ANALYSIS AND REAL-TIME IMPLEMENTATION OF STATE-DEPENDENT RICCATI EQUATION CONTROLLED SYSTEMS

BY

EVRIN BILGE ERDEM

B.S., Middle East Technical University, 1994
M.S., Middle East Technical University, 1996

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2001

Urbana, Illinois

Abstract

This study deals with one of the recently proposed methods for control of nonlinear dynamical systems, the State-Dependent Riccati Equation (SDRE) method. This method can be thought of as the nonlinear counterpart of LQR based control design, with the matrices $A(x)$, $B(x)$, $Q(x)$ and $R(x)$ being functions of the states. The greatest advantage offered by SDRE is the design flexibility of tuning the state and input penalty matrices $Q(x)$ and $R(x)$, respectively, as functions of states. This flexibility softens the classical state-error vs. input tradeoff seen in classical LQR control.

Of the many open issues regarding SDRE, two are dealt with in this thesis: stability and real-time control implementation.

The closed-loop system equations of an SDRE controlled system are usually not known explicitly, which makes stability analysis quite difficult. The SDRE controlled closed-loop system is known to be locally asymptotically stable, but how large this region of stability is is not known. In this work, analytical conditions are found to globally asymptotically stabilize second order nonlinear systems affine in the input. Comparison with feedback linearization is made, which reveals the advantages of SDRE. The theoretical results are demonstrated experimentally on a magnetically levitated ball tracking a square wave in real-time.

A method for estimating the region of stability of SDRE controlled systems based on vector norms and overvaluing systems is also developed. This method requires knowledge of only the maximum and the minimum values of the feedback gains over a chosen domain in the state-space. Also, by this method, higher order systems can be reduced to tractable lower order ones at the expense of more conservative estimates.

Finally, experimental real-time control using SDRE is addressed. The plant used is the Pendubot, a two-link underactuated robot, which is a highly nonlinear 4^{th} order system. The SDRE control is calculated by solving the Algebraic Riccati Equation online and the Pendubot is regulated at one of its unstable positions. The results show that the computational power requirement associated with SDRE real-time control is not very high. Simulation and experimental results reveal the advantages of SDRE control over LQR.

To My Father

Acknowledgements

First and foremost, I would like to thank my research advisor, Professor Andrew Alleyne, for his valuable guidance, patience, financial support and friendliness during my doctoral study.

I thank my parents Nur□en Erdem and Ali □nal Erdem for their ever-present support and encouragement during my studies.

I am grateful to Mr. Dan Block for letting me perform experiments on the Pendubot and his helpfulness and patience. Thanks are also due to fellow graduate student and friend Mr. Alper □ngör, for his help in modifying the controller C-code to run in real-time. I also appreciate the help of the talented photographer, fellow graduate student and friend Mr. Danian Zheng for his help in videotaping and photographing the experiments.

Finally, I am grateful for having known all the people who have impacted my life in some way during the last five years, either positively or negatively and some more than others, all in all making my Champaign-Urbana experience a valuable one and helping me grow.

Table of Contents

Chapter 1

Introduction

1.1 Scope of This Study

The desire to improve performance in controlled systems leads to more and more precise modeling. However, if a model is a good representation of a system over a wide operating range, it is most often nonlinear. Consequently, the commonly used tools for linear control design become insufficient and it becomes necessary to use other methods.

As opposed to the control theory of linear systems, the theory for control of nonlinear systems is far from unified. There exist numerous nonlinear control design techniques, each with its advantages and drawbacks. Most of them are limited in their range of applicability, and use of one particular nonlinear control method for a specific system usually demands trade-off between performance/robustness/optimality/ (computational) cost. Some of the well-known nonlinear control methods are feedback linearization, adaptive control, gain scheduling, nonlinear predictive control and sliding mode control.

One of the recently proposed nonlinear control methods is the State-Dependent Riccati Equation (SDRE) method. It can be thought of as the nonlinear counterpart of LQR based control design, with the only difference being that the system matrices $A(x)$, $B(x)$ and the penalty matrices $Q(x)$ and $R(x)$ being functions of the states. Thus, the solution of the (state-dependent) Algebraic Riccati Equation $P(x)$ is also state dependent, as is the feedback control $u(x)$. The greatest advantage offered by SDRE is the design flexibility of tuning the state and input penalty matrices $Q(x)$ and $R(x)$, respectively, as functions of states. This flexibility gives the designer more intuitive "control" over the behavior of the controlled system as well as softens the classical state-error vs. input tradeoff in classical LQR control. For example, let the weighting matrices be chosen so that $Q(x)$ increases and $R(x)$ decreases as x increases. This would lead to saving on control effort near the origin. Far from the origin, this choice would make sure that the system is driven to its equilibrium.

Since its recognition less than a decade ago, SDRE has been shown to work well for nonlinear regulation, observer designs, parameter estimation, etc. However, almost all of the

results are simulation-based. There exists a lack of theoretical and experimental results regarding this control method. Results that will establish a sound theoretical basis and free SDRE control from its "ad-hoc" impression are seriously called for.

1.2 Contributions of This Research

The goal of this study is to help fill some of the many voids in the foundation of SDRE control. Two main issues are attacked for this purpose:

1) Stability of the closed-loop system; i.e. whether it is globally asymptotically stable and if not, how large the region of attraction is,
2) Realization and evaluation of the feasibility of SDRE control of a physical system in real-time.

The SDRE controlled closed-loop system is known to be locally asymptotically stable. Global stability, as opposed to linear systems, is more difficult to prove because having stable eigenvalues in the whole state space does not guarantee global asymptotic stability. In this work, analytical conditions are found to globally asymptotically stabilize second order nonlinear systems affine in the input. These conditions involve tuning of the penalty matrices $Q(x)$ and $R(x)$ as certain functions of the states. The theoretical results are demonstrated experimentally on a magnetically levitated ball tracking a square wave in real-time by SDRE control. Comparison of SDRE with feedback linearization for this class of systems is also made.

The closed-loop system equations of an SDRE controlled system are usually not known explicitly. This makes stability analysis quite difficult, since the straightforward Lyapunov approaches cannot be used. Resorting to time-domain simulations, on the other hand, would be too cumbersome and costly. In this study, a method for estimating the region of stability of SDRE controlled systems based on vector norms is developed based on the work done in [2]. An overvaluing matrix M for the closed-loop coefficient matrix $A_{CL}(x)$ is found such that $\dot{z} = Mz$ becomes a comparison system for the original system, $\dot{x} = A_{CL}(x)x$. The stability region of the comparison system is an estimate of that of the original nonlinear

system. This method requires knowledge of only the maximum and the minimum values of the feedback gains over a chosen domain in the state-space. Also, by this method, higher order systems can be reduced to tractable lower order ones, possibly at the expense of more conservative estimates.

Finally, experimental real-time control using SDRE is addressed in this thesis. The plant used for this experiment is a two-link underactuated robot, namely the Pendubot [1], which is a highly nonlinear 4^{th} order system. The SDRE control is calculated by solving the Algebraic Riccati Equation online at each time step and the Pendubot is regulated at one of its unstable positions. Simulation results with LQR and SDRE control are also included. Also, an attempt to estimate the region of attraction is presented.

The remainder of this thesis is organized as follows. In Chapter 2, a review of some existing nonlinear control methods, namely feedback linearization, adaptive control, gain scheduling, nonlinear model predictive control and sliding mode control is given. In Chapter 3, the SDRE formulation and a literature survey of SDRE research is presented. In Chapter 4 theoretical conditions are derived analytically for global asymptotic stabilization of second order systems. The found conditions are demonstrated experimentally on a magnetically levitated ball system. In Chapter 5, a method for estimating the stability regions of SDRE controlled systems is developed and presented with examples. Realization and results of experimental real-time control of the Pendubot using SDRE is presented in Chapter 6. Conclusion and comments are given in Chapter 7.

The C-code for the S-function used to solve the SDRE gains online for the Pendubot experiment in Chapter 6 is given in Appendix A. In Appendix B, an approach for solving the SDRE gains online using existing dSPACE data transfer modules to and from the MATLAB environment, namely MLIB and MTRACE, is presented.

1.3 References

[1] D.J. Block, *Mechanical Design and Control of the Pendubot*, M.S. Thesis, University of Illinois at Urbana-Champaign, 1991.

[2] P. Borne, J.P. Richard, N.E. Radhy, "Stability, stabilization, regulation using vector norms", *Chapter 2 in Nonlinear Systems Vol.2: Stability and Stabilization,* A.J. Fossard and D. Normand-Cyrot (editors), Chapman & Hall, 1996.

[3] A.J. Fossard and D. Normand-Cyrot, *Nonlinear Systems Volume 3: Control*, Chapman & Hall and Masson, 1997.

Chapter 2

Review of Common Nonlinear Control Methods

Over the past several years, the inability of linear control design techniques to handle strongly nonlinear systems satisfactorily has led to a push in the development of nonlinear controller synthesis theory and techniques. Many methods have been proposed but there still is a lack of unified methodology for the control of nonlinear systems. Each of the proposed methods is at a different level of maturity, has its own set of systems to which it can be applied, and different performance and robustness properties which usually demand some tradeoff. Unlike in linear systems control, even the most mature of these nonlinear control methods lacks the ability to address, to a satisfying extent, the performance/ robustness/applicability properties desired from a controller for a wide range of dynamical systems. The most common of these methods are discussed in this chapter. A comparison is also given at the end of the chapter.

2.1 Feedback Linearization

The central idea behind feedback linearization [3, 9] is to algebraically transform a nonlinear system dynamics into a (fully or partly) linear one so that linear control techniques can be applied. As opposed to conventional linearization (i.e., Jacobian linearization) feedback linearization is achieved by exact state transformations and feedback, rather than by linear approximation of the dynamics.

In its simplest form, feedback linearization amounts to canceling nonlinearities in a nonlinear system so that the closed-loop dynamics is in a linear form. The method is simply applicable to systems in controllability canonical form (CCF). Even if the equations are not in CCF, they can be transformed to that form by using algebraic transformations. Another option in such a case would be to use partial linearization of the original dynamics, instead of full linearization.

There are two main ways of doing feedback linearization. The first one is *input-state linearization* where the full state equations are linearized. The second one is *input-output*

linearization where the input-output map from the control input u to the output y is linearized, even if the state equations are only partially linearized.

2.1.1 Input-state linearization

For a nonlinear system of the form

$$\dot{x} = f(x,u) \tag{2.1}$$

input-state linearization is done in two steps. First, a state transformation $z = z(x)$ and an input transformation $u=u(x,v)$ is found so that the nonlinear system dynamics is transformed into an equivalent dynamics in the form

$$\dot{z} = Az + Bv \tag{2.2}$$

Then v is designed by using standard linear control techniques, such as pole placement.

Example 2.1: Consider the system [9]

$$\begin{aligned} \dot{x}_1 &= -2x_1 + ax_2 + \sin x_1 \\ \dot{x}_2 &= -x_2 \cos x_1 + u\cos(2x_1) \end{aligned} \tag{2.3}$$

A difficulty is the nonlinearity in the first equation, which cannot be directly canceled by the control input u. However, if (2.3) is rewritten in the new set of coordinates

$$\begin{aligned} z_1 &= x_1 \\ z_2 &= ax_2 + \sin x_1 \end{aligned} \tag{2.4}$$

then the new state equations become

$$\begin{aligned} \dot{z}_1 &= -2z_1 + z_2 \\ \dot{z}_2 &= -2z_1 \cos z_1 + \cos z_1 \sin z_1 + au\cos(2z_1) \end{aligned} \tag{2.5}$$

Which have an equilibrium point at (0,0), just like the original equations (2.3). Now the nonlinearities in (2.5) can be canceled by a control law of the form

$$u = \frac{1}{a\cos(2z_1)}(v - \cos z_1 \sin z_1 + 2z_1 \cos z_1) \tag{2.6}$$

where v is an equivalent input to be designed, leading to a linear input-state relation

$$\begin{aligned} \dot{z}_1 &= -2z_1 + z_2 \\ \dot{z}_2 &= v \end{aligned} \tag{2.7}$$

Since the new dynamics is linear and controllable, the linear state feedback control law such as

$$v = -k_1 z_1 - k_2 z_2 \tag{2.8}$$

can place the poles anywhere with proper choices of feedback gains. For example, the choice $v = -2z_2$ results in the closed–loop dynamics

$$\begin{aligned} \dot{z}_1 &= -2z_1 + z_2 \\ \dot{z}_2 &= -2z_2 \end{aligned} \tag{2.9}$$

where both poles are placed at –2. In terms of the states x_1 and x_2 this control law corresponds to the original input

$$u = \frac{1}{\cos(2x_1)}(-2ax_2 - 2\sin x_1 - \cos x_1 \sin x_1 + 2x_1 \cos x_1) \tag{2.10}$$

The original states (x_1, x_2) is given from (z_1, z_2) by

$$\begin{aligned} x_1 &= z_1 \\ x_2 &= (z_2 - \sin z_1)/a \end{aligned} \tag{2.11}$$

Since both z_1 and z_2 converge to zero, the original states x_1 and x_2 converge to zero.

A *diffeomorphism* is a continuously differentiable map with a continuously differentiable inverse. For the mapping T in $z=T(x)$ to be a diffeomorphism, it must be invertible, i.e., $x = T^{-1}(z)$ must exist for all $z \in T(D_x)$.

Definition 2.1: *Input-state linearizability* [3]

A nonlinear system

$$\dot{x} = f(x) + g(x)u \tag{2.12}$$

where $f : D_x \to R^n$ and $g : D_x \to R^{nxp}$ are sufficiently smooth on a domain $D_x \subset R^n$ is input-state linearizable if there exists a domain D_z and a mapping T such that $D_z = T(D_x)$ contains the origin and the choice of variables $z=T(x)$ transforms the system (2.12) into the form

$$\dot{z} = Az + B\beta^{-1}(x)[u - \alpha(x)] \tag{2.13}$$

with (A, B) controllable and $\beta(x)$ nonsingular for all $x \in D_x$.

When a system is input-state linearizable, the map $z=T(x)$ that satisfies (2.13) is not unique. Guidelines to find proper linearizations for systems that are input-state linearizable is by itself an issue and details about this subject can be found in [3, 9].

In general, the system model is relied on for both the controller design and the computation of z. If there is uncertainty in the model, this will cause error in the computation of both the new state z and the control input u. This can lead to instability in many cases, meaning that feedback linearization can be very sensitive to model error.

2.1.2 Input-Output Linearization

Linearizing the state equations as in input-state linearization does not necessarily linearize the output equation. The idea in input-output linearization is to obtain a simple and direct relationship between the system output y and input u.

If one needs to differentiate the output of a nonlinear system r times to obtain an explicit relationship between u and y, the system is said to have a relative degree r. This is consistent with the notion of relative degree in linear systems (excess of poles over zeros). If the relative degree is less than the degree of state equations, i.e., $r < n$, then a part of the system dynamics called the "internal dynamics" has been rendered unobservable in the input-output linearization; and the system is not input-state linearizable but input-output linearizable. When $r = n$, there is no internal dynamics; and the system is both input-state and input-output linearizable. In order for the closed-loop system to be stable, the internal dynamics must be stable.

It is possible for one choice of output to yield a stable (or no) internal dynamics while another choice would lead to an unstable one. Therefore, if possible, one should choose the output y ("designer output") such that the internal dynamics is stable, without restricting y to be a physically meaningful quantity.

For linear systems the stability of the internal dynamics is simply determined by the location of zeros. This can be extended to nonlinear systems, by defining the so-called zero-dynamics for a nonlinear system. *The zero-dynamics is defined to be the internal dynamics of the system when the system output is kept at zero by the input.* Asymptotic stability of the zero-dynamics is enough to guarantee local asymptotic stability of the internal dynamics. However, no results on the global stability or even large range stability can be drawn for internal dynamics of nonlinear systems. In fact, only local stability is guaranteed for the internal dynamics even if the zero-dynamics is globally exponentially stable.

The procedure of input-output linearization can be summarized in three steps:

1) Differentiate the equation for output y until the input u appears.
2) Choose input u to cancel nonlinearities and guarantee tracking convergence.
3) Make sure the internal dynamics is stable.

The foregoing concepts are illustrated in the following example from [9].

Example 2.2: Consider the nonlinear system

$$\begin{aligned} \dot{x}_1 &= x_1^2 x_2 \\ \dot{x}_2 &= 3x_2 + u \end{aligned} \tag{2.14}$$

The system's linearization at $x = 0$ is

$$\begin{aligned} \dot{x}_1 &= 0 \\ \dot{x}_2 &= 3x_2 + u \end{aligned} \tag{2.15}$$

and thus has an uncontrollable mode corresponding to a pure integrator. Let the output function be defined as

$$y = -2x_1 - x_2 \tag{2.16}$$

Corresponding to this output, the relative degree of the system is 1, because

$$\dot{y} = -2\dot{x}_1 - \dot{x}_2 = -2x_1^2 x_2 - 3x_2 - u \tag{2.17}$$

The associated zero-dynamics (obtained by setting $y = 0$) is simply

$$\dot{x}_1 = -2x_1^3 \tag{2.18}$$

and thus is asymptotically stable. Therefore, the control law

$$u = -2x_1^2 x_2 - 4x_2 - 2x_1 \tag{2.19}$$

locally asymptotically stabilizes the system. ♦

Although conceptually simple, feedback linearization has some important limitations. First of all, it cannot be used for all nonlinear systems. If the zero-dynamics are unstable,

feedback linearization cannot be used. Both input-state and input-output linearization techniques impose somewhat stringent conditions, with those of the former being more stringent than those of the latter.

Secondly, the full state has to be measured. For nonmeasurable states, convergent observers are needed, which may be difficult to find for nonlinear systems.

Thirdly and most importantly, no robustness is guaranteed in the presence of parameter uncertainty or unmodeled dynamics. This problem is due to the fact that the exact model of the nonlinear system is not available in performing feedback linearization. The sensitivity to modeling errors may be particularly severe when the linearizing transformation is poorly conditioned.

Even if the model is known to a reasonably good accuracy, feedback linearization removes all nonlinearities, including the beneficial ones. Moreover, the control input needed to cancel some nonlinearities may be exceedingly large for physical actuator limits.

2.2 Adaptive Control

In some control tasks, the systems to be controlled have parameter uncertainty at the beginning of the control operation. Unless such parameter uncertainty is gradually reduced on-line by an adaptation or estimation mechanism, it may cause inaccuracy or instability for the control systems.

An adaptive controller [1, 3, 9] is a controller with adjustable parameters and a mechanism for adjusting the parameters. Even if the plant is linear, the controller becomes nonlinear because of the parameter adjustment mechanism. An adaptive control system can be thought of as having two loops. One loop is a normal feedback with the process and the controller. The other loop is the parameter adjustment loop. Usually, the control loop is slower that the parameter adjustment loop.

