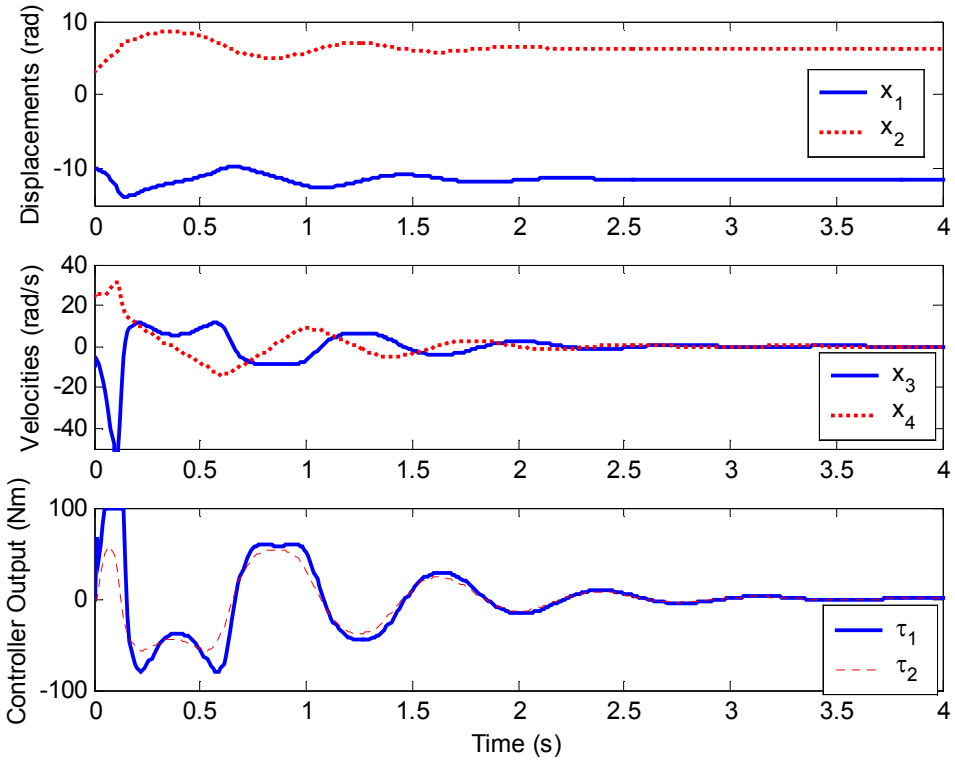

Figure 5.8 GP Control of Fully Actuated DiscBot, $x_0 = [-10\ \pi\ -10\ 25]^T$

5.2 Underactuated DiscBot

5.2.1 Linear Control

The linear approximation of the underactuated DiscBot about the origin, given the parameters in Table 2.1, is

$$\dot{\mathbf{x}} \approx \mathbf{A}\mathbf{x} + \mathbf{B}\boldsymbol{\tau}$$

where

$$\mathbf{A} = \begin{bmatrix} 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & -36.6 & 0.0 & 0.0 \\ 0.0 & 25.6 & 0.0 & 0.0 \end{bmatrix}, \text{ and } \mathbf{B} = \begin{bmatrix} 0.0 \\ 0.0 \\ -16.2 \\ 6.3 \end{bmatrix}. \tag{5.7}$$

The LQR matrices **Q** and **R** are

$$\mathbf{Q} = \begin{bmatrix} 0.2 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.8 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} 0.01 \end{bmatrix}, \tag{5.8}$$

which lead to the optimal gain

$$\mathbf{F} = \begin{bmatrix} 4.47 & 44.57 & 3.54 & 12.28 \end{bmatrix}.$$

The LQR controller steers system state asymptotically to zero, from any initial state within its basin of attraction. Figure 5.9 illustrates typical LQR controller performance.

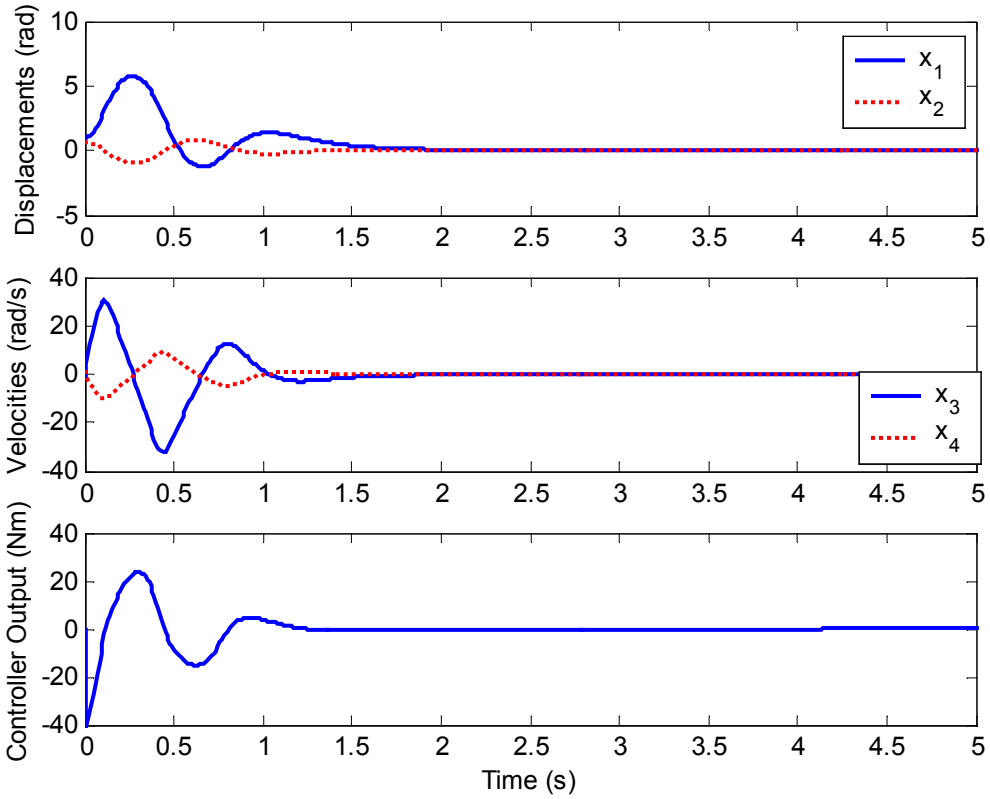

Figure 5.9 LQR Control of Underactuated DiscBot, $x_0 = [1\ \pi/5\ -1\ 1]^T$

When the LQR is connected to the full nonlinear plant, it can stabilize only when the initial state is near the origin. Figure 5.10 demonstrates an initial state for which the LQR fails to stabilize the system.

A numerical study of LQR stability was performed on the DiscBot. Several simulations were run, starting from different initial conditions (all of zero velocity). The region of LQR stability, given zero initial velocity, is shown in Figure 5.11.

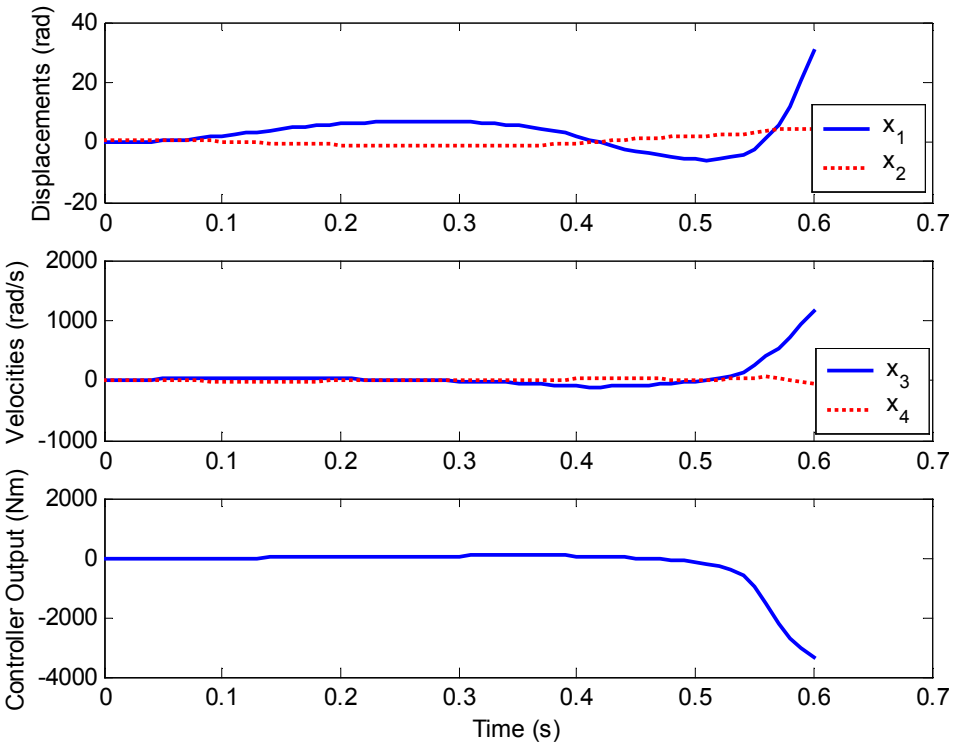

Figure 5.10 LQR Control of Underactuated DiscBot, $x_0 = [0\ 5\pi/16\ 0\ 0]^T$

5.2.2 Feedback Linearization Control

The DiscBot contains only one input, but we wish to control two outputs: the disc and pendulum displacements. Since Feedback Linearization applies only to square systems, only one output can be linearized. The pendulum displacement is chosen, as its stabilization is the primary control task.