There exists relatively little general theory for the adaptive control of nonlinear systems, as opposed to linear systems. However, for some important classes of nonlinear control problems such as in robotics, adaptive control has been successfully developed. Such problems usually satisfy the following conditions:

1) The nonlinear plant dynamics can be linearly parametrized.
2) The full state is measurable.

3) Nonlinearities can be canceled stably (i.e., without unstable hidden modes or dynamics) by the control input if the parameters are known.

There are two main approaches for constructing adaptive controllers. One is the so-called model reference adaptive control method and the other is the so-called self-tuning method.

2.2.1 Model Reference Adaptive Control (MRAC)

This method is used to solve a control problem where the performance specifications are given in terms of a reference model for which the input/output relationship is known. This model tells how the process output ideally should respond to the command signal.

A MRAC system can be schematically represented as in Figure 2.1. It is composed of four parts: a plant containing unknown parameters, a reference model for compactly specifying the desired output of the control system, a feedback control law containing adjustable parameters, and an adaptation mechanism for updating the parameters.

The plant is assumed to have a known structure, although the parameters are unknown. For nonlinear plants, this implies that the structure of the dynamic equations is known, but that some parameters are not.

The reference model provides the ideal plant response which the adaptation mechanism should seek in adjusting the parameters. There are some inherent constraints on the structure of the reference model to be chosen (e.g. its order and relative degree) given the assumed structure of the plant model.

The controller is usually parametrized by a number of adjustable parameters. The controller should have perfect tracking capacity in order to allow the possibility of tracking convergence. This means that when the plant parameters are exactly known, the corresponding controller parameters should make the plant output identical to that of the reference model.

The adaptation mechanism is used to adjust the parameters in the control law. The main issue here is to synthesize an adaptation mechanism which will guarantee that the control system remains stable and the tracking error converges to zero as the parameters are varied.

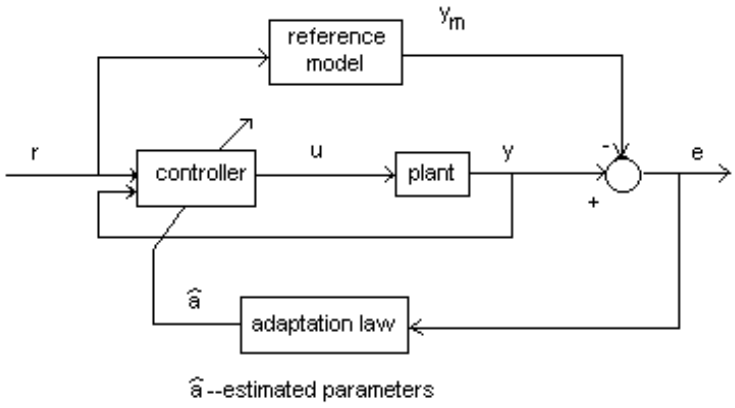


Figure 2.1 A model-reference adaptive control system

2.2.2 Self-Tuning Controllers (STC)

As opposed to MRAC, in the STC problem, the process parameters are unknown so they are replaced by their estimates, as provided by a parameter estimator. Parameter estimation can be described simply as the process of finding a set of parameters that fits the available input-output data from a plant. This is different from parameter adaptation in MRAC systems, where the parameters are adjusted so that the tracking error converges to zero. The *estimates* of the process parameters are updated and the controller parameters are obtained from the solution of a design problem using the estimated parameters.

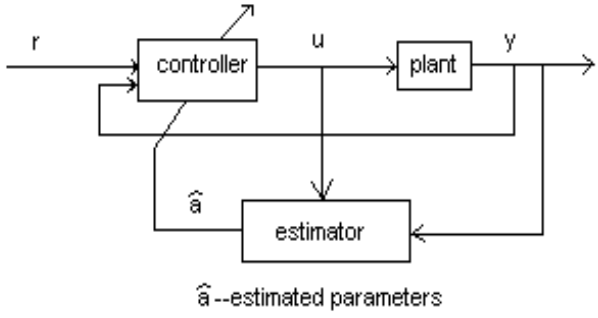


Figure 2.2 A self-tuning controller

The schematic structure of a STC is shown in Figure 2.2. At each time step the estimator sends to the controller a set of estimated plant parameters, $\hat{a}$, which is computed based on the past plant input u and the output y. The computer finds the corresponding controller parameters and the computes a control input u based on the controller parameters

and measured signals. This control input *u* causes a new plant output to be generated, and the whole cycle of parameter and input updates is repeated. The real-time estimates of the process parameters are used as if they were equal to the true parameters (i.e., the uncertainties of the estimates is not considered). This is known as the *certainty equivalence principle* [9].

Example 2.3 This example from [9] is the basic algorithm for adaptive control of a robotic system whose equations of motion in the joint coordinates *q* is given by

$$\ddot{q} = H^{-1}(\tau - C(\dot{q},q)\dot{q} + g(q)) \tag{2.20}$$

where $q \in R^n$ is the position vector, $H \in R^{nxn}$ is the inertia matrix, $C \in R^{nxn}$ is the matrix of nonlinear motion components such as Coriolis accelerations and $g \in R^n$ is the vector of gravitational forces. τ is the *p*-dimensional vector of actuator inputs, where *p=n* unless the manipulator is underactuated (*p<n*) or redundant (*p>n*).

Let the desired trajectory be $q_d(t)$. In the presence of some unknown parameters, the adaptive controller design problem is to derive a control law for the actuator torques, and an estimation law for the unknown parameters such that the manipulator output positions *q*(*t*) closely tracks the desired trajectory. With *a* being a constant vector of unknown parameters describing the mass properties and $\hat{a}$ its estimate, the parameter estimation error is defined as $\tilde{a} = \hat{a} - a$. Let the surface *s* where *s=0* corresponds to zero error, be defined (as in sliding control section) as

$$s = \dot{\tilde{q}} - \Lambda\tilde{q} = \dot{q} - \dot{q}_r \tag{2.21}$$

where Λ is a symmetric matrix such that -Λ is Hurwitz and the "reference velocity" vector $\dot{q}_r$ is formed by shifting the desired velocities $\dot{q}_d$ according to the position error $\tilde{q}$.

Consider the Lyapunov function candidate

$$V(t) = \frac{1}{2}[s^T Hs + \tilde{a}^T \Gamma^{-1}\tilde{a}] \tag{2.22}$$

where Γ is a symmetric positive definite matrix. Differentiating (2.22) and using (2.21) yields

$$\dot{V}(t) = s^T(\tau - H\ddot{q}_r - C\dot{q} + g) + \dot{\hat{a}}^T\Gamma^{-1}\tilde{a} \tag{2.23}$$

Since the terms $H(q)$, $C(\dot{q},q)$ and $g(q)$ all depend *linearly* on the unknown mass properties a, a known matrix $Y = Y(\dot{q},q,\ddot{q}_r,\dot{q}_r)$ can be defined such that

$$H(q)\ddot{q}_r + C(\dot{q},q)\dot{q}_r + g(q) = Y(\dot{q},q,\ddot{q}_r,\dot{q}_r)a \tag{2.24}$$

Taking the control law to be

$$\tau = Y\hat{a} - K_D s \tag{2.25}$$

which includes a "feedforward" term $Y\hat{a}$ in addition to a simple PD term $K_D s$ leads to

$$\dot{V}(t) = s^T Y\tilde{a} - s^T K_D s + \dot{\hat{a}}^T\Gamma^{-1}\tilde{a} \tag{2.26}$$

Updating the parameter estimates $\hat{a}$ according to the correlation integrals

$$\dot{\hat{a}} = -\Gamma Y^T s \tag{2.27}$$

then yields

$$\dot{V}(t) = -s^T K_D s \le 0 \tag{2.28}$$

This implies that the output error converges to the surface $s = 0$ which in turn shows that $\tilde{q}$ and $\dot{\tilde{q}}$ tend to zero at steady-state. Therefore, both global stability of the system (i.e., boundedness of q, $\dot{q}$ and $\hat{a}$) and convergence of the tracking error are guaranteed by the above adaptive controller. ♦

Adaptive control is an appealing approach for controlling uncertain linear systems. However, so far it can only deal with parametric uncertainties; it cannot deal with unmodeled dynamics such as disturbance inputs. For nonlinear systems, its range of applicability is rather limited. It generally requires linear parametrization of the control law or of the system dynamics.

2.3 Gain Scheduling

Control of nonlinear systems by gain scheduling [2, 3, 8] is based on the idea of linearizing (to be explained in the sequel) the system equations around certain operating points and designing a linear controller for each region of operation. The resulting controllers are then combined by interpolation in order to obtain an overall controller for the original nonlinear system. The controller coefficients are varied continuously according to the value of the *scheduling signals*, also called scheduling variables, which might be either exogenous signals or endogenous signals with respect to the plant. In broad terms, the design of a gain scheduled controller for a nonlinear plant can be described as a four-step procedure, though differing technical avenues are available in each step.

First, a linear parameter-varying model of the plant is computed. The most common approach for doing this is based on Jacobian linearization of the nonlinear plant about a family of equilibrium points, also called operating points or set points. This yields a parametrized family of linearized plants and forms the basis of what is called *linearization scheduling.* Another approach is *quasi-LPV* scheduling, in which the plant dynamics are rewritten to disguise nonlinearities as time-varying parameters that are used as scheduling variables. Depending on the specific plant structure, quasi-LPV scheduling need not involve any Jacobian linearization of the plant dynamics.

The second step is to use linear design methods to design linear controllers for the LPV plant model that arises in either linearization or quasi-LPV scheduling. This design process may result directly in a family of linear controllers corresponding to the linear parameter dependent plant, or there may be an interpolation process to arrive at a family of linear controllers from a set of controller designs at isolated values of the scheduling variables.

The actual gain scheduling, the third step, involves implementing the family of linear controllers such that the controller coefficients (gains) are varied (scheduled) according to the

current value of the scheduling variables. Various issues arise here, depending on whether a linearization scheduling or quasi-LPV approach is used.

The fourth step is performance assessment. Locally speaking, the performance and stability assessment of the controller may allow analytical investigation, due to its linear nature. However, the global performance and stability properties of the overall system can be investigated only by (extensive) simulations.

2.3.1 Linearization Gain Scheduling

To design a linearization scheduling controller, first, the nonlinear plant equations given below (2.29) should be rewritten in a linear parameter dependent form.

$$\begin{aligned} \dot{x} &= a(x,u,w,v) \\ z &= c_1(x,u,w,v) \\ y &= c_2(x,w,v) \end{aligned} \tag{2.29}$$

where x is the state, u is the input, z denotes an error signal to be controlled, and y denotes a measured output available to the controller and usually includes penalized variables, tracking commands, and some of the state variables. The signal w captures parametric dependence of the plant on exogenous variables, while the signal v represents external "input functions" such as reference signals, disturbances and noises. The variable w is included as part of the parametrization of the linearization family, therefore, in linearization scheduling, Eqn.(2.29) is not linearized with respect to w.

Definition 2.2: *Equilibrium family* [7]

The functions $x_e(\sigma)$, $u_e(\sigma)$, $w_e(\sigma)$ and $v_e(\sigma)$ define an *equilibrium family* for the plant (2.29) on the set S if

$$a(x_e(\sigma),u_e(\sigma),w_e(\sigma),v_e(\sigma)) = 0\,, \qquad \sigma \in S \tag{2.30}$$

Associated with this equilibrium family is the error equilibrium family

$$z_e(\sigma) = c_1(x_e(\sigma), u_e(\sigma), w_e(\sigma), v_e(\sigma)) = 0\,, \qquad \sigma \in S \tag{2.31}$$

and the measured output equilibrium family

$$y_e(\sigma) = c_2(x_e(\sigma), w_e(\sigma), v_e(\sigma)) = 0\,, \qquad \sigma \in S \tag{2.32}$$

With the definition of the deviation variables as

$$x_\delta(t) = x(t) - x_e(\sigma)\,,\;\; z_\delta(t) = z(t) - z_e(\sigma)\,,\;\; y_\delta(t) = y(t) - y_e(\sigma)\,,\; u_\delta(t) = u(t) - u_e(\sigma) \tag{2.33}$$

there exists a plant linearization family that can be written in the linear parameter-varying form

$$\begin{bmatrix} \dot{x}_\delta \\ z_\delta \\ y_\delta \end{bmatrix} = \begin{bmatrix} A(\sigma) & B_1(\sigma) & B_2(\sigma) \\ C_1(\sigma) & D_{11}(\sigma) & D_{12}(\sigma) \\ C_2(\sigma) & D_{21}(\sigma) & 0 \end{bmatrix} \begin{bmatrix} x \\ v \\ u \end{bmatrix}\,,\; \sigma \in S \tag{2.34}$$

where, for example,

$$A(\sigma) = \frac{\partial a}{\partial x}(x_e(\sigma), u_e(\sigma), w_e(\sigma), v_e(\sigma)) \tag{2.35}$$

and so on. The design of a linearization gain scheduled controller entails designing a linear controller family corresponding to the plant linearization family (2.34). At each fixed σ linearization (2.34) describes the local behavior of the nonlinear plant about the corresponding equilibrium. This results in a linear controller family parametrized by σ, with input signal $y_\delta(t)$ (including, if appropriate, $r_\delta(t)$) and output signal $u_\delta(t)$. The controller family is of the form

$$\begin{bmatrix} \dot{x}^c_\delta \\ u_\delta \end{bmatrix} = \begin{bmatrix} F(\sigma) & G(\sigma) \\ E(\sigma) & H(\sigma) \end{bmatrix} \begin{bmatrix} x^c_\delta \\ y_\delta \end{bmatrix}, \qquad \sigma \in S \tag{2.36}$$

In the best case a family of controllers can be directly designed by some particular design method like LPV or LFT (linear fractional transformation). However, in most cases, a set of so-called *point designs* is performed at selected equilibria, with the restriction that all designs are of the same dimension. Then a family of linear controllers is computed by interpolating the point designs. In order to avoid discontinuities in the controller coefficients or possible chattering behavior, the indexed controllers are interpolated with respect to the parameter σ in a smooth, at least continuous way. Interpolating state-space controller coefficients is one option. Another option is to interpolate features of the input-output representations of the indexed set of controllers, for example, coefficients of the transfer function.

Example 2.4: Consider the scalar nonlinear plant [7]

$$\begin{aligned} \dot{x} &= -x + u, \\ y &= \tanh x, \\ z &= r - y = r - \tanh x \end{aligned} \tag{2.37}$$

Let the objective be that $y(t)$ track an exogenous reference signal $r(t)$ with acceptable transient response and zero steady-state error for constant $r(t)$. There is an obvious equilibrium family that corresponds to zero tracking error, and this family can be parametrized by $\sigma \in (-1,1)$ as

$$\begin{aligned} x_e(\sigma) &= u_e(\sigma) = \tanh^{-1}\sigma, \\ r_e(\sigma) &= y_e(\sigma) = \sigma, \\ z_e(\sigma) &= 0 \end{aligned} \tag{2.38}$$

Defining the deviation variables as dictated by (2.33), the linearized plant family in deviation coordinates becomes

$$\begin{aligned} \dot{x}_\delta &= -x_\delta + u_\delta, \\ y_\delta &= (1-\sigma^2)x_\delta, \\ z_\delta &= r_\delta - y_\delta \end{aligned} \tag{2.39}$$

Now, (11) can be considered in terms of the parametrized transfer function

$$\frac{Y_\delta(s)}{U_\delta(s)} = \frac{1-\sigma^2}{s+1} \tag{2.40}$$

Consider now unity feedback with a PI controller acting on the tracking error. Such a linear controller has a transfer function of the form

$$\frac{U_\delta(s)}{Z_\delta(s)} = k_p(\sigma) + \frac{k_i(\sigma)}{s} \tag{2.41}$$

The PI gains can be selected as functions of σ so that the family of closed-loop linear systems has a transfer function that is independent of σ. For example, choosing

$$k_p(\sigma) = \frac{1}{3(1-\sigma^2)}, \quad k_i(\sigma) = \frac{1}{(1-\sigma^2)} \tag{2.42}$$

yields the closed-loop transfer function

$$\frac{Y_\delta(s)}{R_\delta(s)} = \frac{(1/3)s+1}{s^2+(4/3)s+1} \tag{2.43}$$

Assuming the characteristics of this transfer function are suitable, the linear design process is thus completed and it remains to implement the gain scheduled controller.

2.3.2 Quasi-LPV Gain Scheduling

This approach is based on the possibility of rewriting the plant equation (2.29) in a form where nonlinear terms can be hidden with newly defined, time-varying parameters that are then included in the scheduling variable, in addition to *w*. Since nonlinear terms involve the state, some state variables must be relabeled as parameters in various parts of the model, while they remain dynamical variables elsewhere. Specifically, it is assumed that the state can be divided in two partitions as $x(t) = [x_a(t) \quad x_b(t)]^T$ where x_a comprises those state components that are sometimes rewritten as parameters. With $\sigma(t) = [x_a(t) \quad w(t)]^T$, the

fundamental assumption of the approach is that the plant description (2.29) can be written in the quasi-LPV form as

$$\begin{bmatrix} \dot{x}_a \\ \dot{x}_b \\ z \\ y \end{bmatrix} = \begin{bmatrix} A_{11}(\sigma) & A_{12}(\sigma) & B_{1a}(\sigma) & B_{2a}(\sigma) \\ A_{21}(\sigma) & A_{22}(\sigma) & B_{1b}(\sigma) & B_{2b}(\sigma) \\ C_{1a}(\sigma) & C_{1b}(\sigma) & D_{11}(\sigma) & D_{12}(\sigma) \\ C_{1b}(\sigma) & C_{2b}(\sigma) & D_{21}(\sigma) & D_{22}(\sigma) \end{bmatrix} \begin{bmatrix} x_a \\ x_b \\ v \\ \hat{u}(\sigma,u) \end{bmatrix} \tag{2.44}$$

where $\sigma(t)$ is treated as a time-varying parameter that is unknown a priori, but can be measured for implementation purposes. Also, the function $\hat{u}(\sigma,u)$ is assumed to be invertible with respect to u, that is, there exists a function $\hat{u}^{-1}$ such that

$$\hat{u}^{-1}(\sigma,\hat{u}(\sigma,u)) = u \tag{2.45}$$

If the plant equations can be written as (2.44), then a linear control design for the control input $\hat{u}$ yields a design for the actual control via $u = \hat{u}^{-1}(\sigma,\hat{u})$.

Example2.5*:* A quasi-LPV formulation for plant (2.44) based on replacing x by σ in a portion of the measured output leads to the LPV plant [7]

$$\begin{aligned} \dot{x} &= -x + u, \\ y &= (1/\sigma)\tanh \sigma x, \\ z &= r - y. \end{aligned} \tag{2.46}$$

The plant description (2.44) introduces additional behaviors beyond that of (2.29), and this typically increases the difficulty of obtaining a suitable controller. A particularly difficult situation is that of rapidly varying $\sigma(t)$, in which case a controller must hedge against behaviors of the quasi-LPV description that might be well beyond the capabilities of the original plant. The LPV controller designed as given in (2.47) below where x^c is the controller state variable, is one that measures y, σ, and possibly $\dot{\sigma}$, to produce u so that the effect of the exogenous signal, v, on the error, z, is minimized to an appropriate sense.

$$\begin{bmatrix} \dot{x}^c \\ u \end{bmatrix} = \begin{bmatrix} F(\sigma(t),\dot{\sigma}(t)) & G(\sigma(t),\dot{\sigma}(t)) \\ E(\sigma(t),\dot{\sigma}(t)) & H(\sigma(t),\dot{\sigma}(t)) \end{bmatrix} \begin{bmatrix} x^c \\ y \end{bmatrix} \tag{2.47}$$

2.4 Nonlinear Model Predictive Control

Model predictive control (MPC) [2, 4, 5, 6] refers to the class of controllers in which control inputs are selected based on an optimization criterion that is formulated over a prediction horizon, using an explicit model to predict the effect of future inputs on internal states or outputs. It is especially favored in the chemical process control industry, where the sampling times are larger (minutes) than for mechanical applications. The theory and applications of Linear MPC (LMPC) is a lot more developed than that of Nonlinear Model Predictive Control (NMPC) [2, 4]. However, since linear models are not adequate for highly nonlinear or moderately nonlinear processes with large range of operation, research attention has been devoted to NMPC and-–even though the theory is not complete—commercial NMPC packages are already in the process control market. NMPC is especially included here because it bears quite a resemblance to the SDRE formulation.

NMPC is an optimization-based control strategy which is well suited for constrained, multivariable processes. A sequence of control moves is computed to minimize an objective function which includes predicted future values of the controlled outputs. The predictions are obtained from a nonlinear process model. The optimization problem is solved subject to constraints on input and output variables, as well as constraints imposed by the nonlinear model equations. This formulation yields an open-loop controller. Feedback is included by implementing only the manipulated inputs computed for the present time step, then moving the prediction horizon forward one step and repeating the problem for the next time step using new process measurements. For this reason, NMPC often is called nonlinear receding horizon control. Calculation of the manipulated input sequence requires the on-line solution of a nonlinear programming problem.

2.4.1 Problem Formulation

Since NMPC is solved on-line, the discrete-time formulation of the process is often used. The nonlinear process model is represented by

$$x(k+1) = f(x(k), u(k))$$
$$y(k) = h(x(k)) \tag{2.48}$$

where x is an n-dimensional vector of state variables, u is an m-dimensional vector of input variables, and y is a p-dimensional vector of controlled output variables. The input is found by solving the following optimization problem over the prediction horizon P, with M being the control horizon.

$$\min_{u(k|k),u(k+1|k),\ldots,u(k+M-1|k)} J = \varphi[y(k+P\,|\,k)] + \sum_{j=0}^{P-1} L[y(k+j\,|\,k), u(k+j\,|\,k), \Delta u(k+j\,|\,k)] \tag{2.49}$$

where $u(k+j\,|\,k)$ is the input $u(k+j)$ calculated from information available at time k, $y(k+j\,|\,k)$ is the output $y(k+j)$ calculated from information available at time k, $\Delta u(k+j\,|\,k) = u(k+j\,|\,k) - u(k+j-1\,|\,k)$, φ is the terminal penalty term and L is a (possibly) nonlinear performance cost. Usually, quadratic functions of the following form are chosen for φ and L.

$$\begin{aligned} L &= [y(k+j\,|\,k) - y_s(k)]^T Q[y(k+j\,|\,k) - y_s(k)] + [u(k+j\,|\,k) \\ &- u_s(k)]^T R[u(k+j\,|\,k) - u_s(k)] + \Delta u(k+j\,|\,k) S \Delta u(k+j\,|\,k), \\ \varphi &= [y(k+P\,|\,k) - y_s(k)]^T Q[y(k+P\,|\,k) - y_s(k)] \end{aligned} \tag{2.50}$$

where $u_s(k)$ and $y_s(k)$ are steady-state targets for u and y, respectively, and Q, R, S are positive-definite weighting matrices. The optimization problem is usually a constrained one subject to constraints of the form

$$u_{\min} \le u \le u_{\max}, \quad \Delta u_{\min} \le \Delta u \le \Delta u_{\max}, \quad y_{\min} \le y \le y_{\max} \tag{2.51}$$

Solution of the NMPC optimization problem yields the input sequence $\{u(k\,|\,k), u(k+1\,|\,k), \ldots, u(k+M-1\,|\,k)\}$. Only the first input vector in the sequence is actually implemented: $u(k \mid k)$. Then the prediction horizon is moved forward one time step and the

problem is resolved using new process measurements. The concept of model predictive control is illustrated in Figure 2.3.

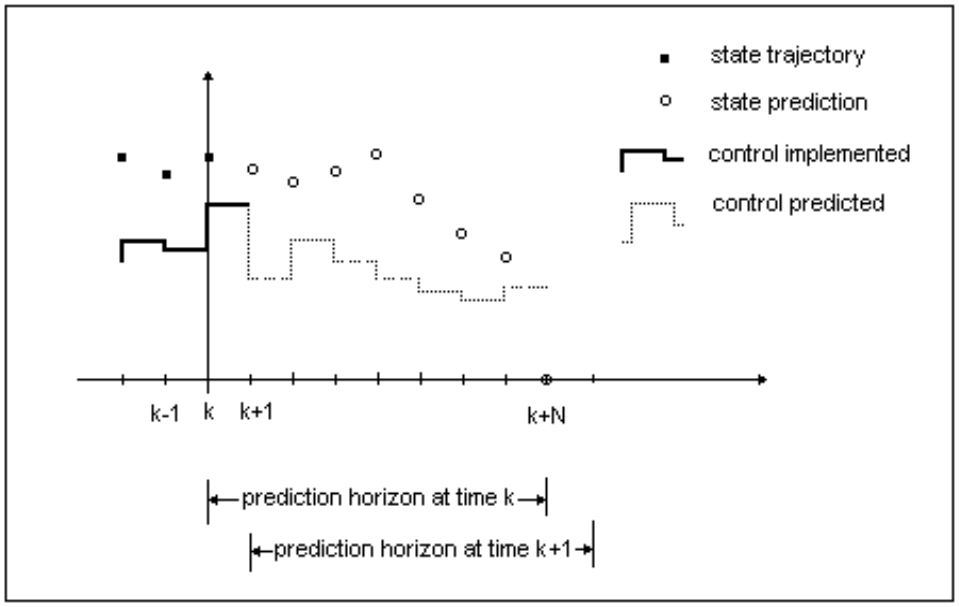


Figure 2.3 The model predictive control concept.

The principal controller tuning parameters are *M, P, Q, R, S* and the sampling period Δt. Current values of state variables are required to compute output predictions. Therefore, in the absence of full state feedback, it becomes necessary to design a nonlinear observer. For stable nonlinear systems, the sampling interval Δt should be chosen to provide a compromise between closed-loop performance and on-line computation. Small values generally improve performance but require a longer prediction horizon to adequately capture the process dynamic, which in turn complicates the (generally) nonconvex optimization problem. Large sampling intervals result in on-line computation, but they can result in poor performance. Moreover, the choice of Δt can have a drastic effect on robustness when the nonlinear system is unstable.

For a fixed prediction horizon, a smaller control horizon *M* yields more sluggish output responses and more conservative input moves. A large control horizon has the opposite effect on the performance. Since larger *M* also require increased on-line computation, in practice, *M* often must be chosen to provide a balance between performance and computation. The prediction horizon *P* has similar effects as the control horizon, *M*. The choice of the weighting matrices *Q, R, S* is rather ad hoc and they depend on the scaling of the problem.

The goal of the NMPC controller is to drive the process inputs and outputs to their target values in an optimal manner. If $u_s(k)$ and $y_s(k)$ are not chosen properly, the controller can exhibit steady-state offset in the presence of unmeasured disturbances and modeling errors. This offset problem may be handled by designing a disturbance estimator which gives the controller implicit integral action.

In many industrial applications, the desired targets $u_s(k)$ and $y_s(k)$ are calculated as a steady-state optimization at the plant level. In these cases, the desired targets are normally constant between plant optimizations, which are calculated at a slower time scale than the MPC controller operates. In some other applications, a final time objective may be optimized instead, which produces a time-varying trajectory for system states. More detail on the choice of $u_s(k)$ and $y_s(k)$ can be found in [6].

In order to provide robustness to model structural and parameter uncertainty, a disturbance estimator has been proposed in [5]. Although most results in this area indicate good performance in the presence of model uncertainty, a rigorous analysis that can provide sufficient results for robustness is not yet complete. The status of the stability problem for NMPC is similar. There exists a gap between observed results and theory. Experimental and simulation results indicate good stability performance on a wide class of problems, but tuning the algorithm to obtain stability is usually necessary.

Nominal stability is defined as the stability of a system which is free from modeling errors and disturbances. Nominal stability results are available for NMPC when the prediction horizon is infinite or a terminal state constraint is imposed. Unfortunately, both of these conditions are problematic from an implementation perspective. It is not possible to maintain the stabilizing properties of NMPC by reformulating the infinite horizon problem as a finite horizon problem with a simple terminal state penalty in the objective function. The assumption that the state vector can be driven to the origin in the presence of state constraints is known as *constrained null controllability.* The terminal state constraint is limiting because it is very difficult to ensure the existence of a control horizon such that a general nonlinear system is constrained null controllable [4]. Consequently, it is necessary to derive stabilizing NMPC formulation which are more suitable for on-line implementation. One of the initial results on nominal stability of NMPC is as follows.

Theorem 2.1:

For the nominal system

$$x(k+1) = F(x(k), u(k))$$
$$F(0,0) = 0 \tag{2.52}$$

if the MPC objective function is continuous, then the origin is an asymptotically stable fixed point under MPC with a region of attraction that consists of the subset of R^n that is constrained null controllable.

Proof of this theorem can be found in [5].

Today, the principle limitations in applying nonlinear models for on-line process control are model identification and robustness of control and software algorithms. To capture any nonlinearity in the process, extensive testing using a multi-level design is desired. This will make the testing period much longer than that of a linear plant test. Because of the use of a nonlinear model, the NMPC calculation usually involves a nonconvex nonlinear problem, for which the numerical solution is very challenging.

NMPC has been applied to a wide variety of simulated processes, most of which are processes with a small number of input and output variables. There is a need to develop NMPC formulations which are amenable to real-time implementation on large-scale processes. Finally, although commercial NMPC tools are available in the market, literature on this subject is lacking in experimental results.

2.5 Sliding Mode Control

Sliding mode control methodology [9] is based on a notational simplification, which amounts to replacing an n^{th} order tracking problem by a 1^{st} order stabilization problem. First, a lower dimensional manifold is found which can be stabilized and made positively invariant by an appropriate control input. Then a second control is chosen so those system trajectories, which are not initialized on the manifold, can be forced to the manifold within finite time. This is achieved by applying a control input that switches discontinuously between an upper and lower bound. The switching logic is determined as a function of the distance from the invariant manifold. Sliding control has shown to achieve robust performance by effectively

accounting for parameter uncertainties and unmodeled dynamics. The drawback is that it results in discontinuous high gain controllers with ensuing chattering of control. To remedy this, various refinements for smooth switching of the control input have also been proposed.

For the sake of simplicity, the formulation for single input systems will be explained here. The control design for multi-input systems is in parallel to that of single input systems, and more information about this subject can be found in [9].

Consider the single input dynamic system

$$y^{(n)} = f(x) + b(x)u \tag{2.53}$$

where y is the (scalar) output of interest, x is the state vector and u is the control input. The (nonlinear) function $f(x)$ is not exactly known, but the extent of the imprecision on $f(x)$ is upper bounded by a known continuous function of x. Similarly, the control gain $b(x)$ is not exactly known, but is of known sign and is bounded by known, continuous functions of x. Let

$$\tilde{x} = x - x_d \tag{2.54}$$

be the tracking error vector in the state. Let a time-varying surface $S(t)$ be defined in the state-space R^n by the scalar equation $s(x,t) = 0$, where

$$s(x,t) = (\frac{d}{dt} + \lambda)^{n-1}\tilde{x} \tag{2.55}$$

where λ is a strictly positive constant, the choice of which is related to the bandwidth. For example, if n=3,

$$s = \ddot{\tilde{x}} + 2\lambda\dot{\tilde{x}} + \lambda^2\tilde{x} \tag{2.56}$$

that is, s is simply a weighted sum of the acceleration, velocity and position error. Given the initial condition $x(0) \equiv x_d(0)$, the problem of tracking $x \equiv x_d$ is equivalent to that of remaining on the surface $S(t)$ for all $t>0$. In fact, with the initial condition $x(0) \equiv x_d(0)$, $s \equiv 0$

represents a linear differential equation whose unique solution is $\tilde{x} \equiv 0$. Thus, the problem of tracking the n-dimensional vector x_d (i.e., the original n^{th} order tracking problem in x) is in effect replaced by a 1^{st} order stabilization problem in s. This stabilization task can be achieved by choosing the control law u in (2.53) such that outside of $S(t)$

$$\frac{1}{2}\frac{d}{dt}s^2 \leq -\eta |s| \tag{2.57}$$

where η is a positive constant. Equation (2.57) is known as the *sliding condition*, which states that the squared "distance" to the surface, as measured by s^2, decreases along all system trajectories; thus constraining the trajectories to move towards the surface $S(t)$. Since $S(t)$ is an invariant set, once the trajectories reach it, they remain there for all times. The robustness property of the controller ensues from because s^2 remains a Lyapunov-like function of the closed-loop system, despite the presence of model imprecision and uncertainties.

The controller design procedure consists of two steps. First, a feedback control law u is selected so as to verify sliding condition (2.57). However, in order to account for modeling imprecision and disturbances, the control law must be discontinuous across $S(t)$. This discontinuity leads to chattering (Figure 2.4), which is undesirable in practice because it involves high control activity and further may excite high-frequency dynamics neglected in the course of modeling. Thus, in a second step, the discontinuous control law u is suitably smoothed to achieve an optimal trade-off between control bandwidth and tracking precision. This can be achieved by smoothing out the control discontinuity in a thin *boundary layer* (Figure 2.5) neighboring the switching surface

$$B(t) = \{x, | s(x,t) | \leq \Phi\}, \quad \Phi > 0 \tag{2.58}$$

While the 1^{st} step accounts for parametric uncertainty, the second step achieves robustness to high frequency unmodeled dynamics. Boundary layer thickness Φ can be made time-varying, and can be monitored so as to well exploit the control "bandwidth" variable.

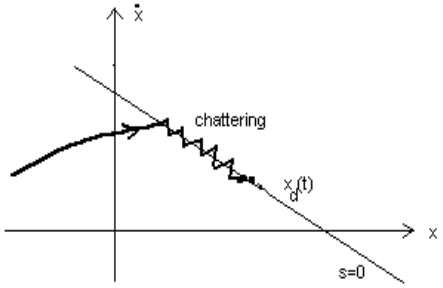


Figure 2.4 Chattering.

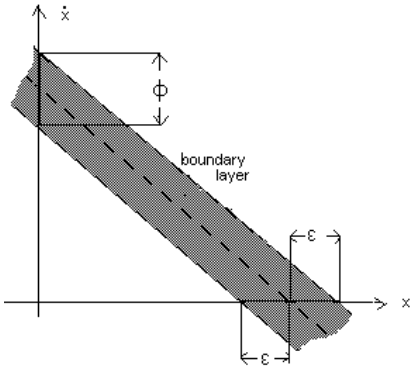


Figure 2.5 The boundary layer

Example 2.6: Consider the first order system [9]

$$\ddot{x} = f + u \tag{2.59}$$

where the dynamics f, possibly nonlinear or time-varying, is not exactly known but estimated as $\hat{f}$. The estimation error on f is assumed to be bounded by some known function $F = F(x, \dot{x})$:

$$| \hat{f} - f | \leq F \tag{2.60}$$

For instance, given the system

$$\ddot{x} + a(t)\dot{x}^2 \cos 3x = u \tag{2.61}$$

where $a(t)$ is unknown but verifies $1 \leq a(t) \leq 2$, one has

$$\hat{f} = -1.5\dot{x}^2 \cos 3x, \quad F = 0.5\dot{x}^2 \cos | 3x | \tag{2.62}$$

In order to have the system track $x(t) \equiv x_d(t)$, let the sliding surface be defined as

$$s = \dot{\tilde{x}} + \lambda \tilde{x} \tag{2.63}$$

Differentiating (2.63), one gets

$$\dot{s} = \ddot{x} - \ddot{x}_d + \lambda\dot{\tilde{x}} + f + u - \ddot{x}_d + \lambda\ddot{\tilde{x}} \tag{2.64}$$

Thus, the best approximation $\hat{u}$ of a continuous control law that would achieve $\dot{s} = 0$ is

$$\hat{u} = -\hat{f} + \ddot{x}_d - \lambda\dot{\tilde{x}} \tag{2.65}$$

which is the best estimate of the equivalent control. In order to satisfy the sliding condition (5) despite uncertainty in the dynamics of *f*, a term discontinuous across the surface *s = 0* is added to $\hat{u}$:

$$u = \hat{u} - k\,\mathrm{sgn}(s) \tag{2.66}$$

By choosing $k = k(x, \dot{x})$ large enough, (2.57) can be satisfied at all times. In fact,

$$\frac{1}{2}\frac{d}{dt}s^2 = \dot{s}.s = [f - \hat{f} - k\,\mathrm{sgn}(s)].s = (f - \hat{f})s - k\,|\,s\,| \tag{2.67}$$

so that, letting

$$k = F + \eta \tag{2.68}$$

$\frac{1}{2}\frac{d}{dt}s^2 \le -\eta\,|\,s\,|$ (2.57) is achieved as desired. The control discontinuity *k* across the surface $s = 0$ increases with the extent of parametric uncertainty.