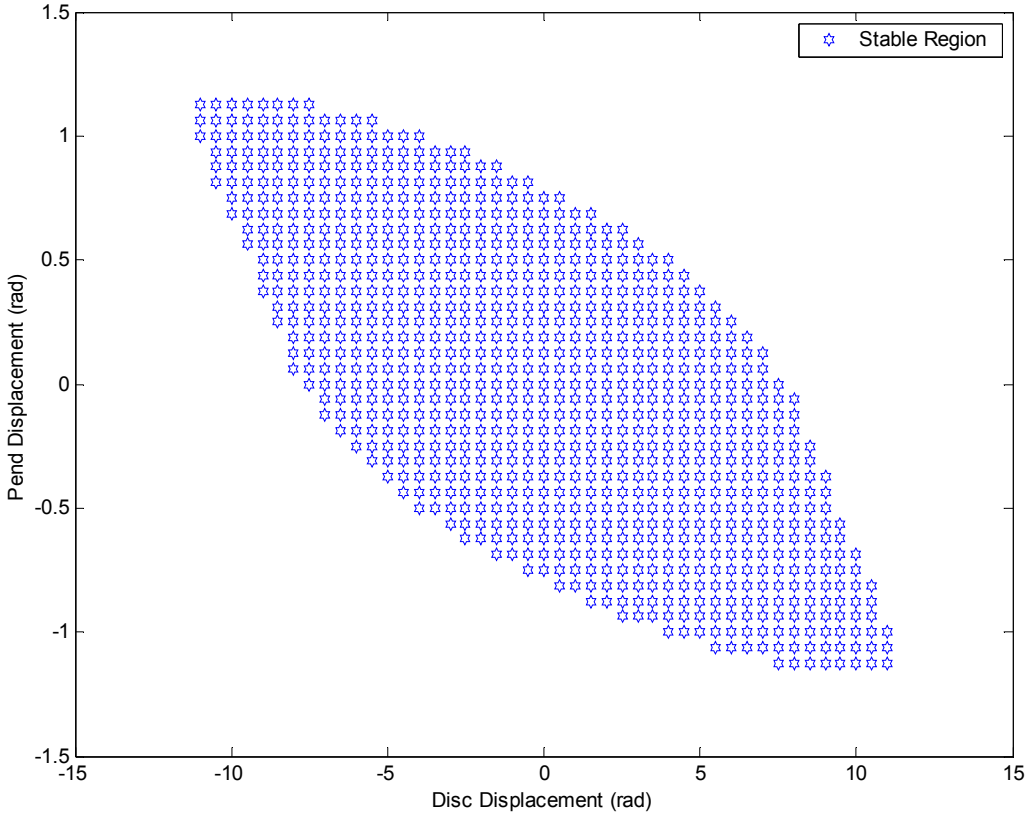

Figure 5.11 Region of Stable Initial Conditions for LQR (zero initial velocity)

Recall that the input-output linearization is given by

$$\tau = \boldsymbol{\alpha}(\mathbf{x}) + \boldsymbol{\beta}(\mathbf{x})\mathbf{v}$$

where

$$\boldsymbol{\alpha}(\mathbf{x}) = \left(\mathbf{J}\mathbf{W}^{-1}\mathbf{M}\right)^{-1}\left(\mathbf{J}\mathbf{W}^{-1}\left(\mathbf{C}+\mathbf{T}_{\mathbf{g}}\right)-\dot{\mathbf{J}}\dot{\mathbf{q}}\right) \tag{5.9}$$

$$\boldsymbol{\beta}(\mathbf{x}) = \left(\mathbf{J}\mathbf{W}^{-1}\mathbf{M}\right)^{-1} \quad .$$

In the underactuated case the input coupling matrix is

$$\mathbf{M} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

Taking the pendulum displacement q_2 as output gives

$$\mathbf{h}(\mathbf{x}) = q_2 \quad \Rightarrow \quad \mathbf{J} = \frac{\partial \mathbf{h}}{\partial \mathbf{q}} = \begin{bmatrix} 0 & 1 \end{bmatrix}.$$

Since $\mathbf{JW^{-1}M}$ is nonsingular, an input-output linearization exists and is given by (5.9). The total relative degree is two, since there is one displacement output. In this case, the total relative degree is less than number of system states (four), and internal dynamics exist.

The new states for the linearized system are given by

$$\begin{aligned} & z_1 = y_1 = q_2 \\ & z_2 = \dot{y}_1 = \dot{q}_2 \\ & \Downarrow \\ & \dot{\mathbf{z}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{z} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{v} = \mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{v} \end{aligned} \tag{5.10}$$

with internal dynamics given by

$$\begin{aligned} & \zeta_1 = q_1 \\ & \zeta_2 = \dot{q}_1 \\ & \Downarrow \\ & \dot{\zeta}_1 = \zeta_2 \\ & \dot{\zeta}_2 = \ddot{q}_1 = \begin{bmatrix} 1 & 0 \end{bmatrix} \ddot{\mathbf{q}} = -\begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{W}^{-1} \left(\mathbf{C} + \boldsymbol{\tau}_\mathbf{g} \right) + \mathbf{W}^{-1} \mathbf{M} \tau \end{aligned} \tag{5.11}$$

where τ is given by (5.9).

An LQR controller based on the linearized portion of the system, with weighting matrices

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \qquad R = \begin{bmatrix} 0.01 \end{bmatrix},$$

produces the state feedback control law

$$v = -\mathbf{F}\mathbf{z}$$
where
$$\mathbf{F} = \begin{bmatrix} 10.00 & 4.47 \end{bmatrix}.$$

To determine the stability of the internal dynamics, the outputs are made identically zero,

$$h(\mathbf{q}) = \mathbf{0} \Rightarrow q_2 = 0 \tag{5.12}$$

which implies

$$\begin{aligned} \dot{q}_2 &= 0 \\ \ddot{q}_2 &= v = 0 \ . \end{aligned} \tag{5.13}$$

Under this constraint, the centripetal/Coriolis and gravitation terms become

$$\begin{aligned} \mathbf{C}(\mathbf{q},\mathbf{q})\mathbf{q} &= \begin{bmatrix} -m_2 rL \sin q_2 q_2^2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \boldsymbol{\tau}_g &= \begin{bmatrix} 0 \\ -m_2 gL \sin q_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned} \ . \tag{5.14}$$

The input becomes

$$\begin{aligned} &\tau = \boldsymbol{\alpha}(\mathbf{x}) + \boldsymbol{\beta}(\mathbf{x})\mathbf{v} = \boldsymbol{\alpha}(\mathbf{x}) = 0 \\ &\text{where} \\ &\boldsymbol{\alpha}(\mathbf{x}) = \left(\mathbf{J}\mathbf{W}^{-1}\mathbf{M}\right)^{-1}\left(\mathbf{J}\mathbf{W}^{-1}\left(\mathbf{C} + \mathbf{T}_\mathbf{g}\right) - \dot{\mathbf{J}}\dot{\mathbf{q}}\right) \\ &\boldsymbol{\alpha}(\mathbf{x}) = 0 \ . \end{aligned} \tag{5.15}$$

Combining (5.12) through (5.15) yields the internal dynamics under the zero-output constraint, or the *zero dynamics:*

$$\begin{aligned}
&\dot{\zeta}_1 = \zeta_2 \\
&\dot{\zeta}_2 = \ddot{q}_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}\ddot{\mathbf{q}} = -\begin{bmatrix} 1 & 0 \end{bmatrix}\mathbf{W}^{-1}\left(\mathbf{C} + \boldsymbol{\tau}_g\right) + \begin{bmatrix} 1 & 0 \end{bmatrix}\mathbf{W}^{-1}\mathbf{M}\tau = 0 \\
&\Downarrow \\
&\dot{\boldsymbol{\zeta}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\boldsymbol{\zeta} \,.
\end{aligned} \tag{5.16}$$

The zero dynamics are a marginally stable linear system. This means that if the disc velocity is nonzero when the pendulum becomes stabilized, it will remain at that value indefinitely. Therefore, straightforward application of Feedback Linearization can stabilize the pendulum, but cannot control the disc displacement. An example is given in Figure 5.12.

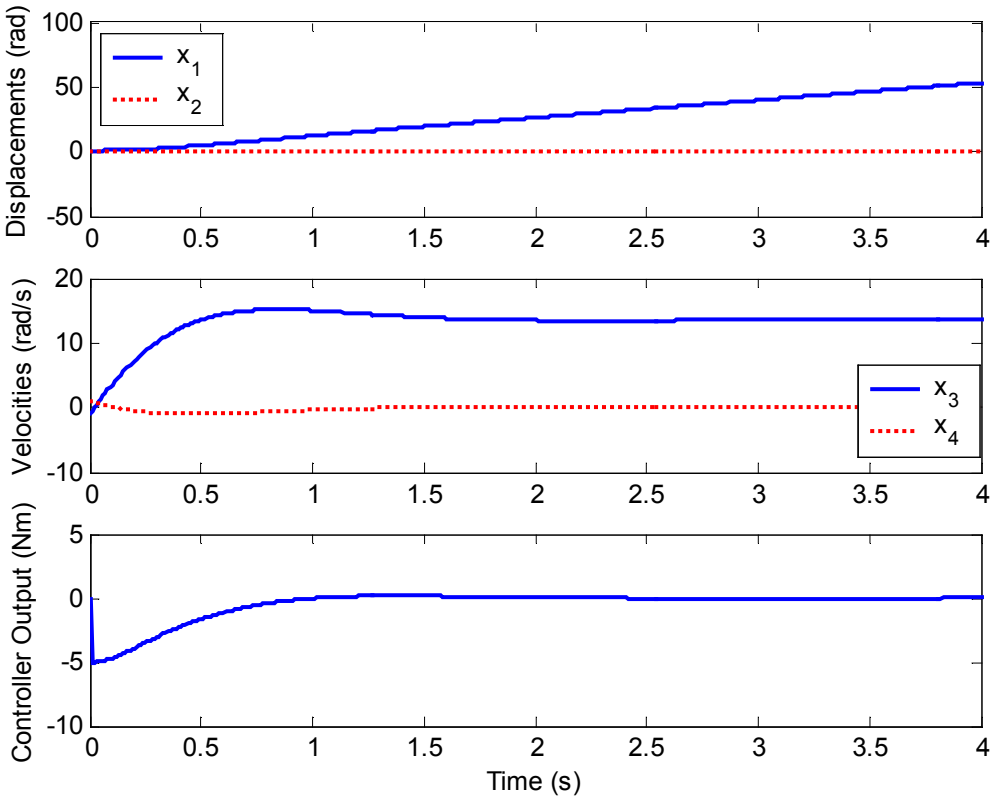

Figure 5.12 Underactuated DiscBot under Feedback Linearization Control

5.2.3 Passivity-Based Control

By considering the total energy of the DiscBot

$$E = K(\dot{q}) + P(q)$$
$$E = \dot{\mathbf{q}}^{\mathbf{T}}\mathbf{W}(\mathbf{q})\dot{\mathbf{q}} + P(\mathbf{q})$$