In order to illustrate smooth switching across the surface $s = 0$, assume that the desired trajectory is $x_d(t) = \sin(\pi t/2)$ and the switching control law in (2.66) is replaced by a *sat* function instead of the *sign* function. Thus,

$$u = \hat{u} - ksat(s/\Phi) \tag{2.69}$$

The simulation results in Figures 2.6 and 2.7 were obtained by choosing λ=20, η=Φ=0.1 and $a(t) = |\sin t| + 1$. Figure 2.6 shows the rather chattering control input and the tracking error obtained by the discontinuous control law (2.66). Figure 2.7 shows the same variables with the smoothed control law chosen as in (2.69). Although the tracking error obtained with the control law (2.66) is smaller, the control input and the tracking error obtained by using (2.69) are smoother. This may be crucial for hardware safety and reliability.

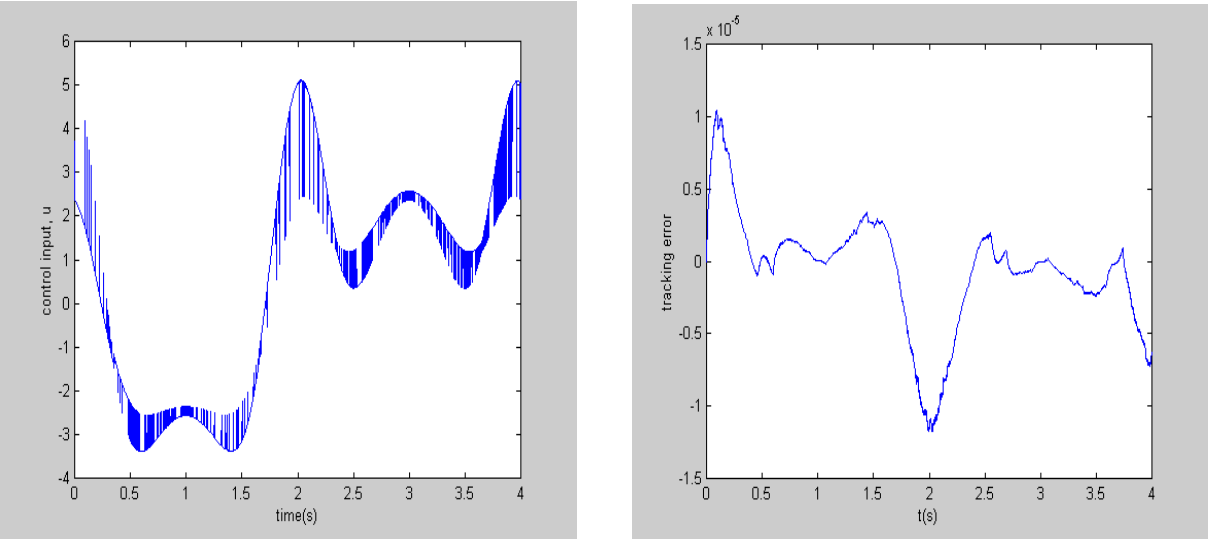


Figure 2.6 Switched control input and the resulting tracking performance

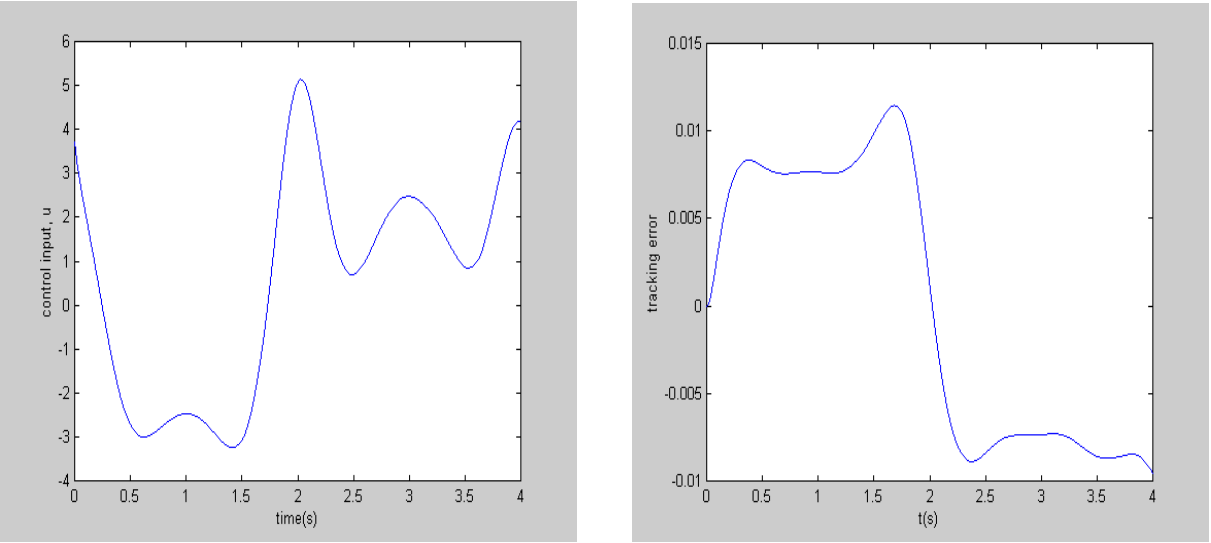


Figure 2.7 Smooth control input and the resulting tracking performance

2.6 Comparisons and Comments

In this section, the nonlinear control methods presented above are compared in terms of their range of applicability, stability, robustness, computational cost and allowance for physical intuition characteristics.

2.6.1 Range of Applicability

Feedback linearization, which is the most traditional nonlinear control method next to Jacobian linearization, can be applied only if the zero-dynamics of the system is stable. Hence, a system is either feedback linearizable or not. Even if it does possess the feedback linearizability property, it may not be feasible to do so because the input needed to cancel the (sometimes beneficial) nonlinearities may be exceedingly large.

Adaptive control works quite well for linear systems, since it deals with parametric uncertainties. For nonlinear systems, it is usually necessary that the whole plant dynamics be known and can be linearly parametrized. In short, the nonlinear counterpart of adaptive control is not very well developed yet.

Gain scheduling is rather "ad-hoc", including the problem formulation itself. While this allows room for physical intuition in the formulation process, for complex higher order systems it becomes a liability. In addition, the linearization scheduling approach does not apply when little information is carried by plant linearizations about constant equilibria. Also, the quasi-LPV approach, because of its conservative nature, may not yield controllers.

With NMPC, controllers can be designed for constrained, multivariable processes. Most of the results available are for processes with a small number of inputs and outputs.

While having superior robustness properties, a drawback of sliding mode control would be that the high gains required for "perfect" control might exceed the capacities of physical actuators. In addition, the system itself must be able to withstand "chattering" if perfect control is desired.

2.6.2 Stability

As mentioned above, feedback linearization can be used only if the zero-dynamics of the system are stable. However, only local stability is guaranteed for the internal dynamics even if the zero-dynamics is globally exponentially stable. But once the system is put into linear form by a change of coordinates, it is straightforward to design a stabilizing controller from linear control theory provided that the nonlinear model is accurate enough.

In gain scheduling, the individual regions of operation are linear and locally asymptotically stable. However, the overall system is still nonlinear, which makes global stability analysis difficult if not impossible. The quasi-LPV approach does offer guaranteed stability at the expense of computational cost. On the other hand, the stability tool of linearization scheduling is the rule of thumb that the structure of the system dynamics (e.g. eigenvalues) vary "sufficiently slowly".

There exists a large gap between theoretical NMPC stability results and practice. The conditions for which theoretical stability results are available are difficult to realize in reality.

Since the idea of sliding mode control is to drive the states to a stable manifold and to keep it there (e.g. drive system states to the origin along a stable eigenvector) it is inherently stable.

2.6.3 Robustness

Since it is based on a purely mathematical transformation, feedback linearization requires that the plant dynamics and parameters be well known for the feedback linearized system to fairly represent the original nonlinear plant. The unmodeled dynamics and parametric uncertainties can have an unpredictable effect on the controlled system since the control law was not designed to account for them. Consequently, feedback linearization is known to have poor robustness characteristics.

Gain scheduling does not require severe structural assumptions on the plant model. It can be used in the absence of complete, analytical plant models (database models). In particular, linearization scheduling can be used when plant information is limited to a few equilibria and the corresponding plant linearizations.

While adaptive control is very efficient for dealing with parametric uncertainties, it cannot compensate for unmodeled dynamics such as disturbances.

Although there does not exist a rigorous analysis on the robustness characteristics of NMPC, results show the characteristics are favorable. Nevertheless, the choice of the sampling interval Δt can be very crucial to robustness especially if the nonlinear system is unstable. Also, poor choices of the steady-state targets $u_s(k)$ and $y_s(k)$ may result in steady-state error in the presence of uncertainties.

Sliding mode control is most appreciated for its good robustness properties. Theoretically, "perfect" control can be achieved with properly modeled uncertainties and unmodeled dynamics. However, "perfect" control requires high actuator power which may not be available, and chattering of control input which may be harmful for the equipment. Nevertheless, a compromise between state error and control input can result in a controller that is more robust than those obtained by other control methods.

2.6.4 Computational Cost

Of the nonlinear control methods reviewed in this chapter, those that require online computation are adaptive control, gain scheduling and nonlinear model predictive control. A self-tuning adaptive controller involves more computation compared to a model reference adaptive controller because the estimated plant parameters are updated at each time step before the control is computed. The computational burden of linearization gain scheduling approaches is much less than that of quasi-LPV approaches.

The most computationally burdensome among these methods is the nonlinear model predictive control. The control input is computed by the on-line solution of a nonlinear programming problem. Nevertheless, in the process control industry where NMPC is used the most, the time intervals are too large (minutes) for this to impose a problem. The prediction horizon P and the control horizon M must be chosen to achieve a proper trade-off between performance and computational cost.

2.6.5 Allowance for Physical Intuition

It is desirable that there be room for physical intuition in controller design. This gives the designer the opportunity to tune the performance of the controller by adjusting the physically significant parameters.

Feedback linearization is a method which allows very little room for physical intuition, for the physical significance of the terms and/or parameters are easily lost during coordinate transformation. Design by gain scheduling, on the other hand, preserves well-understood linear intuition and is carried out using the physical variables in the plant model. Naturally, ad-hoc methods allow more room for physical intuition.

2.7 References

[1] K.J. Astrom and B. Wittenmark, *Adaptive Control*, Addison-Wesley, 1995.

[2] M.A. Henson, "Nonlinear model predictive control: current status and future directions", *Computers and Chemical Engineering*, vol.23, 187-202, 1998.

[3] H.K. Khalil, *Nonlinear Systems*, Prentice-Hall 1996.

[4] E.S. Meadows, "Dynamic Programming and Model Predictive Control", *Proceedings of the American Control Conference*, 1997.

[5] E.S. Meadows and J. B. Rawlings, "Model Predictive Control", Chapter 5 of ?.

[6] J.B. Rawlings, "Tutorial: Model Predictive Control Technology", *Proceedings of the American Control Conference*, 1999.

[7] W.J. Rugh, and J.S. Shamma, "Survey Paper, Research on Gain Scheduling", *Automatica*, vol.36, pp. 1401-1425, 2000.

[8] J.S. Shamma and M. Athans, "Analysis of Gain Scheduled Control for Nonlinear Plants", *IEEE Transactions on Automatic Control*, vol.35, no.8, 898-907, 1990.

[9] J-J.E. Slotine, and W. Li, *Applied Nonlinear Control,* Prentice-Hall, 1991.

Chapter 3

State-Dependent Riccati Equation (SDRE) Control

This chapter consists of background material regarding SDRE control. First, the formulation of the SDRE technique is presented. Then follows a discussion about the characteristics of SDRE as a nonlinear control method. Next, a literature survey of SDRE research grouped under specific topics is presented, the topics being stability, optimality, robustness, simulation results and experimental results. Finally, a list of open issues in SDRE research is given.

3.1 SDRE Formulation

The problem considered is the infinite-horizon regulation of general autonomous nonlinear systems affine in the input. Given the system equation

$$\dot{x} = f(x) + g(x)u \tag{3.1}$$

and the performance index

$$J = \frac{1}{2}\int_0^\infty (x^T Q(x)x + u^T R(x)u)dt \tag{3.2}$$

which allows for trading-off state error x versus control input u, via the weighting matrices $Q(x) \geq 0$, $R(x) > 0 \ \ \forall x$, respectively, where $x \in R^n$, $u \in R^m$, $f(x) \in C^k$, $g(x) \in C^k$, $Q(x) \in C^k$, $R(x) \in C^k$, with $k \geq 1$. Assuming that $f(0)=0$ and $g(x) \neq 0 \ \forall x$, it is desired to find a feedback control law $u(x)$ which will regulate the system to the origin $\forall x$. The SDRE method approaches the problem by mimicking the LQR formulation for linear systems. Accordingly, the system equations are first written in the form

$$\dot{x} = A(x)x + B(x)u \tag{3.3}$$

where $f(x) = A(x)x$ (the choice of the matrix $A(x)$ is not unique) and $g(x)=B(x)$. The former parametrization is possible if and only if $f(0)=0$ and $f(x)$ is continuously differentiable [1, 3]. Then it is proceeded as in the linear time-invariant case, and a state-feedback control law is obtained in the form of

$$u(x) = -K(x)x = -R^{-1}(x)B^T(x)P(x)x \tag{3.4}$$

where $P(x)$ is the unique, symmetric, positive-definite solution of the state-dependent algebraic Riccati equation

$$A^T(x)P(x) + P(x)A(x) + Q(x) - P(x)B(x)R^{-1}(x)B^T(x)P(x) = 0 \tag{3.5}$$

This method requires that the full state measurement vector be available and that the pair $(A(x), B(x))$ be pointwise controllable in the linear sense $\forall x$. This can be checked by forming the controllability matrix as in the linear systems sense and making sure that it has full rank in the domain of interest [3]. This condition simply ensures that the Algebraic Riccati Equation has a solution at the particular state x. The pointwise controllability condition is not necessarily equivalent to nonlinear controllability [11]. Due to the non-uniqueness of $A(x)$, different $A(x)$ choices may yield different controllability matrices and thus different pointwise controllability characteristics. In [11], connections between pointwise controllability and true nonlinear controllability are investigated. It is shown that global pointwise controllability of a state-dependent factorization is sufficient to guarantee weak controllability on a neighborhood of the origin, and global controllability of both types holds when the input matrix function $B(x)$ has rank equal to the dimension of the state space n, for all x. It is also concluded in this reference that whereas pointwise controllability (plus observability) for all x is sufficient to guarantee well-posedness of SDRE-based control algorithms, it is, in general, not sufficient to guarantee true controllability outside some possibly small neighborhood of the origin.

Despite quite encouraging simulation results, which make up the most part of SDRE literature, theoretical results on main issues like stability, optimality, robustness are quite scarce. This is hardly surprising, since state dependence of the matrices makes analyses quite complicated, more so for higher order systems.

3.2 Characteristics of SDRE

The formulation and computation of SDRE control is a simple, systematic procedure that is applicable to a wide range of nonlinear dynamical systems. There are infinitely many choices for the parametrization ($A(x)$, $B(x)$) among which a pointwise stabilizable pair must be chosen. While this is an easy task for lower order systems, it can be complicated for higher order systems. On the other hand, the extra design degrees of freedom arising from the non-uniqueness of the state-dependent coefficient parametrization can be utilized to enhance controller performance.

SDRE allows one to impose hard bounds on the control, control rate or even control acceleration to avoid actuator saturation. This capability is illustrated in [4, 23]. It is also reported in [23] that SDRE can directly be applied to unstable nonminimum phase systems. In the same reference, it was also reported from experience that SDRE does not cancel beneficial nonlinearities.

SDRE control produces a closed-loop system matrix $A_{CL}(x) = A(x) - B(x)K(x)$ which is pointwise Hurwitz $\forall x,$ in particular for x=0. Therefore, the origin of the closed-loop system is *locally* asymptotically stable [1, 3, 20]. However, for a nonlinear system, all eigenvalues of $A_{CL}(x)$ having negative real parts $\forall x \in R^n$ does not guarantee *global* asymptotic stability [20, 21, 29]. Stability of SDRE systems is still an open issue. Research results in SDRE literature on stability are summarized in Section 3.3.1.

Although a formal robustness analysis does not exist yet, simulation results show that SDRE control inherently has robustness characteristics due to its LQR nature. In addition, SDRE H_∞ control formulation has been proposed [3, 4, 13].

In implementing SDRE control, the most desirable option is to solve the state-dependent Riccati equation analytically. This may be possible for lower order systems or for systems with a special structure. In general, however, this is not possible and one has to solve

the SDRE numerically. This is quite straightforward with numerical tools such as MATLAB. The control computation can be done either on-line or off-line. Although computationally more costly, it may be desirable to compute the feedback control in real-time by solving the SDRE at a relatively high sampling rate. Real-time control becomes essential when the disturbances or the trajectories are not known, for example, in obstacle avoidance of a robot in an unstructured environment. On the other hand, off-line control computation may be more desirable in some cases, for example for safety reasons in aircraft control.

The computational burden associated with real-time SDRE control is higher than most nonlinear control methods, since the state-dependent Algebraic Riccati Equation must be solved at each time step. This is a major reason for the scarcity of experimental SDRE control results. It was observed that choice of the sampling interval less than a critical value could lead to instability. However, the computational cost is of polynomial growth rate with state dimension and with faster computers this will hopefully cease to be an issue. In fact, the experimental results that do exist reveal that the state-of-the-art computational power is more than enough to sustain SDRE control of mechanical systems with moderate to fast dynamics. Results of the Pendubot experiment presented in Chapter 6 demonstrate this.

The greatest advantage of SDRE control is that physical intuition is always there and the designer can directly control the performance by tuning the weighting matrices $Q(x)$ and $R(x)$. In other words, via SDRE, the design flexibility of LQR formulation is directly translated to control of nonlinear systems. Moreover, $Q(x)$ and $R(x)$ are not only allowed to be constant, but can also vary as functions of states. In this way, different modes of behavior can be imposed in different regions of the state-space. Consider the following example:

Example 3.1: Consider the following system

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -x_1 + x_2(1 - 3x_1^2 - 2x_2^2) + u \end{aligned} \tag{3.6}$$

The open-loop system has unstable eigenvalues equal to $0.5 \pm j0.866$ at the origin. However, it is pointwise stabilizable for the following choices of $A(x)$ and $B(x)$

$$A(x) = \begin{bmatrix} 0 & 1 \\ -1 & (1-3x_1^2-2x_2^2) \end{bmatrix}, \qquad B(x) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{3.7}$$

since the controllability matrix

$$W_{ctrb}(x) = [B \mid AB] = \begin{bmatrix} 0 & 1 \\ 1 & 1-3x_1^2-2x_2^2 \end{bmatrix} \tag{3.8}$$

is of full rank $\forall x$.

Figures 3.1-3 show the simulation results from the initial condition (0.5, -0.5) for different choices of $Q(x)$=diag($q_1(x)$, $q_2(x)$) and $R(x)$. In LQR (a), which corresponds to the top rows, LQR control was used with q_1=50, q_2=50, r=200. This choice yields loose regulation, with small control effort, yet large state norm. In LQR (b), which corresponds to the middle rows, LQR control was used with q_1=250, q_2=250, r=1. In contrast to the previous case, this choice yields tight regulation with small state norm, yet large control effort. In the third case, the plots of which are in the bottom rows, SDRE was used with

$$q_1(x) = 50 + 400 \mid x_1 \mid, \qquad q_2(x) = 50 + 400 \mid x_2 \mid, \qquad r(x) = 1 + 400(x_1^2 + x_2^2) \tag{3.9}$$

This choice of $Q(x)$ and $R(x)$ makes possible having the better traits of the two previous cases simultaneously. It can be seen that SDRE control yields small state norm *and* small control effort. This example illustrates how the design flexibility offered by SDRE can soften the classical state vs. input trade-off that is encountered with other control methods.

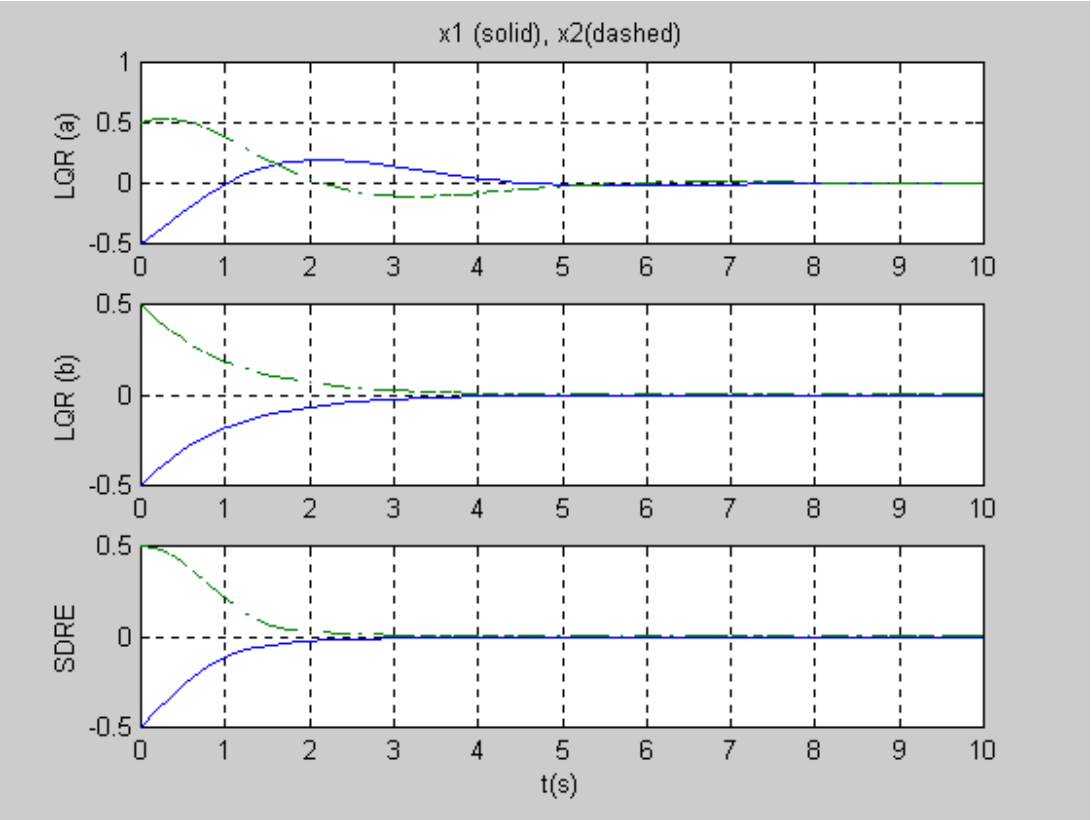


Figure 3.1 States x_1 and x_2.

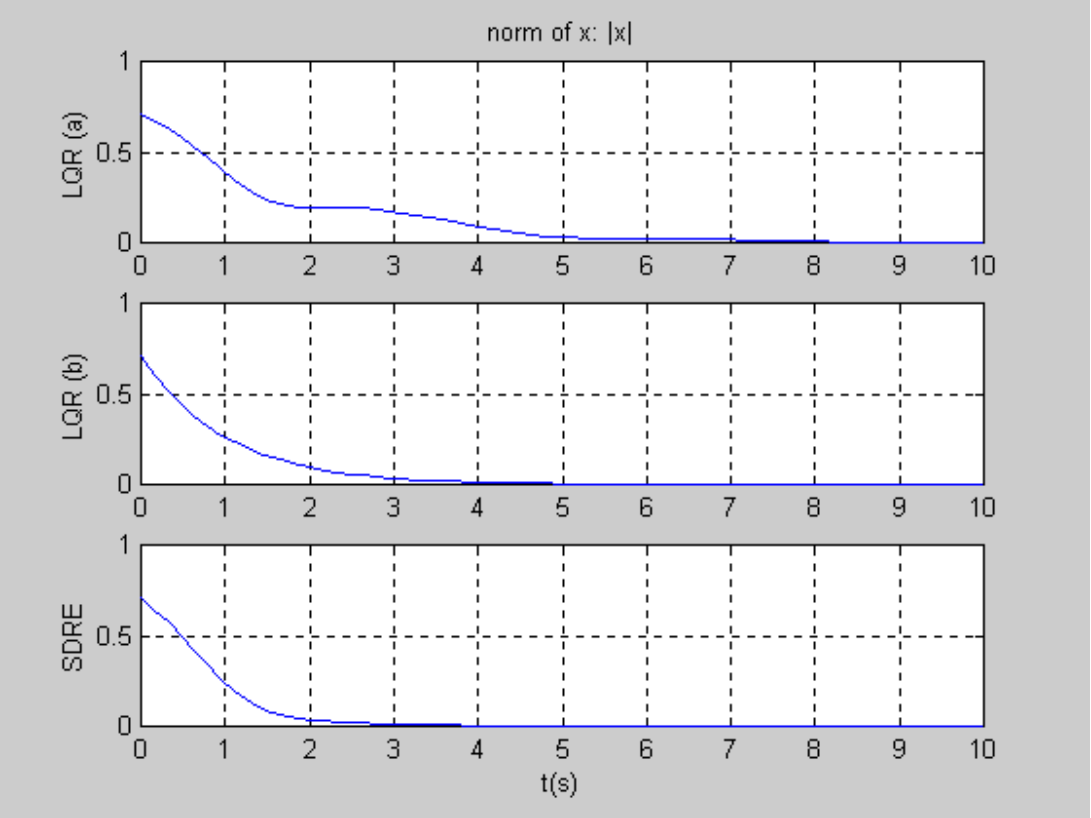


Figure 3.2 State norm: $|x|$.

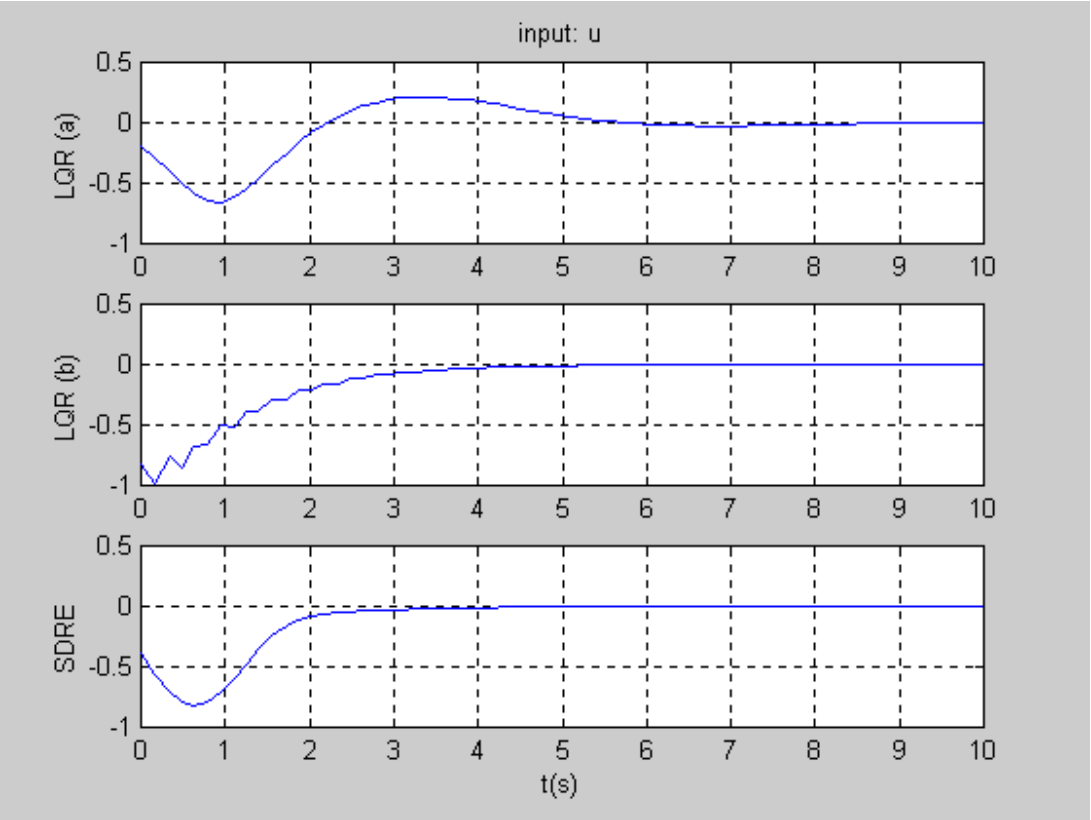


Figure 3.3 Input *u*.

3.3 Survey of SDRE Literature

The idea of using state feedback Riccati equation based linear control methods for nonlinear systems was originally proposed as far back as 1962 [27]. In this reference, the steady-state stabilizing solution to a state and time-dependent Riccati differential equation is obtained analytically for low-order example systems to provide a basis for a suboptimal control algorithm. After a limited number of variations on this idea for three decades, the idea gained renewed interest and was proposed by Banks in 1992 [1] and independently by Cloutier, et al. in 1996 [3]. Although the underlying theory is yet scant, impressive simulation results have caused the use of SDRE method for designing nonlinear controllers, observers and filters to rapidly emerge.

3.3.1 Stability

The origin of an SDRE-controlled nonlinear system is locally asymptotically stable [1,3]. This follows from the fact that the closed-loop coefficient matrix has stable eigenvalues for all points in the state-space. However, unlike linear systems, this is not enough to guarantee *global* asymptotic stability of the controlled system, which is the property that from *any* point in state-space, the origin can be reached by employing the SDRE control law. Issues like invariant sets and finite escape time may enter the picture and destroy the system's global asymptotic stability. In addition to having stable eigenvalues in a compact set containing the origin, it must be guaranteed that once the state trajectories enter there, they remain there forever. Thus, the region of stability of the origin must be an invariant set.

Unfortunately, in the first proposition of the theory by Banks [1], there are flaws in the stability proof [20]. In the paper it is claimed that having the eigenvalues of the closed loop $A(x)$ matrix in the left-hand plane for all x is sufficient to deduce global asymptotic stability. This claim was proved to be incorrect by a counterexample by Tsitoras, et al. [29]. In this example which is presented below, the system has a finite escape time even though the eigenvalues are stable everywhere. Thus it is not globally asymptotically stable.

Example 3.2: Consider the nonlinear system

$$\begin{aligned} \dot{x}_1 &= -x_1 + x_1^2 x_2 \\ \dot{x}_2 &= -x_2 \end{aligned} \tag{3.10}$$

These equations can be parametrized in the form $\dot{x} = A(x)x$ where A is analytic and given by

$$A(x) = \begin{bmatrix} -1 & x_1^2 \\ 0 & -1 \end{bmatrix} \tag{3.11}$$

where for each x the eigenvalues of A equal -1. With initial state $(x_1(0), x_2(0)) = (2,2)$, a routine calculation shows that for $t \in [0, T_c)$ with $T_c = \ln\sqrt{2}$, the solutions are

$$x_1(t) = \frac{2x_2(t)}{x_2^2(t) - 2} \text{ and } x_2(t) = 2e^{-t} \tag{3.12}$$

As can easily be checked, the solutions have a finite escape time at $T_c = \ln\sqrt{2}$; this is because $x_1(t)$ grows without bound as $t \to T_c$. Hence, the original system is not asymptotically stable for all initial states $x(0)$.

In [21, 22], sufficient conditions that guarantee local and global asymptotic stability are derived. The authors prove that if $A(x)$ is stable and continuous $\forall x \in R^n$, then the origin is an asymptotically stable equilibrium point. Moreover, if there exists an upper bound $M>0$ to the solution $\left\| e^{A(x)t} \right\|$ for all t, then the origin is globally asymptotically stable. This bound, however, is not easy to determine or enforce by feedback control. In the same paper, an upper bound for a stable matrix $A(x)$ is derived by using its Jordan canonical form. From here an estimate of the region of attraction of the origin is found, which is fairly conservative. In [9] sufficient conditions are derived to achieve global asymptotic stability for second order systems. These conditions are realized by tuning the state and input weighting matrices $Q(x)$ and $R(x)$ as certain functions of the states. The results are also verified experimentally on a magnetic levitation example. This work is explained in more detail in Chapter 4. In reference [15], stability conditions are found for scalar analytic systems and systems with full rank constant B matrices.

In [12], conditions guaranteeing semiglobal asymptotic stability of sampled data SDRE regulators for analytic nonlinear systems are derived, using discrete-time Lyapunov stability theory. It is shown that, if the discretized system equations satisfy stabilizability and detectability requirements, and if there exists a solution to the sampled data SDRE, then semiglobal asymptotic stability can be achieved by choosing the sampling interval less than a certain value δ_k, which is given by

$$\delta_k < \frac{1}{n(\max\limits_{i,j,t\in[t_k,t_{k+1}]} |A_{ij}(t)|)} \tag{3.13}$$

where n is the order of the system. Furthermore in [14], the condition that the state weighting matrix Q be strictly positive definite for stability is relaxed to positive semi-definiteness, by using LaSalle's principle and invariant set theory.

Lately, in [19], a modified Hopfield neural network (MHNN) is used to solve SDRE. Like the classical Hopfield network, the stability of MHNN is guaranteed. In addition, it provides more degrees of freedom, which can be used to accommodate different applications.

3.3.2 Optimality, or "lack thereof"

The SDRE control that is obtained is not necessarily the optimal one that minimizes the performance index (3.2), but usually "suboptimal". This is due to the non-uniqueness of the parametrization $f(x)=A(x)x$. Optimality of the control is not a concern in this study, given the benefits of the design flexibility of SDRE. Nevertheless, it has been reported that the SDRE control is not very different from the optimal one [3, 4, 23, 25]. In SDRE nonlinear regulation, the necessary condition for optimality, $H_u=0$, where H is the Hamiltonian of the system is always satisfied [3, 4]. Moreover, if the matrix functions $A(x)$, $B(x)$, $Q(x)$, $R(x)$ and $P(x)$ along with their gradients with respect to x_i are bounded in an open ball containing the origin, then under asymptotic stability, the necessary condition $\dot{\lambda} = -H_x$ is asymptotically satisfied at a quadratic rate as the state x is driven to zero. Thus, the SDRE control converges to the optimal one at a quadratic rate and the two controls are "almost" the same around the origin.

In [17] it is shown that if the optimal cost function $V(x)$ of the HJB equation

$$0 = \frac{\partial V}{\partial x} f - \frac{1}{4} \frac{\partial V}{\partial x} g R^{-1} g^T \frac{\partial V}{\partial x}^T + x^T Q x, \qquad V(0)=0. \tag{3.14}$$

where V is the optimal value of J, i.e., $V(x) = \min_{u(t)} J(x)$, satisfies the gradient condition

$$\frac{\partial V(x)}{\partial x} = 2x^T P(x) \tag{3.15}$$

for some positive definite matrix-valued function $P(x)$, then there always exists a parametrization $f(x)=A(x)x$ such that $P(x)$ is the solution of the SDRE (3.5) which gives the optimal feedback controller. It is also shown that there exists *at most* one $V(x)$ that satisfies condition (3.15). The bottom line of this argument is that it is possible for the SDRE control to match the optimal one if the "right" $A(x)$ is chosen. However, choosing the $A(x)$ that will yield the optimal control may not be straightforward. The following example from [17] illustrates this.

Example 3.2: Consider the optimal regulation problem for the following plant:

$$\begin{aligned} \dot{x}_1 &= -x_1 + x_1 x_2^2 \\ \dot{x}_2 &= -x_2 + x_1 u \end{aligned}, \qquad B = [0 \quad 1]^T \tag{3.16}$$

The cost function is as defined in (3.2) with $Q=2I$, $R=0.5$, where I is the *2x2* identity matrix. From the solution of the Hamilton-Jacobi Equation, the optimal control is found as:

$$u = -2x_1 x_2 \tag{3.17}$$

If $A(x)$ happens to be chosen as

$$A(x) = \begin{bmatrix} -1 + x_2^2 & 0 \\ 0 & -1 \end{bmatrix} \tag{3.18}$$

then SDRE fails to give a positive definite solution at any point since the pair $[A(x), B(x)]$ is not controllable for any x. If $A(x)$ is chosen to be

$$A(x) = \begin{bmatrix} -1 & x_1 x_2 \\ 0 & -1 \end{bmatrix} \tag{3.19}$$

then the SDRE defines a stabilizing controller; however, it is not optimal. The $A(x)$ matrix that yields the optimal solution is the less obvious choice

$$A(x)=\begin{bmatrix} -1 & x_1x_2 \\ -x_1x_2 & -1+x_1^2 \end{bmatrix} \tag{3.20}$$

which is difficult to guess from the dynamics of the plant unless the optimal cost is known.

Though the HJB equation yields the optimal control law, its solution is possible/feasible only for simple, low dimensional systems. On the other hand, SDRE offers certain computational advantages over HJB solution, since the computational complexity of the Algebraic Riccati Equation is only of polynomial growth rate with state dimension [17].

In a recent paper [16], it is shown that a nonlinear regulator constructed by SDRE is a local optimal solution of the original control problem, but not necessarily the true minimum. This argument is proved by using the three most conventional methods: Lagrange multipliers, minimum principle and dynamic programming.

In [5] the $A(x)$ matrix is parametrized as $A(x,\alpha(x))$ in order to investigate optimality properties. As such, if a certain partial differential equation can be solved, then $\alpha(x)$ can be determined such that all of the necessary conditions for optimality are satisfied. However, such a solution is hardly real-time implementable. Instead, a state-dependent parametrization updating scheme for the $A(x)$ matrix is developed, where the update is used to enhance control performance. A feedback control scheme is employed to approximately optimize over $\alpha(x)$.