its derivative can be calculated as

$$\dot{E} = \dot{\mathbf{q}}^{\mathbf{T}}\mathbf{W}(\mathbf{q})\ddot{\mathbf{q}} + \tfrac{1}{2}\dot{\mathbf{q}}^{\mathbf{T}}\dot{\mathbf{W}}\dot{\mathbf{q}} + \dot{\mathbf{q}}^{\mathbf{T}}\boldsymbol{\tau}_g(\mathbf{q})$$
$$\text{where } \boldsymbol{\tau}_g(\mathbf{q}) = \frac{\partial P}{\partial \mathbf{q}}$$
$$\dot{E} = \dot{\mathbf{q}}^{\mathbf{T}}\left(-\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} - \boldsymbol{\tau}_g(\mathbf{q}) + \mathbf{M}u\right) + \tfrac{1}{2}\dot{\mathbf{q}}^{\mathbf{T}}\dot{\mathbf{W}}\dot{\mathbf{q}} + \dot{\mathbf{q}}^{\mathbf{T}}\boldsymbol{\tau}_g(\mathbf{q})$$
$$\dot{E} = \dot{\mathbf{q}}^{\mathbf{T}}\left(\tfrac{1}{2}\dot{\mathbf{W}} - \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\right)\dot{\mathbf{q}} + \dot{\mathbf{q}}^{\mathbf{T}}\mathbf{M}\tau \ .$$

Recalling the skew-symmetric property of $\left(\dot{\mathbf{W}} - \mathbf{2C}\right)$, we are left with

$$\dot{E} = \dot{\mathbf{q}}^{\mathbf{T}}\mathbf{M}\tau$$
$$\dot{E} = \dot{\mathbf{q}}^{\mathbf{T}}\begin{bmatrix} -1 \\ 1 \end{bmatrix}\tau = (\dot{q}_2 - \dot{q}_1)\tau = \dot{\tilde{q}}\tau \qquad (5.17)$$
$$\text{where}$$
$$\tilde{q} = q_2 - q_1 = \mathbf{Mq}, \qquad \dot{\tilde{q}} = \dot{q}_2 - \dot{q}_1 = \mathbf{M}\dot{\mathbf{q}} \ .$$

This result simply states that the total power flowing into the system is equal to the power generated by the motor at the joint between the disc and pendulum, which is defined as the product of the motor torque and the joint's relative rotational velocity.

Now a candidate Lyapunov function,

$$V(\mathbf{q},\dot{\mathbf{q}}) = \tfrac{1}{2}k_E(E - E_d)^2 + \tfrac{1}{2}k_v\dot{\tilde{q}}^2 + \tfrac{1}{2}k_p\tilde{q}^2, \qquad (5.18)$$

may be constructed as in [12]. Its time derivative is

$$\dot{V}(x) = k_E E\dot{E} + k_v\dot{\tilde{q}}\ddot{\tilde{q}} + k_p\tilde{q}\dot{\tilde{q}} \ .$$

Substituting the passivity property from (5.17) yields

$$\dot{V}(x) = k_E E\dot{\tilde{q}}\tau + k_v\dot{\tilde{q}}\ddot{\tilde{q}} + k_p\tilde{q}\dot{\tilde{q}}$$
$$\dot{V}(x) = \left[k_E E\tau + k_v\ddot{\tilde{q}} + k_p\tilde{q}\right]\dot{\tilde{q}} \ .$$

Substitution of the system dynamics (2.4) yields

$$\begin{aligned}
\dot{V}(x) &= \left[k_E E\tau + k_v\mathbf{M}^T\left(-\mathbf{W}(\mathbf{q})^{-1}\left(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \boldsymbol{\tau}_\mathbf{g}(\mathbf{q})\right) + \mathbf{W}(\mathbf{q})^{-1}\mathbf{M}\tau\right) + k_p\tilde{q}\right]\dot{\tilde{q}} \\
\dot{V}(x) &= \left[k_E E\tau - k_v\mathbf{M}^T\mathbf{W}(\mathbf{q})^{-1}\left(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \boldsymbol{\tau}_\mathbf{g}(\mathbf{q})\right) + k_v\mathbf{M}^T\mathbf{W}(\mathbf{q})^{-1}\mathbf{M}\tau + k_p\tilde{q}\right]\dot{\tilde{q}} \\
\dot{V}(x) &= \left[k_p\tilde{q} - k_v\mathbf{M}^T\mathbf{W}(\mathbf{q})^{-1}\left(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \boldsymbol{\tau}_\mathbf{g}(\mathbf{q})\right) + \left(k_E E + k_v\mathbf{M}^T\mathbf{W}(\mathbf{q})^{-1}\mathbf{M}\right)\tau\right]\dot{\tilde{q}}
\end{aligned} \tag{5.19}$$

Now $\dot{V}(\mathbf{q},\dot{\mathbf{q}})$ is *defined* to be non-increasing, as

$$\dot{V}(\mathbf{q},\dot{\mathbf{q}}) = -k_\delta\dot{\tilde{q}}^2 \ ,$$

which leads to

$$k_p\tilde{q} - k_v\mathbf{M}^T\mathbf{W}(\mathbf{q})^{-1}\left(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \boldsymbol{\tau}_\mathbf{g}(\mathbf{q})\right) + \left(k_E E + k_v\mathbf{M}^T\mathbf{W}(\mathbf{q})^{-1}\mathbf{M}\right)\tau = -k_\delta\dot{\tilde{q}} \ .$$

Solving for τ,

$$\tau = \frac{-k_p\tilde{q} - k_\delta\dot{\tilde{q}} + k_v\mathbf{M}^T\mathbf{W}(\mathbf{q})^{-1}\left(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \boldsymbol{\tau}_\mathbf{g}(\mathbf{q})\right)}{\left(k_E E + k_v\mathbf{M}^T\mathbf{W}(\mathbf{q})^{-1}\mathbf{M}\right)}, \tag{5.20}$$

which has a solution so long as

$$k_E E + k_v\mathbf{M}^T\mathbf{W}(\mathbf{q})^{-1}\mathbf{M} \neq 0 \quad . \tag{5.21}$$

One way to enforce (5.21) is to constrain it such that

$$k_E E + k_v\mathbf{M}^T\mathbf{W}(\mathbf{q})^{-1}\mathbf{M} > 0 \ .$$

The minimum energy possible in the DiscBot system occurs when joint velocities are zero and the pendulum points straight down ($\mathbf{x} = [0\ \pi\ 0\ 0]^T$):

$$E_{\min} = -m_2 gL \ .$$

Substitution of E_{min} and expanding produces

$$-k_E m_2 g L + k_v \left(\frac{w_{11} w_{22} - w_{12}^2}{w_{11} + w_{22} + 2w_{12}} \right) > 0$$

where

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}, \qquad w_{21} = w_{12} \ .$$

Rearranging the above expression yields the constraint

$$\frac{k_v}{k_E} > m_2 g L \left(\frac{w_{11} + w_{22} + 2w_{12}}{w_{11} w_{22} - w_{12}^2} \right). \tag{5.22}$$

The maximum conceivable value of (5.22) occurs at

$$\frac{k_v}{k_E} > m_2 g L \left(\frac{\max\left(w_{11} + w_{22} + 2w_{12}\right)}{\min\left(w_{11} w_{22} - w_{12}^2\right)} \right).$$

Recalling the definition of **W** for the DiscBot,

$$W = \begin{bmatrix} (m_1 + m_2) r^2 + J_1 & m_2 r L \cos q_2 \\ m_2 r L \cos q_2 & m_2 L^2 + J_2 \end{bmatrix},$$

it is clear that the numerator is maximized for q_2=0 and the denominator is minimized for q_2=π. The constraint then becomes

$$\frac{k_v}{k_E} > m_2 g L \left(\frac{w_{11} + w_{22} + 2m_2 r L}{w_{11} w_{22} - \left(m_2 r L\right)^2} \right).$$

Given the parameters given in Table 2.1, this translates to

$$\frac{k_v}{k_E} > 221.3 \, .$$

From (5.20) it is also clear that τ is zero at the lower equilibrium point ($\mathbf{x} = [0 \ \pi \ 0 \ 0]^T$). In order to avoid getting stuck at this low point, we can constrain the initial energy of

the system, and rely on the non-increasing property of $V(\mathbf{q},\dot{\mathbf{q}})$ to ensure that the system energy never approaches that of the lower equilibrium point.