The results in [5] and [17] slightly contradict the main result of a later paper, [6], where the recoverability of an optimal nonlinear state feedback control law of the form $u=k(x)$, $k(0)=0$, by extended linearization control techniques is investigated. An extended linearization technique is defined as one in which the nonlinear system equations are factored in a linear-like form $A(x)x + B(x)u$, and linear control techniques are used to obtain a closed loop system which has a pointwise Hurwitz coefficient matrix. Clearly, SDRE is an extended linearization technique. The main result of [6] is that there exists *no* extended linearization technique that is capable of recovering the optimal control law.

However, it is also shown in this paper that two alternative SDRE formulations are capable of recovering the optimal: the first one is by using negative-definite or even indefinite $Q(x)$ matrices over a subset of the state space. The second one is by using one of the infinitely many nonsymmetric solutions of the SDRE; since there are only $n(n+1)/2$ equations with n^2

unknowns. In order to find these n^2 equations, the SDRE is solved in conjunction with the symmetry condition

$$P_{ij}(x) + \sum_{k=1}^{n} \frac{\partial P_{ik}(x)}{\partial x_j} x_k = P_{ji}(x) + \sum_{k=1}^{n} \frac{\partial P_{jk}(x)}{\partial x_i} x_k \tag{3.21}$$

However, neither of these two alternative SDRE solutions is really an extended linearization technique, because they produce non-Hurwitz closed-loop coefficient matrices.

In [28], a suboptimal regulator for nonlinear affine systems is proposed based upon the combination of receding horizon and SDRE techniques. The proposed controller is shown to be nearly optimal, provided that a good approximation to the storage function in the original HJB equation is known in a neighborhood of the origin. The controller is capable of globally asymptotically stabilizing the plant, provided that enough computational power is available to solve a finite horizon optimization problem online. Since solving the HJB equation is very difficult, the authors introduce a finite horizon approximation to this infinite horizon optimization problem. The following lemma is the motivation for this idea.

Lemma 3.1:

Let S be a compact set containing the origin and the optimal storage function $V(x)$ be known for all $x \in S$. Let $c = \min_{x \in \partial S} V(x)$ and define the set $S_v = \{x : V(x) < c\}$. Consider the following two optimization problems:

$$\min_u \{J(x,u) = \frac{1}{2} \int_0^{\infty} (x^T Q(x) x + u^T R(x) u) dt\} \tag{3.22}$$

$$\min_u \{J_T(x,u) = \frac{1}{2} \int_0^{T} (x^T Q(x) x + u^T R(x) u) dt + V[x(T)]\} \tag{3.23}$$

subject to (3.1). Then the following hold:

1) An optimal solution for problem (3.23) is also optimal for (3.22) in the interval [0, T] provided that $x(T) \in S_v$.

2) Consider $T_1 > T$. If $x(T) \in S_v$, then a controller that optimizes J_T is also optimal with respect to J_{T_1} in [0, T].

This lemma suggests that if a solution to the HJB equation is known in a neighborhood of the origin, then it can be extended via an explicit finite horizon optimization. Thus, a receding horizon type control can be obtained by combining on-line optimization with an off-line phase to find a local solution to the HJB equation. However, finding/storing this local solution can be very computationally demanding as the dimension of the systems increase. So, the optimization problem is approximated by the following problem, using a positive-definite function $\Psi(x)$.

$$J_\Psi(x,t) = \min_u \int_t^T (x^T Q(x)x + u^T R(x)u)d\tau + \Psi[x(T)]\} \tag{3.24}$$

subject to (3.1). The error in this approximation, $e(x,t) \equiv J_\Psi(x,t) - V(x)$, is of first order if $\Psi(x)$ is chosen so that $\partial e[x(T),T]/\partial x \cong 0$. The proposed control algorithm in the paper is as follows:

1) If $x(t) \in S_v$, $\quad u = -\frac{1}{2} R^{-1} g^T \frac{\partial V}{\partial x}$ (3.25)

2) $x(t) \notin S_v$, solve a sequence of problems (3.23) until a solution such that $x(T) \in S_v$ is found.

3) Use the corresponding control law in the interval $[t_0, t_0 + \delta t]$.

The resulting control is globally optimal and thus, globally stabilizing. The algorithm is further modified to compensate for the computational complexity of finding $V(x)$.

3.3.3 Robustness

SDRE robustness analyses have been done individually for example problems rather than explicitly investigated as a separate topic. Although not quantified, it has been reported from experience that due to its LQR-like nature, SDRE exhibits robustness to disturbances

and unmodeled dynamics. In addition, SDRE nonlinear H_∞ control has been proposed [3, 4]. The following is a summary of the SDRE nonlinear H_∞ control formulation in [4].

Consider the general nonlinear system

$$\begin{aligned} \dot{x} &= f(x) + B_1(x)w + B_2(x)u \\ z &= c_1(x) + D_{12}(x)u \\ y &= c_2(x) + D_{21}(x)w \end{aligned} \tag{3.26}$$

where $D_{12}(x)$ and $D_{21}(x)$ have full rank, $D_{12}{}^T(x)D_{12}(x) = R_u$, $D_{21}(x)D^T{}_{21}(x) = R_w$, and w is a vector of exogenous inputs which may include tracking commands and/or disturbances. It is desired to bound the L_2-gain of the system. For $\gamma \geq 0$, the system (3.26) has L_2-gain less than or equal to γ if

$$\int_0^T \|z(t)\|^2 dt \leq \gamma^2 \int_0^T \|w(t)\|^2 dt \tag{3.27}$$

for all $T \geq 0$ and all $w \in L_2(0,T)$. If a controller can be found such that the closed-loop system is internally stable and such that the inequality (3.27) is satisfied, the exogenous signals will be locally attenuated by γ. The inequality (3.27) can be satisfied by solving the max-min differential game problem

$$\max_{w \in L_{2+}} \min_{u \in L_{2+}} \frac{1}{2} \int_0^\infty \|z(t)\|^2 - \gamma^2 \|w(t)\|^2 dt \tag{3.28}$$

subject to the constraints (3.26). The SDRE approach for obtaining an approximate solution of the nonlinear H_∞ problem (3.28) is

1) Use direct parametrization to bring the nonlinear dynamics (3.26) to the state-dependent coefficient form

$$\begin{aligned} \dot{x} &= A(x)x + B_1(x)w + B_2(x)u \\ z &= C_1(x) + D_{12}(x)u \\ y &= C_2(x) + D_{21}(x)w \end{aligned} \tag{3.29}$$

where $(A,\ B_1)$, $(A,\ B_2)$ and (C_1,A), $(C_2,\ A)$ are stabilizable and detectable, respectively, for $x \in \Omega$, where Ω is the region of interest which may be the entire state space.

2) With γ sufficiently large so that the solutions $P(\hat{x}) \geq 0$, $Q(\hat{x}) \geq 0$ exist with $\lambda_{\max}[P(\hat{x})Q(\hat{x})] < \gamma^2$, solve the SDRE's which are given below in terms of their state-dependent Hamiltonian matrices.

$$\begin{bmatrix} A - B_2 {R_u}^{-1} {D_{12}}^T C_1 & \gamma^{-2} B_1 {B_1}^T - B_2 {R_u}^{-1} {B_2}^T \\ \hat{C_1}^T \hat{C_1} & -(A - B_2 {R_u}^{-1} {D_{12}}^T C_1)^T \end{bmatrix} \tag{3.30}$$

$$\begin{bmatrix} (A - B_1 {D_{21}}^T {R_w}^{-1} C_2)^T & \gamma^{-2} {C^T}_1 C_1 - {C_2}^T {R_w}^{-1} C_2 \\ -\hat{B}_1 \hat{B}_1^T & -(A - B_1 {D_{21}}^T {R_w}^{-1} C_2) \end{bmatrix} \tag{3.31}$$

where

$$\hat{B}_1 = B_1(I - {D_{21}}^T {R_w}^{-1} D_{21}), \qquad \hat{C}_1 = (I - D_{21} {R_u}^{-1} {D_{21}}^T) C_1 \tag{3.32}$$

3) Construct the SDRE H_∞ feedback controller via

$$\begin{aligned} \dot{\hat{x}} &= A_o(\hat{x})\hat{x} + B_o(\hat{x})y \\ u &= F(\hat{x})\hat{x} \end{aligned} \tag{3.33}$$

where

$$
\begin{aligned}
A_o &= A + B_2F + \gamma^{-2}B_1B_1{}^TP + ZL(C_2 + \gamma^{-2}D_{21}B_1{}^TP) \\
B_o &= -ZL \\
Z &= (I - \gamma^{-2}QP)^{-1} \\
F &= -R_u{}^{-1}(B_2{}^TP + D_{12}{}^TC_1) \\
L &= -(QC_2{}^T + B_1D_{21}{}^T)R_w{}^{-1}
\end{aligned}
\tag{3.34}
$$

In case of full state information, the third equation in (3.26) and (3.31) disappear along with the observer equation (3.33) and the static nonlinear controller is given by (3.34) where $\hat{x} = x$. The following theorem from [3] addresses the stability of the SDRE H_∞ control solution.

Theorem 3.1: In addition to all the functions in (3.29) being smooth, assume that $A(x)$ is also smooth. Also assume that the parametrizations (A, B_2) and (C_1, A) are pointwise stabilizable and detectable in the linear sense, respectively, for $x \in \Omega$, where Ω is the region of interest. Then, for the case of full state information, the SDRE H_∞ closed-loop solution is internally locally asymptotically stable.

Proof: See [3].

3.3.4 Simulation Results

Simulation results regarding SDRE are far more abundant than theoretical or experimental ones. Some of these works are as follows: SDRE based designs have been used in advanced guidance law development in [3], attitude control of a satellite in [26]. In [4], SDRE is used to design a guidance law for the midcourse phase of flight of a hypersonic vehicle. During the midcourse phase, the acceleration of the vehicle cannot be controlled effectively by using an acceleration feedback autopilot due to the small acceleration capability of the vehicle in the low dynamic pressure environment. Therefore, two first-order SDRE controllers are designed for the angle of attack and bank angle command inputs.

In [18, 23, 25], the nonlinear benchmark problem proposed by Bupp et al. is solved by SDRE and issues of optimality, robustness, disturbance attenuation are addressed on this specific problem. In the absence of disturbances and uncertainties, the SDRE nonlinear

feedback solution compares very favorably to the open-loop solution obtained via numerical optimization. It is also shown via simulation that the closed-loop system has stability robustness against parametric variations and attenuates sinusoidal disturbances. In the second design, it is demonstrated how a hard bound can be imposed on the control magnitude to avoid actuator saturation. The region of stability of the closed-loop system is also found by extensive simulations.

In [24] the SDRE nonlinear filter (SDREF) is introduced, which is derived by constructing the dual of the SDRE nonlinear regulator. The SDREF has the same structure as the continuous steady-state linear Kalman filter. The behavioral differences between SDREF and LKF (linearized Kalman filter) and EKF (extended Kalman filter), which are based on linearization, are compared using a simple pendulum problem. The theoretical aspects of the filter are not presented since they are yet to be investigated.

In [10] SDREF is used to develop a two-stage state/parameter estimator for simultaneously estimating the state and unknown parameters for systems affine in the unknown parameters. The estimator is used in an example for friction compensation and treatment of unknown bias noise. In [30] the pseudolinear Kalman filter and the continuous discrete (CD) SDRE filter are used for angular rate estimation applied to real data obtained from the RXTE satellite. A comparison between the two reveals that the estimates obtained by the latter is less noisy than those obtained by the former. In [2] the SDREF is used for a thermal rapid processing system.

In [7], SDRE is used for the control of a nonlinear nonminimum-phase continuously stirred tank reactor (CSTR). SDRE can be directly applied to nonminimum-phase systems and hard bounds can be placed on control activity. Simulations indicate that a complicated $Q(x)$ would be necessary to obtain satisfactory response if the unique, symmetric, positive-definite solution of the SDRE, $P(x)$ were used. In order to bypass this complicated choice, the authors use a constant Q and use a nonsymmetric solution of the SDRE, solved in parallel with the symmetry condition (3.19). The stability and robustness analyses of controllers with asymmetric SDRE solutions are not available; but the simulation results in [7] yielded good characteristics in these aspects.

3.3.5 Experimental Results

To the best knowledge of the author**,** the first implementation of SDRE in real-time appears in [20, 21, 22], where the plant is that in the 3rd order nonlinear benchmark problem proposed by Kokotovic et.al. in 1991. In the setup, one 75 MHz Pentium computer simulates the plant dynamics, and another one of the same kind acts as the SDRE controller. The sampling rate of used is 100 Hz. Despite the relatively slow speed of the computers, the results are quite good which shows that the associated computational burden is not that much to rule out SDRE from real-time control applications. In [9], SDRE control is implemented on a physical plant—a 2nd order magnetically levitated ball system—using dSPACE DS1102 digital signal processor together with Matlab and Simulink. The sampling rate used is as high as 20 kHz since the control law is used in its analytical form. In [31], SDRE is used for path control of 2 of the 5 joints of a slow moving robot manipulator. The SDRE gains are calculated offline, based on the manipulator model identified by least squares techniques. Then the gains that are functions of time are sent to the controller at each integration step. The sampling rate is slightly above 1 kHz, which would have been much lower had the SDRE been solved online. Initially the gains are noisy due to the parametrization $f(x)/x$, so they are filtered before implementation. The SDRE results turn out to be satisfactory compared to the standard PID and computed torque control methods.

The main reason why usage of SDRE control in real-time applications is very scarce is that the Algebraic Riccati Equation needs to be solved at each time step, which is computationally very demanding. Ironically, the formulation itself is very suitable for online control, since the gains vary as functions of the states.

Motivated by the promising results of Langson's pioneering work [20, 21, 22], as part of this research, SDRE control was applied in real-time to control a two-link underactuated robot (the Pendubot). A maximum sampling rate of 100 Hz could be achieved using a 60 MHz dSPACE digital signal processing board. This experiment is explained in detail in Chapter 6.

3.4 Open Issues in SDRE Research

- Identification and proof of conditions leading to closed loop stability, estimation of the region of attraction.
- Investigation of real-time implementation issues.
- Analysis of the effects of various state-dependent parametrizations.
- Determining if and under what conditions an analytical solution to the SDRE is possible for systems of order higher than two and finding the relative tradeoffs associated with analytical and numerical solution of SDREs.
- Investigation of the relation between SDRE and optimal nonlinear regulation, finding conditions for optimality of the SDRE based control.
- Examination of correlation with existing proposed nonlinear H_∞ solution techniques
- Determining robustness properties associated with the method.

In this thesis, the first two of these issues are addressed. Chapters 4 and 5 address the first issue. The second issue is addressed in Chapter 6.

3.5 References

[1] S. P. Banks and K. J. Manha, "Optimal Control and Stabilization for Nonlinear Systems", *IMA Journal of Mathematical Control and Information*, vol. 9 pp. 179-196, 1992.

[2] S. Belikov, J. Kamali, B. Friedland, "A State-dependent Riccati Equation filter for a rapid thermal processing system" *Proceedings of the 36th IEEE Conference on Decision and Control,* vol.3, pp.2547-8, 1997.

[3] J. R. Cloutier, "State-Dependent Riccati Equation Techniques: An Overview", *Proceedings of the American Control Conference*, vol.2, pp.932-936, 1997.

[4] J.R Cloutier, C.N. D'Souza and C.P. Mracek, "Nonlinear regulation and Nonlinear H_∞ Control via the State-Dependent Riccati Equation Technique: Part 1, Theory, Part 2, Examples", *Proceedings of the 1st International Conference on Nonlinear Problems in Aviation and Aerospace*, pp.117-141, 1996.

[5] J.R. Cloutier, C. P. Mracek, "Parametric enhancement of State-Dependent Riccati Equation based control", *Proceedings of the American Control Conference*, vol.2, pp.1072-3, 1997.

[6] J. R. Cloutier, D. T. Stansbery, M. Sznaier, "On the recoverability of nonlinear state feedback laws by extended linearization control techniques", *Proceedings of the American Control Conference,* vol.3, pp. 1515-19, 1999.

[7] J. R. Cloutier, D. T. Stansbery, "Control of a continuously stirred tank reactor using an asymmetric solution of the State-Dependent Riccati Equation", *Proceedings of the IEEE International Conference on Control Applications*, vol. 2, pp.893-8, 1999.

[8] J. R. Cloutier, P. H. Zipfel, "Hypersonic guidance via the State-Dependent Riccati Equation control method", *Proceedings of the IEEE International Conference on Control Applications,* vol. 1, pp.219-24, 1999.

[9] E. B. Erdem and A. G. Alleyne, "Globally Stabilizing Second Order Nonlinear Systems by SDRE Control"*, Proceedings of the American Control Conference*, pp. 2501-2505, 1999.

[10] D. Haessig, B. Friedland, "A method for simultaneous state and parameter estimation in nonlinear systems", *Proceedings of the American Control Conference*, vol.2, pp.947-51, 1997.

[11] K. D. Hammett, C. D. Hall, D. B. Ridgely, "Controllability issues in nonlinear the State-Dependent Riccati Equation control", *Journal of Guidance Control & Dynamics,* vol.21, no.5, pp.767-73, Sept.-Oct. 1998.

[12] K. D. Hammett, D. B. Ridgely, "Semiglobal stability of sampled data state feedback SDRE nonlinear regulators", *Proceedings of the American Control Conference*, vol.4, pp.2592-3 vol.4, 1997.

[13] K. D. Hammett, *Control of Nonlinear Systems via State Feedback SDRE Techniques*, Ph.D. thesis, USAF, 1997.

[14] K. D. Hammett, D. B. Ridgely, "Relaxed Conditions for Asymptotic Stability of Nonlinear SDRE Regulators", *Proceedings of the 36th IEEE Conference on Decision and Control,* pp.4012-4017, 1997.

[15] K. D. Hammett, D. B. Ridgely, "Locally stabilizing analytic state feedback controllers for scalar systems via SDRE nonlinear regulation", *Proceedings of the American Control Conference,* vol.2, pp.1070-1, 1997.

[16] M. Hayase, T. Yamazaki, E. Rijanto, "Nonlinear optimal control: principle of local optimality", *Proceedings of IEEE International Conference on Industrial Technology*, vol.1, pp.202-5, Mumbai, India, 2001.

[17] Y. Huang, W-M. Lu, "Nonlinear Optimal Control: Alternatives to Hamilton-Jacobi Equation", *Proceedings of the 35th IEEE Conference on Decision and Control,* pp.3942-3947, 1996.