We require

$$\left|E(0)-E_d\right| < E_{\min}. \tag{5.23}$$

Since V is a non-increasing function, then if

$$\begin{aligned} V(0) &< \tfrac{1}{2}K_E(E_{\min})^2 \\ V(0) &< \tfrac{1}{2}K_E(m_2gL)^2 \end{aligned} \tag{5.24}$$

then (5.23) will always remain true. Referring to (5.18) in light of (5.24), it is apparent that a new constraint is imposed on the relationship between E(0), K_E, K_p, and K_v:

$$\begin{aligned} V(0) &= \tfrac{1}{2}k_E(E(0)-E_d)^2 + \tfrac{1}{2}k_v\dot{\tilde{q}}^2 + \tfrac{1}{2}k_p\tilde{q}^2 < \tfrac{1}{2}K_E(m_2gL)^2 \\ k_E(E(0)-E_d)^2 &+ k_v\dot{\tilde{q}}(0)^2 + k_p\tilde{q}(0)^2 < K_E(m_2gL)^2 \\ k_p &< \frac{-k_E(E(0)-E_d)^2 - k_v\dot{\tilde{q}}(0)^2 + K_E(m_2gL)^2}{\tilde{q}(0)^2}. \end{aligned}$$

To summarize, the control law (5.20) is stable in the Lyapunov sense if the following constraints are imposed:

$$\frac{k_v}{k_E} > m_2gL\left(\frac{w_{11}+w_{22}+2m_2rL}{w_{11}w_{22}-(m_2rL)^2}\right)$$

$$\left|E(0)-E_d\right| < E_{\min}$$

$$k_p < \frac{-k_E(E(0)-E_d)^2 - k_v\dot{\tilde{q}}(0)^2 + K_E(m_2gL)^2}{\tilde{q}(0)^2}.$$

Note that if the initial kinetic energy is zero, then the pendulum-down equilibrium point must not be an initial condition. In addition, since the initial displacement is arbitrary,

K_p should decrease as initial disc displacement increases. However, doing so decreases the convergence rate.

Stability Analysis

The system's trajectory may be proved to converge to a homoclinic orbit by using LaSalle's Theorem. The proof follows similar ones in [12]. The homoclinic orbit can be determined by substituting the desired system energy and state into the system energy equation:

$$E = K(\dot{q}) + P(q)$$
$$E = \dot{\mathbf{q}}^{\mathrm{T}}\mathbf{W}(\mathbf{q})\dot{\mathbf{q}} + P(\mathbf{q}) \ .$$

At the origin, the DiscBot has no kinetic energy and its potential energy is given by

$$P_0 = m_2 gL\cos 0 = m_2 gL = E_0$$

and the displacements by

$$q_1 = q_2 = 0 \, .$$

The kinetic energy is then

$$K = \dot{\mathbf{q}}^{\mathrm{T}}\mathbf{W}(\mathbf{q})\dot{\mathbf{q}}$$
$$K = \begin{bmatrix} q_2 & q_2 \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} \\ w_{12} & w_{22} \end{bmatrix} \begin{bmatrix} q_2 \\ q_2 \end{bmatrix} .$$
$$K = \dot{q}_2^2 \left(w_{11} + w_{22} + 2w_{12} \right)$$

Substituting into the total system energy yields the expression for the homoclinic orbit:

$$E = K + P$$
$$m_2 gL = \dot{q}_2^2 \left(w_{11} + w_{22} + 2w_{12} \right) + m_2 gL \cos q_2$$
$$\dot{q}_2 = \sqrt{\frac{m_2 gL\left(1 - \cos q_2\right)}{\left(w_{11} + w_{22} + 2w_{12}\right)}}$$

where

$$\mathbf{W} = \begin{bmatrix} \left((m_1 + m_2) r^2 + J_1 \right) & m_2 rL \cos q_2 \\ m_2 rL \cos q_2 & m_2 L^2 + J_2 \end{bmatrix}.$$

Figure 5.13 illustrates the homoclinic orbit.

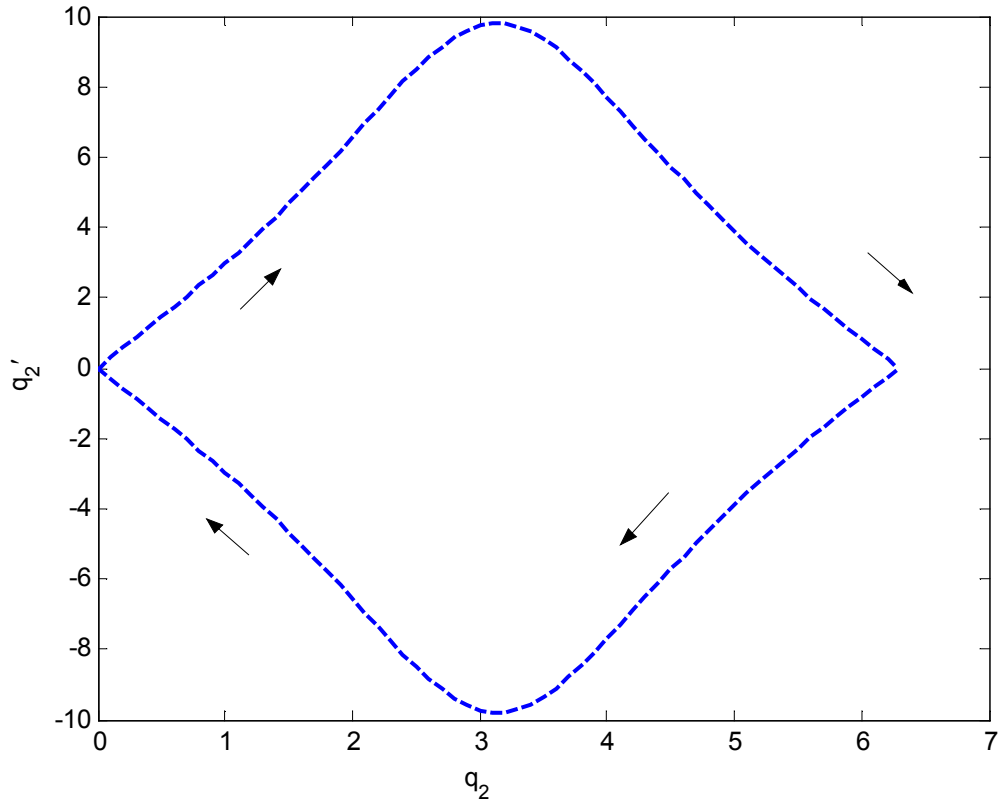

Figure 5.13 Homoclinic orbit

Switching Control

Since the Passivity-Based Controller converges to a homoclinic orbit, and not a fixed point, it will never stop swinging back and forth. To stabilize the controller to a fixed point, control must be switched to another controller.

After the Passivity controller brings the system state within the basin of attraction of the LQR controller considered earlier, control is switched to it, which brings the DiscBot state to the origin.

Simulation Results

Figure 5.14 and Figure 5.15 demonstrate the Passivity-Based Controller given initial state $x_0 = [1 \ \pi/5 \ -1 \ 1]^T$. Note the spike in $V(q,\dot{q})$, which occurs at the switchover from passive control to LQR control. Figure 5.16 illustrates convergence to the homoclinic orbit. Figure 5.17, Figure 5.18, and Figure 5.19 illustrate similar histories for the initial state $x_0 = [-10 \ \pi \ -1 \ 0]^T$. Note how much longer is required to stabilize when the pendulum displacement is far from zero.

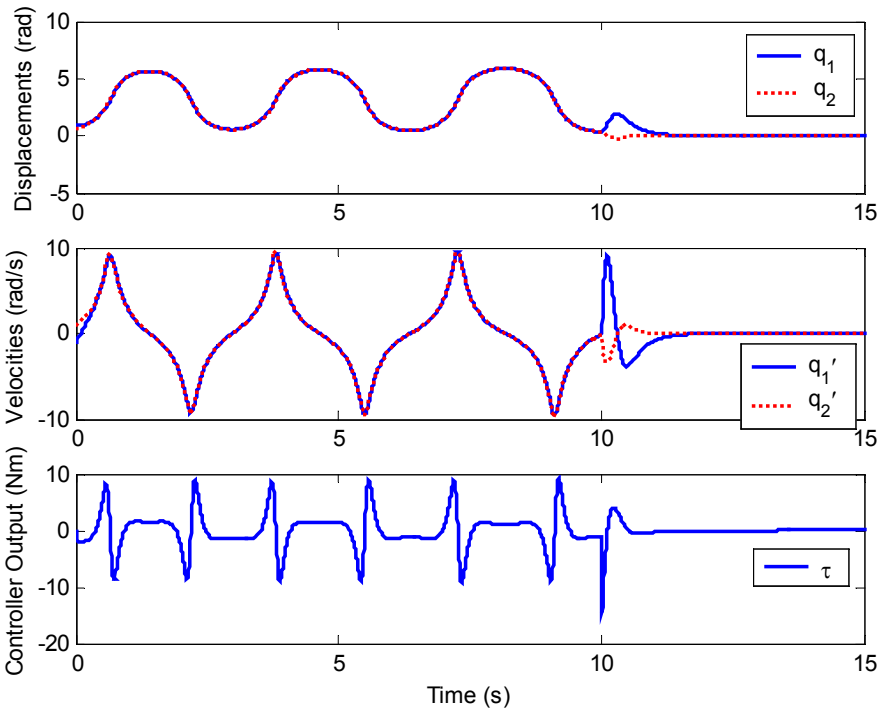

Figure 5.14 Passivity-Based Control of Underactuated DiscBot, $x_0 = [1\ \pi/5\ -1\ 1]^T$

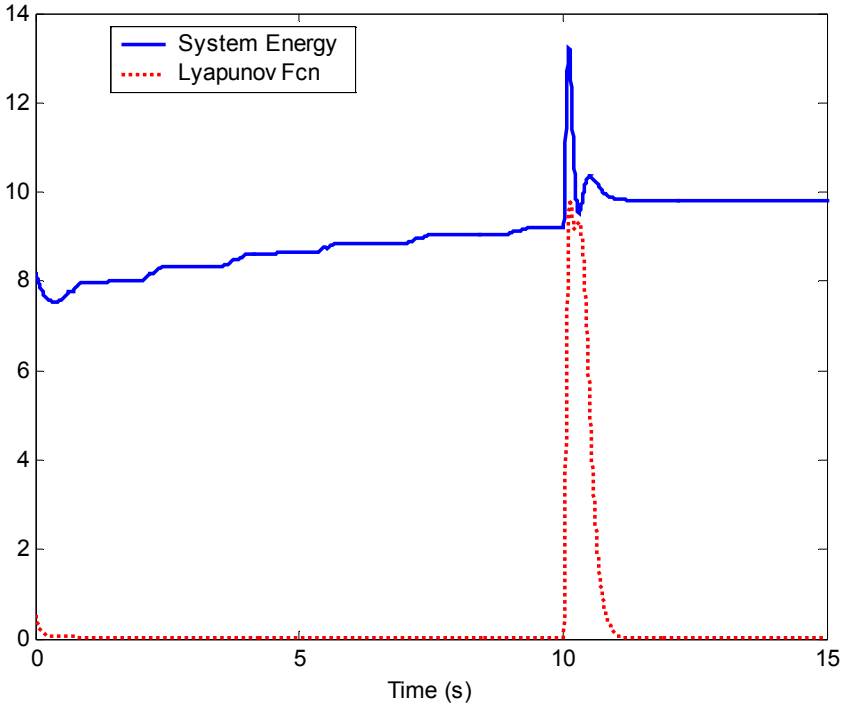

Figure 5.15 Lyapunov function time history for $x_0 = [1\ \pi/5\ -1\ 1]^T$

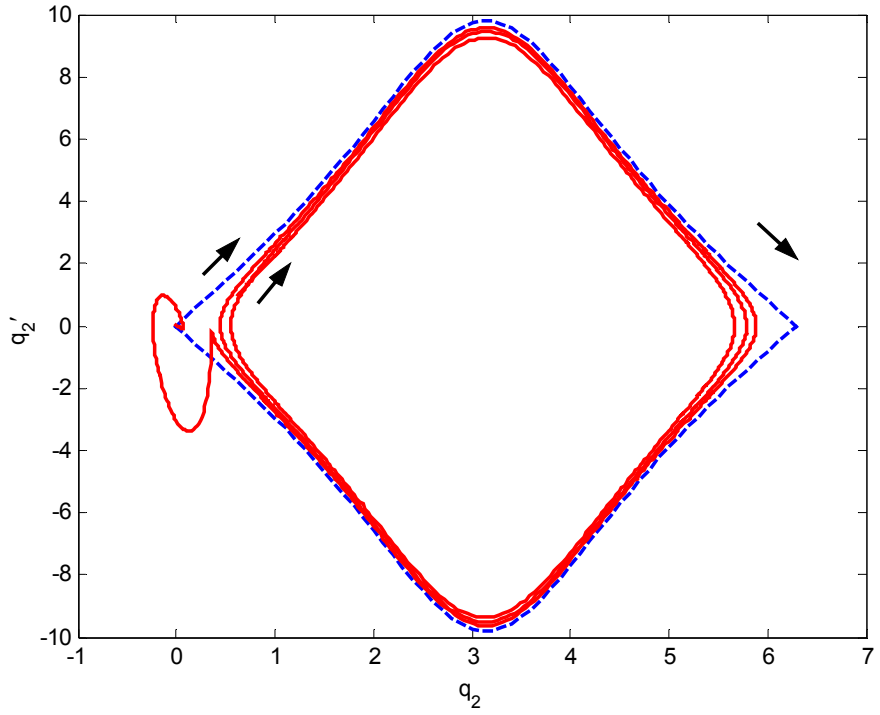

Figure 5.16 Homoclinic orbit convergence for $x_0 = [1\ \pi/5\ -1\ 1]^T$

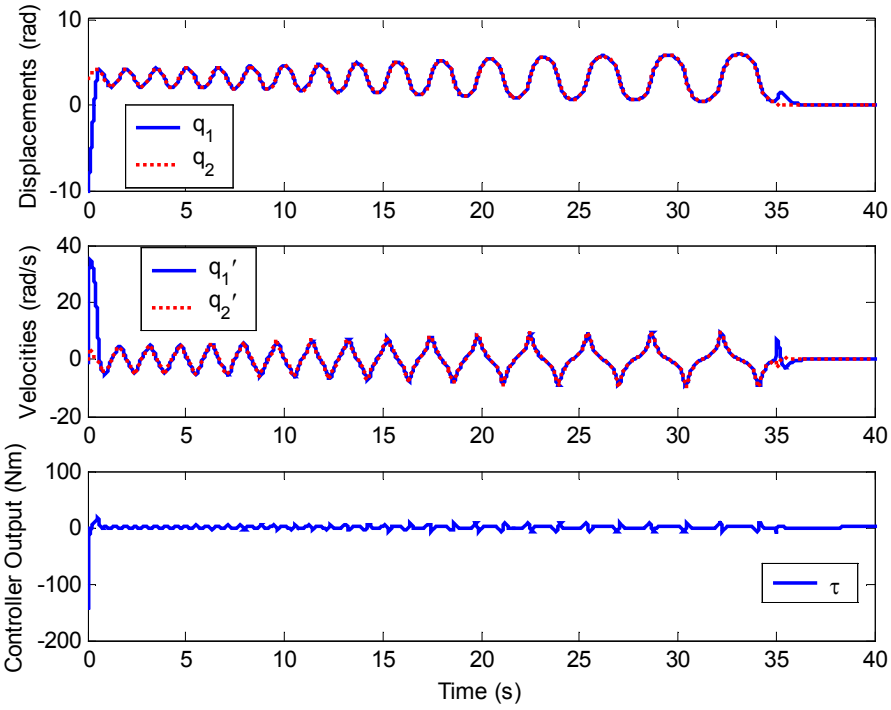

Figure 5.17 Passivity-Based Control of Underactuated DiscBot, $x_0 = [-10\ \pi\ -1\ 0]^T$

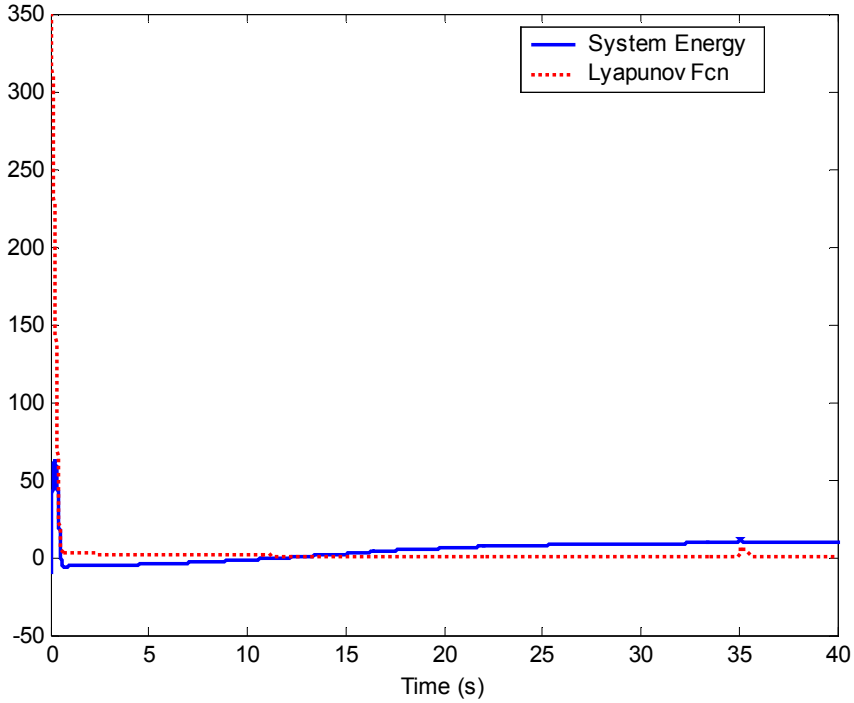

Figure 5.18 Lyapunov function time history for $x_0 = [-10\ \pi\ -1\ 0]^T$

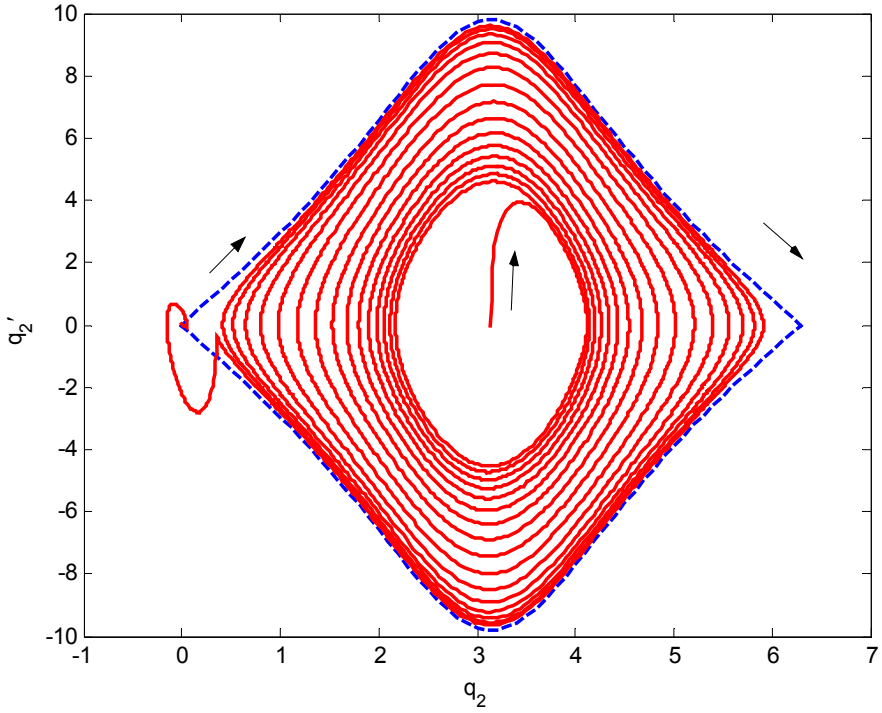

Figure 5.19 Homoclinic orbit convergence for $x_0 = [-10\ \pi\ -1\ 0]^T$

Summary

Passivity-Based control is an elegant means to utilize the DiscBot's natural dynamics to achieve semi-global stabilization. The controller's behavior is markedly different from both the LQR and the Feedback Linearization-based controllers. The fact that the PBC converges only to a homoclinic orbit and not a fixed point requires switching to another controller at some point in order to stabilize completely. Analysis of when to switch from one controller to another is not a well-studied topic [16].