[18] R. A. Hull, J. R. Cloutier, C. P. Mracek, D. T. Stansbery, "State-Dependent Riccati Equation solution of the toy nonlinear optimal control problem", *Proceedings of the American Control Conference,* vol.3, pp.1658-62, 1998.

[19] Jie Shen, S. N. Balakrishnan, "A class of modified Hopfield networks for control of linear and nonlinear systems", *Proceedings of the American Control Conference*, vol.2, pp.964-9, 1998.

[20] W. Langson, *Optimal and Suboptimal Control of a Class of Nonlinear Systems*, M.S. thesis, University of Illinois at Urbana-Champaign, 1997.

[21] W. Langson, A. G. Alleyne, "Infinite horizon optimal control for a class of nonlinear systems", *Proceedings of the American Control Conference,* vol.5, pp.3017-22, 1997.

[22] W. Langson, A. G. Alleyne, "A Stability Result with Application to Nonlinear Regulation: Theory and Experiments", *Proceedings of the American Control Conference,* pp.3051-3056, 1999.

[23] C. P. Mracek, J. R. Cloutier, "Control designs for the nonlinear benchmark problem via the State-Dependent Riccati Equation method", *International Journal of Robust & Nonlinear Control,* vol.8, no.4-5, pp.401-33, 15-30 April 1998.

[24] C. P. Mracek, J. R. Cloutier, C. A. D'Souza, "A new technique for nonlinear estimation", *Proceedings of the 1996 IEEE International Conference on Control Applications. IEEE International Conference on Control Applications held together with IEEE International Symposium on Intelligent Control, IEEE International Symposium on Computer-Aided Control Systems Design*, *pp.338-43, 1996.*

[25] C. P. Mracek, J. R. Cloutier, "A preliminary control design for the nonlinear benchmark problem", *Proceedings of the 1996 IEEE International Conference on Control Applications. IEEE International Conference on Control Applications held together with IEEE International Symposium on Intelligent Control, IEEE International Symposium on Computer-Aided Control Systems Design*, pp.265-72, 1996.

[26] D. K. Parrish, D. B. Ridgely, “Attitude control of a satellite using the SDRE method”, *Proceedings of the American Control Conference,* vol.2, pp.942-6, 1997.

[27] J. D. Pearson, “Approximation Methods in Optimal Control”, *Journal of Electronics and Control,* vol.13, pp. 453-465, 1962.

[28] M. Sznaier, J. R. Cloutier, R. Hull, D. Jacques, C. Mracek, “A receding horizon state-dependent Riccati equation approach to suboptimal regulation of nonlinear systems”, *Proceedings of the 37th IEEE Conference on Decision and Control,* vol.2, pp.1792-7, 1998.

[29] P. Tsitoras, M. Corless, M. Rotea, “Counterexample to a Recent Result on the Stability of Nonlinear Systems”, *IMA Journal of Mathematical Control & Information,* vol.13, pp. 129-130, 1996.

[30] K. A. Wise, J. L. Sedwick, “Nonlinear control of agile missiles using state-dependent Riccati equations”, *Proceedings of the American Control Conference,* vol.1, pp.379-80, 1997.

[31] M. Innocenti, F. Barolli, F.Salotti, “Manipulator path control using SDRE”. *Proceedings of the American Control Conference,* vol.5, pp.3348-52, 2000.

Chapter 4

Globally Asymptotically Stabilizing Second Order Systems by SDRE Control

In this chapter, global asymptotic stability of second order systems using the State Dependent Riccati Equation (SDRE) method is considered. By a convenient parametrization of the $A(x)$ matrix, the state-dependent algebraic Riccati equation is solved analytically. Thus, the closed-loop system equations are obtained in analytical form. Global stability analysis is performed by the second method of Lyapunov. Accordingly, a relatively simple condition for global asymptotic stability of the closed-loop system is derived. The found results are demonstrated on an experimental magnetic levitation setup.

4.1 Main Result

The main result for global asymptotic stability is stated in the following theorem. The systems considered are of the form (4.3). Any second order nonlinear system that is feedback linearizable can be put in to this form by a change of coordinates [6], [11].

Assumptions:

i) $a_{12}(x)g(x) > 0 \quad \forall x \in S,\ S \subset R^2,$ *where* $a_{12}(x) \equiv f_1(x)/x_2$. (4.1)

ii) $Q(x) = diag(q_1(x), q_2(x))$ *and* $R(x) = r(x)$ *where* $q_1(x) > 0,\ q_2(x) \geq 0,\ r(x) > 0$. (4.2)

Theorem 4.1:

Consider the 2nd order nonlinear system

$$\begin{aligned} \dot{x}_1 &= f_1(x) \\ \dot{x}_2 &= f_2(x) + g(x)u \end{aligned} \qquad x \in S \tag{4.3}$$

where the assumptions given by (4.1) and (4.2) hold. If the forms of $q_1(x)$ and $r(x)$ are chosen such that

$$\frac{a_{12}(x)}{g(x)} = k\sqrt{\frac{q_1(x)}{r(x)}} \qquad k>0,\ k\in R. \tag{4.4}$$

holds $\forall x \in S$, then the SDRE controlled closed-loop system is *locally* asymptotically stable in S. If $S=R^2$, then the SDRE controlled closed-loop system is *globally* asymptotically stable.

Proof:

Let the SDRE parametrization of system (4.3) be made in such a way that $a_{11}(x) = a_{21}(x) = 0$, where $a_{ij}(x)$ is the entry of the matrix $A(x)$ in the i^{th} row and j^{th} column. The convenience of this parametrization is that the Riccati equation yields only quadratic equations to be solved for the elements of $P(x)$, as opposed to quartic, which is the case if $a_{11}(x) \neq 0$.

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \underbrace{\begin{bmatrix}0 & f_1(x)/x_2\\ 0 & f_2(x)/x_2\end{bmatrix}}_{A(x)}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \underbrace{\begin{bmatrix}0\\ g(x)\end{bmatrix}}_{B(x)}u(x) \tag{4.5}$$

At this point, the reader can take for granted that the entries $f_1(x)/x_2$ and $f_2(x)/x_2$ do not cause any division by zero in the control $u(x)$, and thus, in the closed-loop system. Hence, it is safe to proceed with this chosen form of $A(x)$. This will become apparent when the closed-loop system equations are obtained.

Before proceeding further, it should be checked whether the parametrization is pointwise controllable $\forall x$. The controllability matrix is

$$W_{ctrb}(x) = \begin{bmatrix}B(x) & A(x)B(x)\end{bmatrix} = \begin{bmatrix}0 & g(x)f_1(x)/x_2\\ g(x) & g(x)f_2(x)/x_2\end{bmatrix} \tag{4.6}$$

The condition to be satisfied for pointwise controllability is

$$\det(W_{ctrb}(x)) = -g^2(x)\frac{f_1(x)}{x_2} = -g(x).(g(x)a_{12}(x)) \neq 0 \quad \forall x \tag{4.7}$$

Since it was assumed that $\text{sgn}(a_{12}(x)g(x)) = 1 \ \forall x$ and since $g(x) \neq 0 \ \forall x$, it follows that this parametrization is pointwise controllable $\forall x$. Now it can be proceeded with the stability analysis.

The matrix $P(x)$ is of the form

$$P(x) = \begin{bmatrix} p_{11}(x) & p_{12}(x) \\ p_{12}(x) & p_{22}(x) \end{bmatrix} \tag{4.8}$$

since it is symmetric. As such, the state-dependent algebraic Riccati equation yields the following three scalar equations for $p_{11}(x)$, $p_{12}(x)$ and p_{22}. For ease of notation, the x-dependence of the variables is omitted in the sequel.

$$q_1 - \frac{1}{r} p_{12}^2 g^2 = 0 \tag{4.9}$$

$$a_{12} p_{11} + a_{22} p_{12} - \frac{1}{r} p_{12} g^2 p_{22} = 0 \tag{4.10}$$

$$2a_{12} p_{12} + 2a_{22} p_{22} + q_2 - \frac{1}{r} p_{22}^2 g^2 = 0 \tag{4.11}$$

from which p_{ij} are solved as

$$p_{11} = \frac{\nu \sqrt{q_1 r} \sqrt{a_{22}^2 + q_2 g^2 / r + 2a_{12} g \sqrt{q_1 / r}}}{a_{12} g} \tag{4.12}$$

$$p_{12} = \sqrt{q_1 r} / g \tag{4.13}$$

$$p_{22} = \frac{a_{22} + \nu \sqrt{a_{22}^2 + q_2 g^2 / r + 2a_{12} g \sqrt{q_1 / r}}}{g^2 / r} \tag{4.14}$$

By imposing $p_{11}(x) > 0 \ \forall x$ as one of the two requirements (the other being $\det(P(x)) > 0 \ \forall x$) for positive definiteness of $P(x)$, one gets the imposition that

$$\nu = \text{sgn}(a_{12}(x)g(x)) = 1 \quad \forall x \tag{4.15}$$

The feedback control is obtained as

$$u(x) = -\frac{g}{r}(p_{12}x_1 + p_{22}x_2) = -\sqrt{\frac{q_1}{r}}x_1 - \frac{a_{22}x_2}{g} - \frac{x_2}{g}\sqrt{a_{22}^2 + q_2 g^2 / r + 2a_{12}g\sqrt{q_1 / r}} \tag{4.16}$$

By recalling that $a_{12}x_2 = f_1$ and $a_{22}x_2 = f_2$, (4.16) can be rewritten as

$$u(x) = -\sqrt{\frac{q_1}{r}}x_1 - \frac{f_2}{g} - \frac{\text{sgn}(x_2)}{g}\sqrt{f_2^2 + q_2 x_2^2 g^2 / r + 2f_1 x_2 g\sqrt{q_1 / r}} \tag{4.17}$$

which, as can be seen, involves no division by zero. Substituting the control above into the system equations (4.3), the closed-loop equations become,

$$\dot{x}_1 = a_{12}x_2 = f_1(x) \tag{4.18}$$

$$\dot{x}_2 = -g\sqrt{\frac{q_1}{r}}x_1 - \text{sgn}(x_2)\sqrt{f_2^2 + q_2 x_2^2 g^2 / r + 2f_1 x_2 g\sqrt{q_1 / r}}$$

$$= -(g\sqrt{\frac{q_1}{r}})x_1 - (\sqrt{a_{22}^2 + q_2 g^2 / r + 2a_{12}g\sqrt{q_1 / r}})x_2$$

The second method of Lyapunov will be used for stability analysis. Using a common quadratic Lyapunov function such as

$$V(x) = (\alpha x_1^2 + \delta x_2^2)/2, \qquad \alpha, \delta > 0 \tag{4.19}$$

with the closed-loop system equations (4.18), the Lyapunov function derivative is obtained as

$$\dot{V} = (\alpha a_{12} - \delta g\sqrt{\frac{q_1}{r}})x_1 x_2 - \delta\sqrt{a_{22}^2 + q_2 g^2 / r + 2a_{12}g\sqrt{q_1 / r}}\,x_2^2 \tag{4.20}$$

The stability requirement that $\dot{V}(x)$ be negative definite $\forall x$ can be relaxed by noticing that a negative semi-definite $\dot{V}(x)$ in the form $\dot{V}(x) = -m(x)x_2^2$, $m(x)>0$ $\forall x$ will suffice to achieve global asymptotic stability. Taking $\dot{V}(x) = -m(x)x_2^2$ and considering the closed-loop equations (4.18), it can easily be seen that

$$\dot{V}(x) = 0 \ \forall x \ \Rightarrow x_2(t) \equiv 0 \Rightarrow x_1(t) \equiv 0 \tag{4.21}$$

Thus, $\dot{V}(x) = 0$ only at the origin. It follows from LaSalle's Invariance Theorem [6] that a negative semi-definite $\dot{V}(x)$ of the assumed form suffices for global asymptotic stability. Then, for the cross-product term in (4.20) to vanish, the following condition should be satisfied $\forall x$:

$$\alpha a_{12}(x) - \delta g(x)\sqrt{\frac{q_1(x)}{r(x)}} = 0 \tag{4.22}$$

By defining $k \equiv \delta/\alpha > 0$, (4.22) yields the global asymptotic stability condition (4.4) of Theorem 1. ♦

Remarks:

i) In the general case when a_{12} and g are functions of x, the choice of $q_1(x)$ and $r(x)$ are dependent on each other.

ii) When a_{12} and g are constants, the closed-loop system can be globally asymptotically stabilized by independent, constant choices of q_1 and r, as long as they are positive. This means that systems of the form

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= f_2(x) + u \end{aligned} \tag{4.23}$$

can be globally asymptotically stabilized by tuning q_1, q_2 and r as arbitrary constants since condition (4.4) reduces to simply

$$1 = k\sqrt{q_1 / r}\,, \qquad k>0. \tag{4.24}$$

iii) The condition for global asymptotic stability (4.4) is independent of the choice of $q_2(x)$. Thus, it can be chosen arbitrarily as long as $q_2(x) \geq 0 \ \forall x$.

Example 4.1:

This example is to illustrate the second remark. The system in this example is from [7]. In the reference paper, the authors demonstrate that when using linear high gain feedback control, as the gains are increased, the stability region shrinks along the *x_1-axis* and is confined to the *x_2-axis* in the extreme case when the gain goes to infinity. Here we show that it is possible to globally asymptotically stabilize this system with the SDRE method. The nonlinear system equations are

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u + \frac{1}{3}x_2^3 \end{aligned} \tag{4.25}$$

The obvious SDRE parametrization for this system is $a_{12}(x) = 1$, $a_{22}(x) = x_2^2/3$ and *g(x)* = 1. It can be easily shown that det($W_{\text{ctrb}}(x)$)=1 $\forall x$, thus it is pointwise stabilizable. Condition (4.4) for global asymptotic stability requires that

$$1 = k\sqrt{q_1(x) / r(x)}\,, \ \forall x, \qquad k>0. \tag{4.26}$$

which can be easily achieved by an infinite number of choices of $q_1(x)$ and $r(x)$, as long as they are positive and have the same functional form. The simplest choice would be to choose them as constants. In this example, they were simply chosen as $q_1(x) = r(x) = 1$. The control resulting from this choice (with $q_2(x) = 1$) is

$$u(x) = -x_1 - \frac{1}{3}x_2^3 - x_2\sqrt{\frac{27 + x_2^4}{9}} \tag{4.27}$$

and the closed-loop system equations become

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -x_1 - x_2\sqrt{\frac{27 + x_2^4}{9}} \tag{4.28}$$

If the Lyapunov function is chosen as

$$V(x) = (x_1^2 + x_2^2)/2 \tag{4.29}$$

its derivative becomes

$$\dot{V}(x) = -x_2^2\sqrt{\frac{27 + x_2^4}{9}} \tag{4.30}$$

which is negative semi-definite. However, noticing that $\dot{V}(x) = 0 \ \forall x \Rightarrow x_2(t) \equiv 0 \Rightarrow x_1(t) \equiv 0$ and applying LaSalle's Theorem, it can be easily seen that the controlled system is globally asymptotically stable.

4.2 Comparison with Feedback Linearization

Second order systems of the form (4.3) are always feedback linearizable. In this section, the advantages of SDRE over feedback linearization for this class of systems are illustrated. The system equations (4.3) can be put in a linear form by a coordinate transformation

$$z_1 = x_1 , \qquad z_2 = f_1 \tag{4.31}$$

Then the equations in the new coordinates (z_1, z_2) become

$$\dot{z}_1 = z_2$$
$$\dot{z}_2 = \frac{\partial f_1}{\partial x_1} f_1 + \frac{\partial f_1}{\partial x_2}(f_2 + gu) = \nu \tag{4.32}$$

The "synthetic input" ν is constructed in a state feedback form, such as

$$\nu = -k_1 z_1 - k_2 z_2 \tag{4.33}$$

Then the actual control input becomes equal to

$$u = \frac{1}{g\dfrac{\partial f_1}{\partial x_1}}\left[\nu - \frac{\partial f_1}{\partial x_1} f_1 - \frac{\partial f_1}{\partial x_2} f_2\right] \tag{4.34}$$

Inspecting the control given by (4.34), it can be seen that when the denominator nears zero, the input u becomes increasingly large, which may cause peaking and/or actuator saturation. The designer has no control over this phenomenon except for possibly choosing large gains k_1 and k_2 in the synthetic control, ν. However, high gain feedback is not a very safe remedy because, as mentioned in the above example, it may cause drastic shrinkage of the stability region. As another case, if the nonlinearities in the numerator of (4.34) are too large, then the synthetic control ν required to cancel these nonlinearities may be too large, which may also cause actuator saturation.

Comparing this control law by the can one obtained by SDRE, (4.17), it can be seen that in the SDRE control formulation, there is no danger of "division by zero", since $g(x) \neq 0$ for all $x \neq 0$. Moreover, excessive control inputs can be avoided by making use of the three design parameters q_1, q_2 and r. The following example illustrates this.

Example 4.2: Consider the following system

$$\begin{aligned} \dot{x}_1 &= \sin x_2 \\ \dot{x}_2 &= -a x_1^2 + x_1 x_2 + \frac{b}{(c + x_1)^2} u \end{aligned} \tag{4.35}$$

where $\{a, b, c\} \in \mathrm{R}$ $x_1 \neq c$ and $x_2 \in (\pi, -\pi)$. This system is feedback linearizable. By using the coordinate transformation $z_1 = x_1$ and $z_2 = \sin x_2$, and carrying out the procedure given by (4.32-34), the control law obtained by feedback linearization becomes

$$u_{FL} = \frac{(c+x_1)^2}{b\cos x_2}\left[(ax_1\cos x_2 - k_1 - x_2\cos x_2)x_1 - k_2\sin x_2\right] \tag{4.36}$$

It can be seen that when $\cos x_2=0$ at $x_2=\pm\pi/2$, the denominator becomes zero and the control becomes infinite. This cannot be avoided by any nonzero choice of the designer parameters, k_1 and k_2. ($k_1=k_2=0$ corresponds to no control).