5.2.4 GP-Based Control

Genetic Programming was applied to the underactuated DiscBot in two distinct ways. The first approach is the same as that used for the fully actuated DiscBot. The second approach utilizes a fitness function more compatible with Passivity-Based Control.

Approach 1

Several Genetic Programming runs were performed on the underactuated DiscBot. A representative run, with parameters given in Table 5.3, is described here.

Table 5.3 Underactuated DiscBot GP Parameters for Approach 1

Population Size	500
Max. Generations	100
Min. Tree Depth	4
Max. Tree Depth	11
P_r	0.10
P_c	0.45
P_m	0.45
Selection Method	Fitness-Proportionate
Simulation Time	4s
Integration Algorithm	Fixed time step 4^{th} order Runge-Kutta
Time Step	10ms

The fitness function used was

$$J = \int_{t_0}^{t_f} \frac{1}{2}\left[\mathbf{x}^T(t)\mathbf{Q}\mathbf{x}(t) + \boldsymbol{\tau}^t(t)\mathbf{R}\boldsymbol{\tau}(t)\right]dt .$$

An incremental approach was used for this GP run. Six separate stages were solved in succession. Early stages weigh pendulum displacement very highly, and disc displacement very lowly. Later stages shift the weights to emphasize disc displacement as well. Each stage uses a "Raw Fitness Threshold" to decide when to move on to the next stage, or to finish the run in the case of the last stage. The stages are summarized in Table 5.4.

Table 5.4 Underactuated DiscBot Incremental GP Parameters (Approach 1)

Stage	Initial State x_0 Set	Fitness Function Q, R Matrices	Raw Fitness Threshold
1	(0 π/8 0 0)*0.5 (1 0 0 0) *0.5 (1 -π/8 0 0) *0.5 (0 π/4 0 0) *0.5 (1 π/4 0 0) *0.5 (1 -π/4 0 0) *0.5	$\mathbf{Q} = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.99 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{R} = 0.0$	50
2	(0 π/8 0 0) (1 0 0 0) (1 -π/8 0 0) (0 π/4 0 0) (1 π/4 0 0) (1 -π/4 0 0)	$\mathbf{Q} = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.99 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{R} = 0.0$	50
3	(0 π/8 0 0)*1.5 (1 0 0 0)*1.5 (1 -π/8 0 0)*1.5 (0 π/4 0 0)*1.5 (1 π/4 0 0)*1.5 (1 -π/4 0 0)*1.5	$\mathbf{Q} = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.99 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{R} = 0.0$	50

4	(0 π/8 0 0)*2 (1 0 0 0) *2 (1 -π/8 0 0) *2 (0 π/4 0 0) *2 (1 π/4 0 0) *2 (1 -π/4 0 0) *2	$\mathbf{Q} = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.99 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{R} = 0.0$	50
5	(0 π/8 0 0)*2 (1 0 0 0) *2 (1 -π/8 0 0) *2 (0 π/4 0 0) *2 (1 π/4 0 0) *2 (1 -π/4 0 0) *2	$\mathbf{Q} = \begin{bmatrix} 0.20 & 0 & 0 & 0 \\ 0 & 0.80 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \mathbf{R} = 0.0$	100
6	(0 π/8 0 0)*2.25 (1 0 0 0) *2.25 (1 -π/8 0 0) *2.25 (0 π/4 0 0) *2.25 (1 π/4 0 0) *2.25 (1 -π/4 0 0) *2.25	$\mathbf{Q} = \begin{bmatrix} 0.20 & 0 & 0 & 0 \\ 0 & 0.80 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{R} = 0.0$	150

The progression of Best Raw Fitness and the fitness stages are shown in Figure 5.20. Note how the Raw Fitness decreases monotonically, except when the fitness stage increments, which indicates the search process is moving toward a minimum.

After 100 generations, this run produced the individual in Figure 5.21, with a raw fitness of 204. The performance of this individual, as simulated during the GP run, is shown in Figure 5.22. The DiscBot appears to be asymptotically stable, although it exhibits large oscillations.

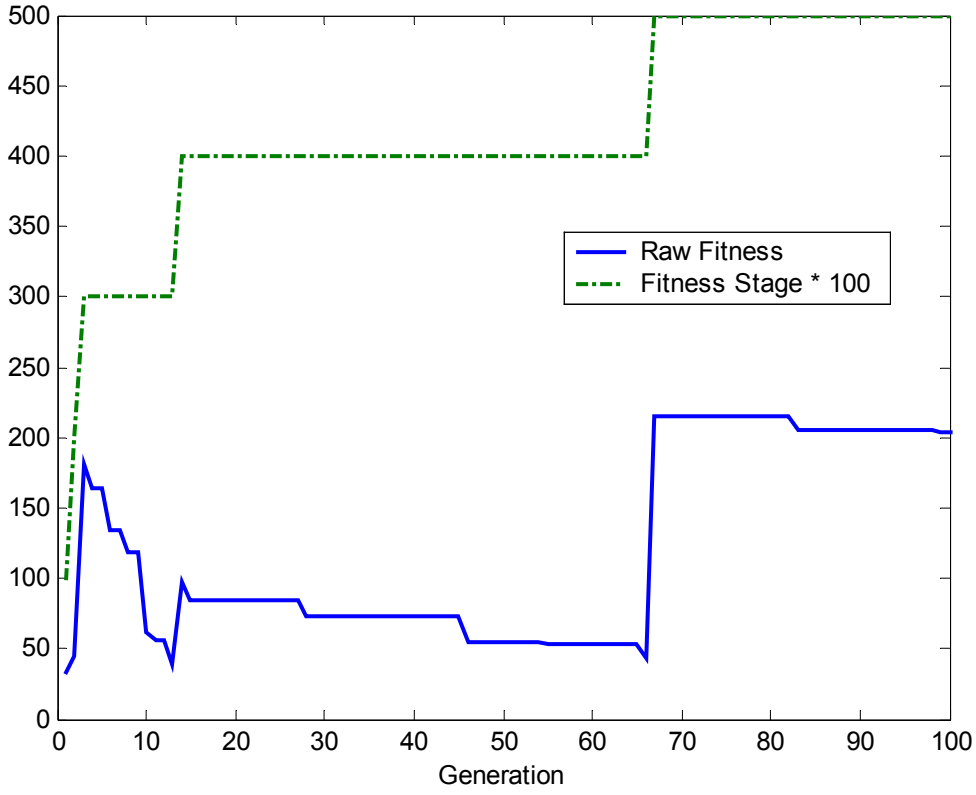

Figure 5.20 Underactuated DiscBot Raw Fitness History (Approach 1)

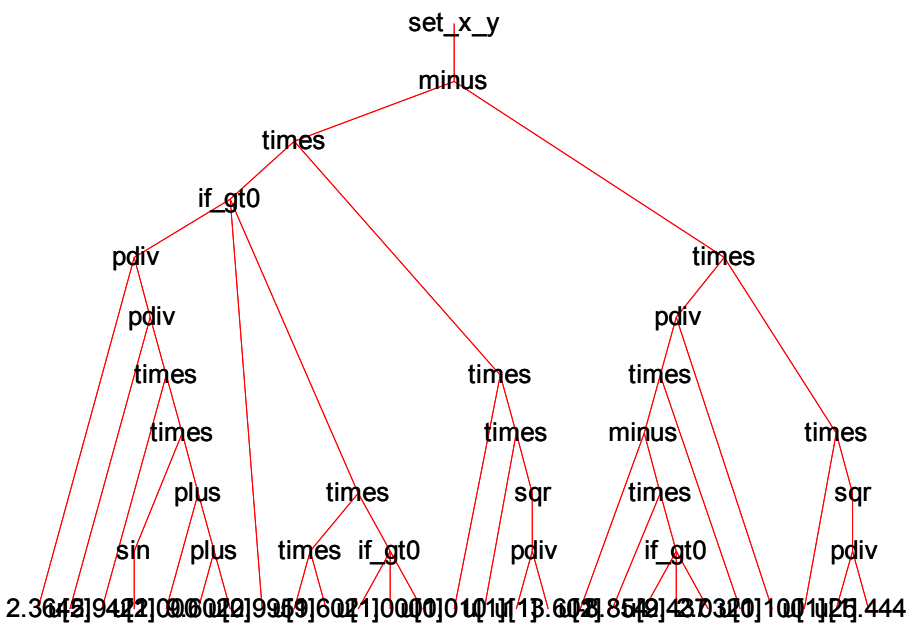

Figure 5.21 Best-of-Run Individual (Approach 1)

Closer inspection of this individual revealed that *reducing* the simulation step size caused the asymptotic stability to *disappear*. The pendulum appears to approach a limit cycle, and may possibly be stable in the Lyapunov sense. Several GP runs were performed at smaller and smaller step sizes, down to 0.5ms (the algorithm is a fourth order Runge-Kutta; step size here refers to major step size). All runs exhibited the same behavior: apparent asymptotic convergence that disappears when the step time is reduced after the GP run concludes. Figure 5.23 displays an example of the "disappeared" asymptotic stability.

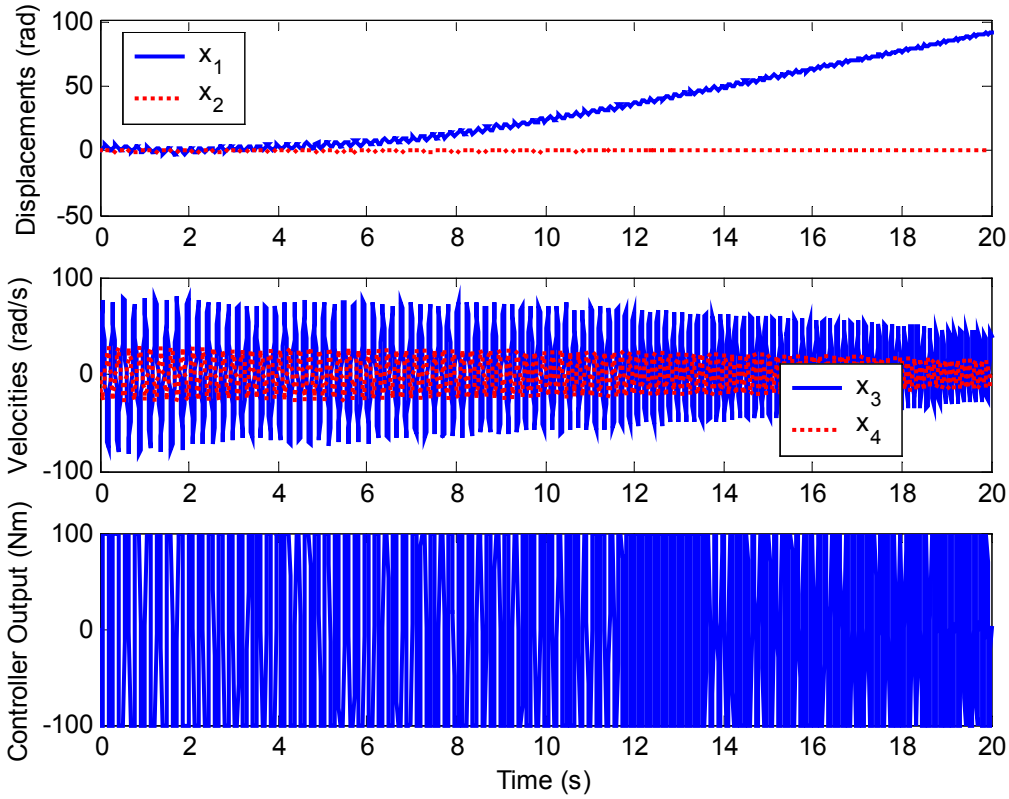

Figure 5.22 Best-of-Run Individual Performance, 10ms Step size (Approach 1)

This behavior is an example of a known problem with all evolutionary algorithms [5]. The algorithm, presented with the optimization problem within the simulation environment, cannot distinguish between the problem and the environment. If a small flaw or approximation exists in the environment, that flaw can be exploited just as easily as any feature of the optimization problem. In this case the exploited flaw is the integration algorithm itself. Inherent errors in the numerical integration process are exaggerated when large accelerations are integrated. A controller can create large accelerations by applying a large motor torque to the DiscBot. In such a case, energy is

no longer conserved, and can "leak" into or out of the simulated system through the integrator algorithm itself. Inspection of Figure 5.22 and Figure 5.23 reveals that the evolved controller does indeed apply large torques on the motor, enough to exploit the integration algorithm.

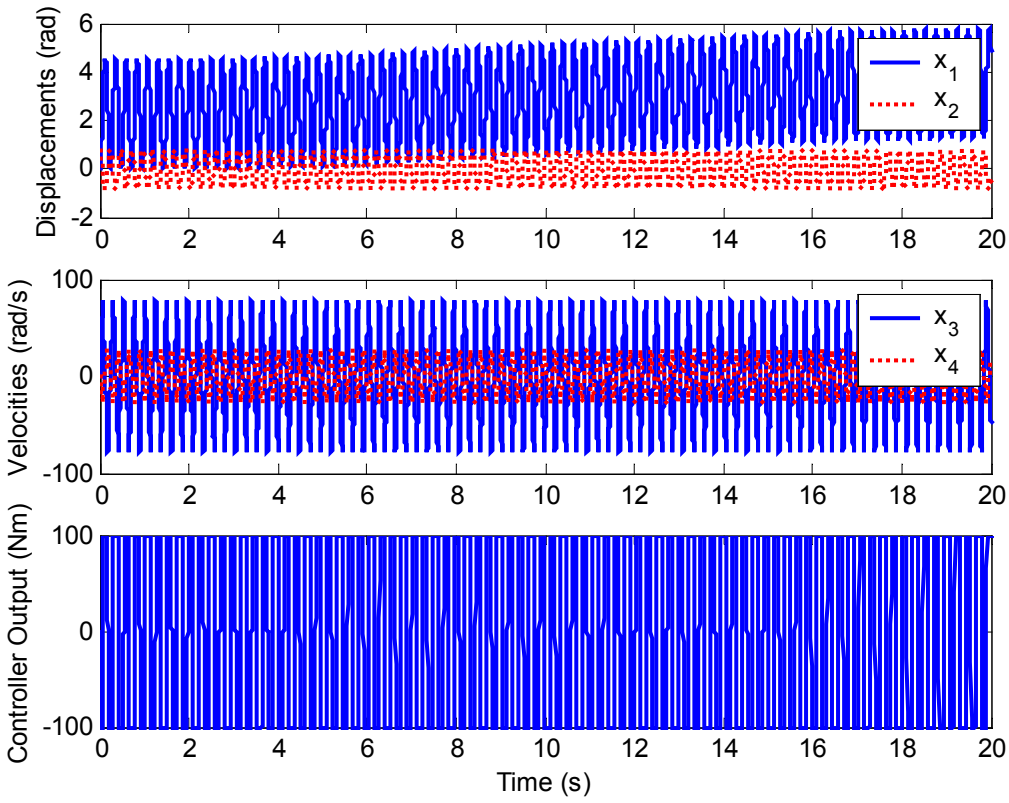

Figure 5.23 Best-of-Run Individual Performance, 2ms Step size (Approach 1)

Approach 2

Approach 2 is motivated by lessons learned from Approach 1 and the Passivity-Based Control results. When one considers the operation of the Passivity-Based Controller, it is clear that a fitness function that simply attempts to minimize state deviation from the origin is unlikely to succeed. The passive controller swings the pendulum back and forth, adding a little energy in each swing. It takes a fairly long time to approach the target state. In light of this, it is clear that the LQR-like fitness function employed in Approach 1 is unlikely to produce a successful result. In fact, any candidate that swings back and forth, rather than using brute force, will be ranked very low. This is evident, as the best performer from Approach 1 uses very large actuator torques to force rapid oscillation about the origin.

The fitness function used in this approach is derived from the Lyapunov function used in the Passivity-Based Controller:

$$J = \tfrac{1}{2}\int_0^{t_f} \left[k_E (E - E_d)^2 + k_v \dot{\tilde{q}}^2 + k_p \tilde{q}^2 \right] t dt \qquad (5.25)$$

where

$$\tilde{q} = q_2 - q_1 = \mathbf{Mq}, \qquad \dot{\tilde{q}} = \dot{q}_2 - \dot{q}_1 = \mathbf{M\dot{q}}$$

Note again that time is a factor in the integrand, to encourage selection of individuals which decrease the integrand over time. The fitness cases used were

$$x_0 = 2.25(0,\pi,0,0)^T$$
$$x_0 = 2.25(1,0,0,0)^T$$
$$x_0 = 2.25(1,-\tfrac{\pi}{8},0,0)^T$$
$$x_0 = 2.25(0,\tfrac{\pi}{4},0,0)^T$$
$$x_0 = 2.25(1,\tfrac{\pi}{4},0,0)^T$$
$$x_0 = 2.25(1,-\tfrac{\pi}{4},0,0)^T \quad .$$

Several Genetic Programming runs were performed on the underactuated DiscBot using (5.25) as the fitness function. A representative run is described here. DiscBot parameters for the run are given in Table 5.5.

Table 5.5 Underactuated DiscBot GP Parameters (Approach 2)

Population Size	200
Max. Generations	60
Min. Tree Depth	4
Max. Tree Depth	11
P_r	0.4
P_c	0.3
P_m	0.3
Selection Method	Fitness-Proportionate
Simulation Time	10s
Integration Algorithm	Fixed time step 4^{th} order Runge-Kutta
Time Step	25ms

Incremental stages were not used with Approach 2, as it is not obvious what the stages should be. Restricting initial conditions does not fit with the passivity approach,

and it is unclear what effect weighting different terms in the fitness function would have.

The progression of Best Raw Fitness is shown in Figure 5.24. After 60 generations, the best-of-run fitness was 229, achieved by the individual in Figure 5.25. The controller's performance is illustrated in Figure 5.26. The overall behavior of this controller is similar to the passive controller. The link velocities tend to converge together, and the pendulum swings back and forth rhythmically. The most obvious difference is that q_1 does not converge to zero, but appears to increase without bound.

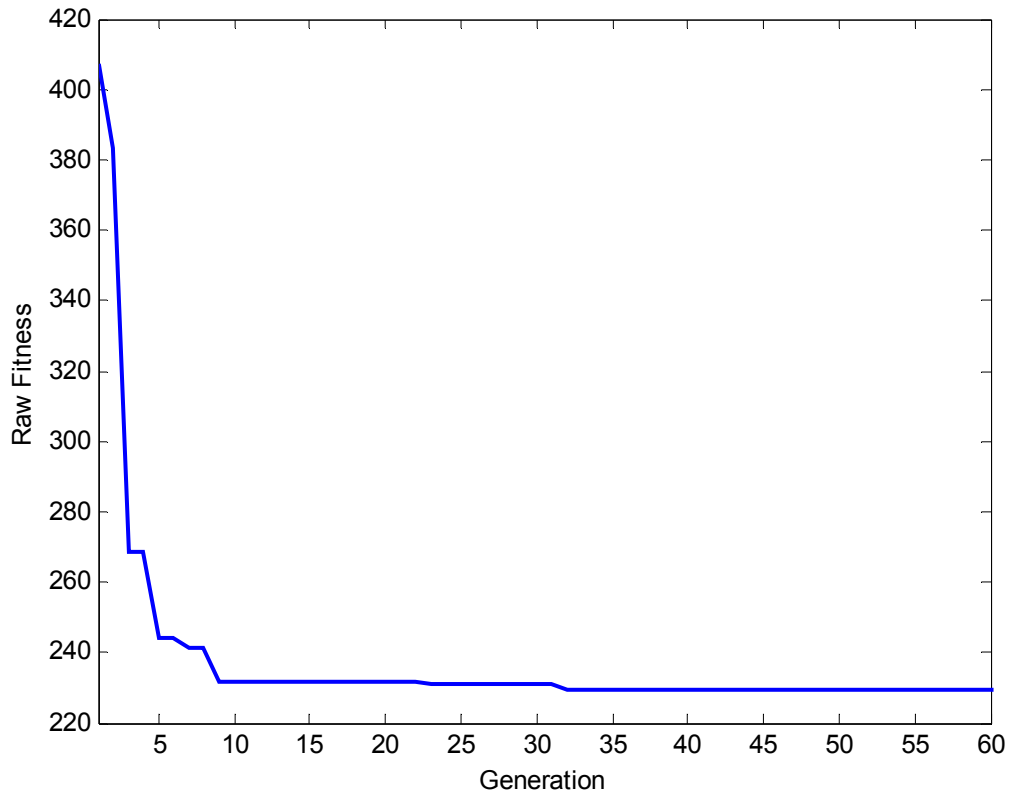

Figure 5.24 Underactuated DiscBot Best Raw Fitness History (Approach 2)

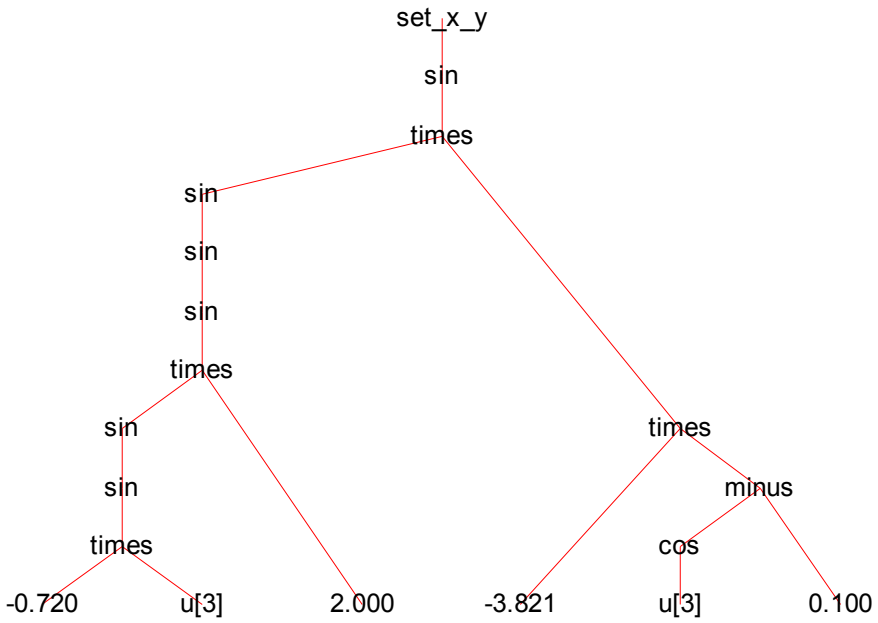

Figure 5.25 Best-of-Run Individual for Underactuated DiscBot (Approach 2)

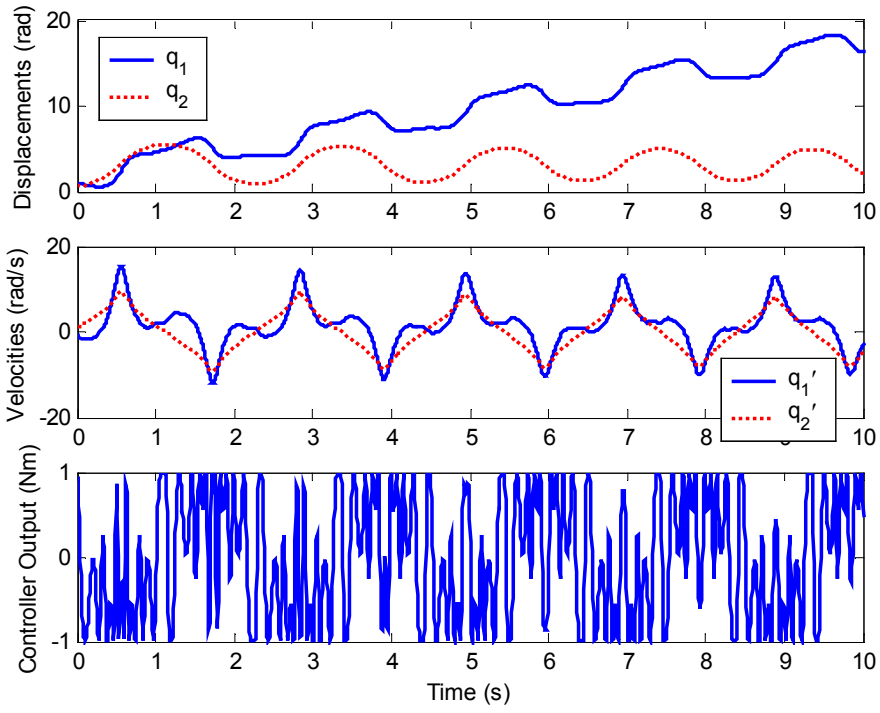

Figure 5.26 GP Control of Underactuated DiscBot, $x_0 = [1\ \pi/5\ -1\ 1]^T$ (Approach 2)

Further explorations of Approach 2 have been inconclusive because of the much greater computational requirement of calculating the energy term in the fitness function. Its inclusion doubles the time required for a GP run. In addition, passivity-type controllers tend to converge slowly, which imposes even greater computational requirements on the GP system. At this time, it appears infeasible to pursue Approach 2 further (all computations were performed on a 1.4GHz AMD Athlon system running Windows 2000 and MATLAB).

CHAPTER 6. CONCLUSIONS

Genetic Programming successfully found a controller for the fully actuated DiscBot. The evolved controller operates the joints independently, and does not achieve zero steady-state error.

The success of Genetic Programming on fully actuated plants is unsurprising, as it is known that an open kinematic chain can be controlled by independent joint controllers [17]. Such controllers are in general not optimal, but do assure stability.

In contrast, no stable controllers were found for the underactuated DiscBot. In fact, it appears easier to exploit the simulation environment than find an underactuated DiscBot controller. Experience gained while working with the GP system leads to the following observations concerning the lack of a satisfactory evolved controller.

In the DiscBot, steady-state pendulum error must converge to zero or the state will grow without bound or possibly approach a limit cycle. No steady-state solution exists without the pendulum being either straight up or straight down. This strict requirement for pendulum control represents a very difficult problem for evolutionary algorithms, which tend to produce approximate solutions that almost never solve the problem exactly. The number of correct programs in the search space is therefore much smaller than in the fully actuated case.

Stabilization of the underactuated DiscBot is best achieved via an indirect approach; i.e. passive control. The passive controller slowly builds energy toward the target level, without applying large torques. Passive control alone is not adequate to

stabilize the DiscBot, however. At some point control must be switched to a locally exponentially stable controller, because the passive controller will never achieve steady state. It is unclear how to best represent these goals in a fitness function.

Genetic Programming does seem appropriate, in some cases, as a method to generate realistic motion for animated characters. In these cases, the control requirements are very specific to the desired motion and need not generalize at all. The fitness functions may also be "tuned" by individuals with little control systems knowledge to achieve desired behavior.

Genetic Programming seems unsuitable for use for controls development for several reasons. First, stability properties are effectively impossible to analyze given the typical complexity of the evolved solutions. Second, solutions that only approximately converge to the desired state appear. Finally, no conclusive data exists to indicate Genetic Programming can reliably stabilize underactuated systems.

BIBLIOGRAPHY

[1] Holland, J.H. "Adaptation in Natural and Artificial Systems," University of Michigan Press, Ann Arbor, 1975

[2] Howley, B. "Genetic Programming and Parametric Sensitivity: A Case Study in Dynamic Control of a Two Link Manipulator," *Proc. Genetic Programming 1997 Conference*, Vol., No., pp. 180-185

[3] Howley, B. "Genetic Programming of Near-Minimum-Time Spacecraft Attitude Maneuvers," *Proc. Genetic Programming 1996 Conference*, Vol. , No., pp.98-106

[4] Gritz, L. and Hahn, J.K. "Genetic Programming for Articulated Figure Motion," *Journal of Visualization and Computer Animation*, 1995

[5] Sims, K. "Evolving Virtual Creatures," *Computer Graphics*, (SIGGRAPH 1994 Proc.), July 1994, pp.15-22

[6] Spong, M.W. "Underactuated Mechanical Systems," *Lecture Notes in Control and Information Sciences* 230, Springer-Verlag, London, 1997

[7] Shabana, Ahmed A. *Computational Dynamics.* John Wiley & Sons, 1994

[8] Slotine, Jean-Jacques E. and Li, Weiping. *Applied Nonlinear Control*. Prentice-Hall, 1991

[9] Kirk, Donald E. *Optimal Control Theory.* Prentice-Hall, Englewood Cliffs, NJ., 1970

[10] Isidori, A. *Nonlinear Control Systems*. Springer-Verlag, 1989.

[11] Kolavennu, Soumitri; Palanki, Srinivas, and Cockburn, Juan C. "Input-Output Linearization for Non-Square MIMO Systems," *American Institute of Chemical Engineers National Meeting,* 1997

[12] Fantoni, Isabelle and Lozano, Rogelio. *Non-linear Control for Underactuated Mechanical Systems*. Springer-Verlag. 2002

[13] Koza, John R. *Genetic Programming*. MIT Press, Cambridge, MA., 1992

[14] Gomez, Faustino and Miikkulainen, Risto. "Incremental Evolution of Complex General Behavior," Adaptive *Behavior*, 5:317-342, 1997

[15] Ogata, Katsuhiko. *Modern Control Engineering,* Second Edition. Prentice-Hall, 1990

[16] Spong, M.W. "Partial Feedback Linearization of Underactuated Mechanical Systems", *IROS'94*, Munich, Germany, pp. 314-321, Sept. 1994.

[17] Spong, M.W. and Vidyasagar, M. *Robot Dynamics and Control*. John Wiley & Sons, 1989