In order for the SDRE control to satisfy the asymptotic stability condition (4.4), let q_1, q_2 and r be chosen as

$$q_1(x) = k_{q1}(\sin x_2 / x_2)^2, \qquad q_2\text{:constant}, \qquad r(x) = \frac{k_r}{(c+x_1)^4} \tag{4.37}$$

where k_{q1} and k_r are multiplicative constants. Then the SDRE control for this system obtained by (4.17) is

$$u_{SDRE} = -\frac{\sin x_2 / x_2}{(c+x_1^2)}x_1\sqrt{\frac{k_{q1}}{k_r}} + \frac{(c+x_1^2)}{b}(ax_1^2 - x_1x_2 - \mathrm{sgn}(x_2)\sqrt{\Gamma(x)}) \tag{4.38}$$

where

$$\Gamma(x) = (-ax_1^2 + x_1x_2)^2 + q_2x_2^2\frac{b^4}{k_r} + \frac{2b\sin^2 x_2}{(c+x_1)^4}\sqrt{\frac{k_{q1}}{k_r}} \tag{4.39}$$

It is seen that the SDRE control does not involve any division by zero, thus remains finite $\forall x$. Figures 4.1 and 4.2 show simulations of the two controllers with the system parameter values a=0.2, b=1.7, c=3.2. The design parameters for F/L (feedback linearization) were chosen as $k_1=k_2=1$, and those for SDRE were chosen as $k_{q1}=1$, $k_r=10$, $q_2=1$. Figure 4.1 shows the system simulated from the initial conditions $x_0=[\pi/4, 0.2]$. It is seen that the F/L response is more oscillatory than the SDRE response, and the F/L control input is larger than the SDRE input. In Figure 4.2, the system is simulated from the initial conditions $x_0=[\pi/8, -1.6]$. It is seen that

while the SDRE controlled system converges to the origin, the F/L controlled system does not. Moreover, the control input required by F/L is larger than the SDRE control input.

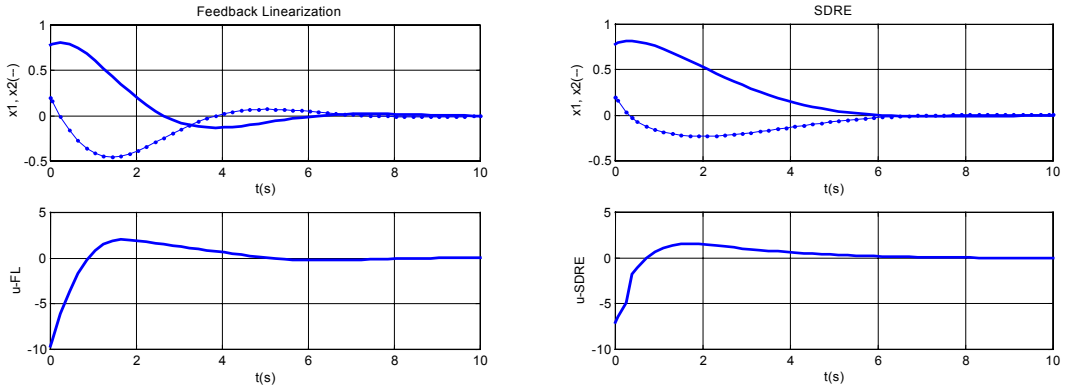


Figure 4.1. F/L and SDRE control with initial conditions x_0=[π/4, 0.2].

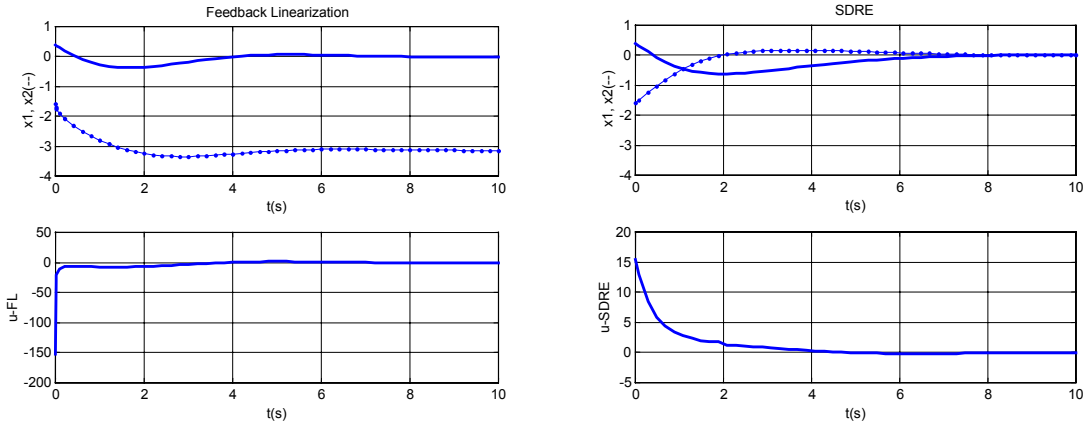


Figure 4.2. F/L and SDRE control with initial conditions x_0=[π/8, -1.6].

4.3 Experimental Magnetic Levitation Example

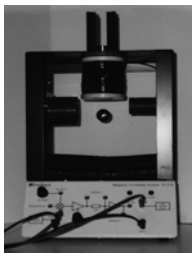


Figure 4.3 The Maglev

The testbed of this experiment is the magnetically levitated ball shown in Figure 4.3 [11]. The aim is to regulate the ball at a desired position by using the force supplied by the magnet at the top. The displacement y is measured negative downward, and y=0 at the top. Denoting displacement of the ball as y_1, velocity by y_2, and the control input by U, the system equations are

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= -g + \frac{K}{(\alpha + y_1)^2} U \end{aligned} \tag{4.40}$$

where g=9.81 m/s^2, K=5.695e–4 m^3/s^2, α =1.234e-2 m. It should be emphasized that these parameters are not constant in time, and actually change due to thermal effects. Thus, there are parameter uncertainties involved in the system model. In (4.40), the equilibrium values of U and y_1 are nonzero. To apply the SDRE formulation, the equations must be rewritten such that both states are zero at the equilibrium, as well as the value of the input. With y_e denoting the equilibrium position, a change of coordinates can be made such that

$$y_1 = y_e + x_1 \text{ and } U = u_e + u \tag{4.41}$$

where $u_e = g(\alpha + y_e)^2 / K$ is the input needed to hold the ball at equilibrium. $u=u(x)$ is the additional effort needed to steer the ball from some x to the equilibrium position. The state equations rewritten are:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{-gx_1(2\alpha + 2y_e + x_1)}{(\alpha + y_e + x_1)^2} + \frac{K}{(\alpha + y_e + x_1)^2} u \end{aligned} \tag{4.42}$$

which is in the standard form (4.3). The system is pointwise controllable $\forall x$, since $\operatorname{sgn}(a_{12}(x)g(x)) = 1$ (4.1) is satisfied. The control input is constructed according to (4.17) and the weighting parameters are tuned according to (4.4). The latter condition requires that

$$1 = k\sqrt{\frac{q_1(x)g^2(x)}{r(x)}} \tag{4.43}$$

be satisfied $\forall x$, k being any positive constant. One way to satisfy condition (4.43) is to choose r as constant and to assume a form for $q_1(x)$ which is proportional to the inverse of $g^2(x)$. This way, the position error x_1 will be penalized more when it is large and less when it is small.

The experiment was carried out in real-time using dSPACE DSP system [3], [10] with a sampling rate of 10 kHz. The equilibrium signal $y_e(t)$ was chosen as a square wave of amplitude 1 mm and period 4s. This corresponds to regulation to the origin of (4.42) from an initial displacement of ±2mm. $y_e(t)$ and the weighting parameters were chosen as

$$y_e(t) = -0.022 + 0.001 square(\pi/2)t \text{ , (m)}$$

$$q_1(x) = k_{q1}\frac{(\alpha + ye + x_1)^4}{K^2} \text{ (m}^{-2}\text{)}, \quad k_{q1} \text{ constant}$$

q_2=8500 (m/s)$^{-2}$, r=8.5x10^6 (dimensionless) (4.44)

The results of three experiments made by holding r and q_2 constant while changing q_1 are shown in Figures 4.4-4.6. $q_1(x)$ is changed by tuning the multiplicative constant, k_{q1}. The values used for k_{q1} were 1250, 1600 and 2000. The variations of u_{eq}, u_{tot} and $k_1(x)$ with time are shown, where $u_{tot}(x) = u_{eq} + u_1(x) + u_2(x)$, $u_1(x) = -k_1(x_1, x_2)x_1$ and $u_2(x) = -k_2(x_1, x_2)x_2$. The variation of $k_2(x)$ is similar to that of $k_1(x)$. Figure 4.4 shows the step responses for the system with relatively low weighting, k_{q1} on the position regulation error. As can be seen in the figure, the system response is relatively slow in achieving equilibrium and a significant steady state error exists. Figure 4.5 demonstrates the effect of simply increasing the weighting factor on the position regulation error while keeping all other control parameters the same as in Figure 4.4. The figure clearly shows the effect of the increased weighting. The system transient response is much faster with a smaller steady-state error. This is consistent with an intuitive understanding of the weighting effects. Figure 4.6 demonstrates the effect of a further increase in k_{q1}. On the axis scale shown, the response is

similar to that of Figure 4.5 with the exception of the increased overshoot in the system response. By increasing the weight on the regulation error, a slight increase in bandwidth was obtained, albeit with an associated decrease in system 'damping'. Increasing the weight on the mass velocity would easily alleviate the overshoot shown in Figure 4.6. In a setting where inevitable disturbances and parameter uncertainties were involved, very good regulation performance could be achieved by SDRE, despite the fact that the square wave signal is a difficult one for such an unstable system to follow. In the best response, the steady state position error is less than 200 micrometers. The residual error, as mentioned before, is largely due to model error in (4.42). The parameters are difficult to know exactly and actually change over time due to thermal effects. One of the main advantages of SDRE control can be observed in this experimental example: although the system is nonlinear, the desired performance can be obtained simply by tuning some constant design parameters, as in the linear case.

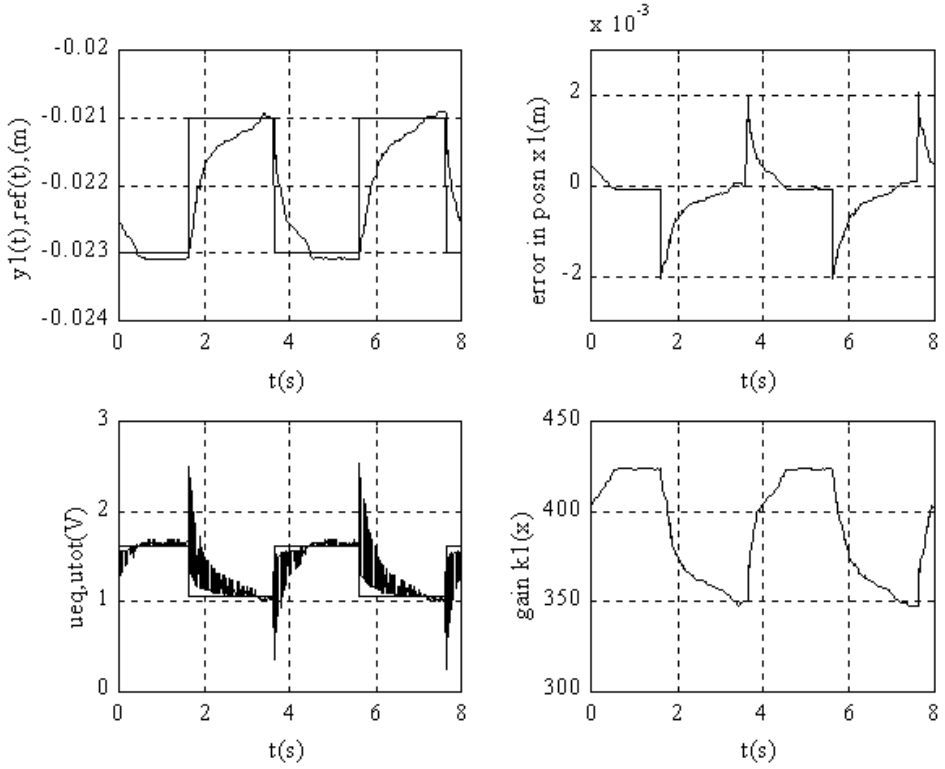


Figure 4.4 k_{q1}=1250, q_2=8500, r=8.5x10^6

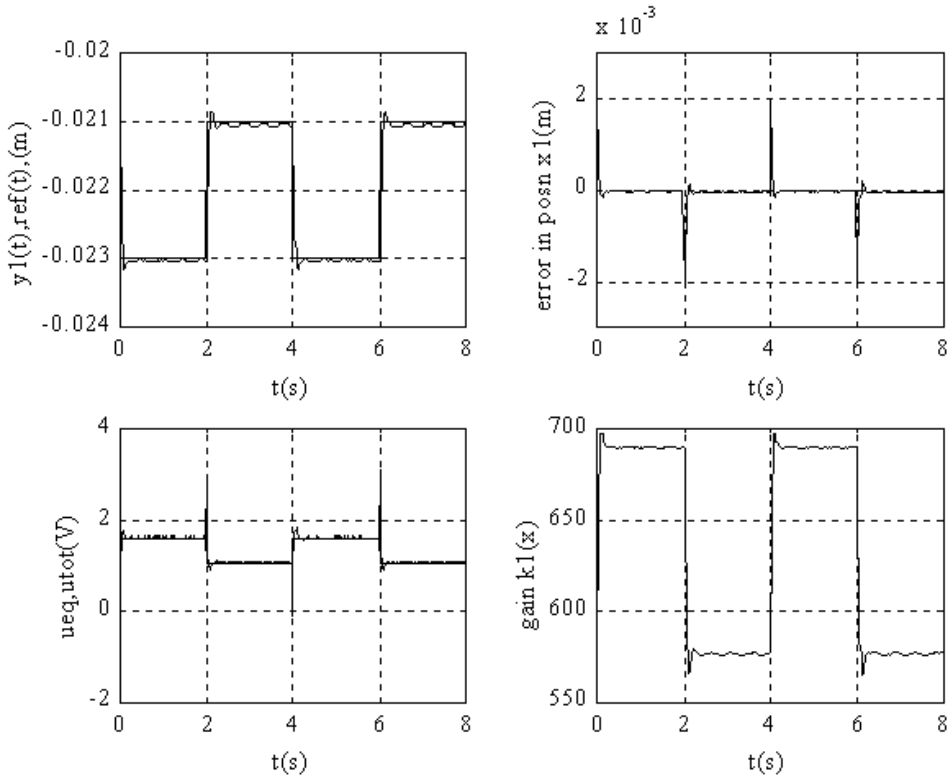


Figure 4.5 k_{q1}=1600, q_2=8500, r=8.5x10^6

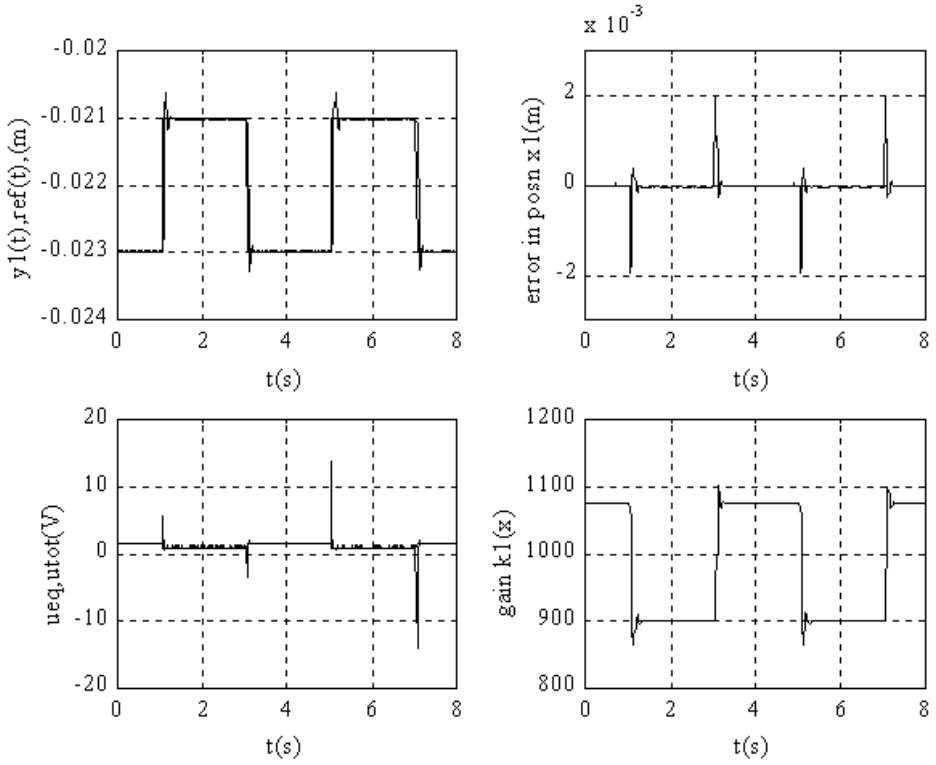


Figure 4.6 k_{q1}=2000, q_2=8500, r=8.5x10^6

4.4 References

[1] J. R. Cloutier, "State-Dependent Riccati Equation Techniques: An Overview", *Proceedings of the American Control Conference*, vol.2, pp.932-936, 1997.

[2] J.R Cloutier, C.N. D'Souza and C.P. Mracek, "Nonlinear regulation and Nonlinear H_∞ Control via the State-Dependent Ricatti Equation Technique: Part 1, Theory, Part 2, Examples", *Proceedings of the 1st International Conference on Nonlinear Problems in Aviation and Aerospace*, pp.117-141, 1996.

[3] E. B. Erdem and A. G. Alleyne, "Globally Stabilizing Second Order Nonlinear Systems by SDRE Control", *Proceedings of the American Control Conference*, pp. 2501-2505, 1999.

[4] K.D. Hammett, *Control of Nonlinear Systems via State Feedback SDRE Techniques,* Ph.D. thesis, USAF, 1997.

[5] Y. Huang, W-M Lu, "Nonlinear Optimal Control: Alternatives to Hamilton-Jacobi Equation", *Proceedings of the 35th Conference of Decision and Control*, vol. 4, pp.3942-3947, 1996.

[6] H. K. Khalil, *Nonlinear Systems,* Macmillan, 1992.

[7] P. Kokotovic, R. Marino, "On Vanishing Stability Regions in Nonlinear Systems with High Gain Feedback", *IEEE Transactions on Automatic Control, 31(10)* : pp.967-970, 1986.

[8] W. Langson, *Optimal and Suboptimal Control of a Class of Nonlinear Systems,* M.S. thesis, University of Illinois at Urbana-Champaign, 1997.

[9] W. Langson and A. G. Alleyne, "A Stability Result with Application to Nonlinear Regulation: Theory and Experiments", *Proceedings of the American Control Conference*, vol.4, pp. 3051-3056, 1999.

[10] C.P. Mracek and J.R. Cloutier, "Control Designs for the Nonlinear Benchmark Problem via the State-Dependent Riccati Equation Method", *Int. J. of Robust and Nonlinear Control*, vol.8, no.4/5, pp.401-433, 1998.

[11] M.A. Pomykalski, *Control of a Class of Nonlinear Systems Subject to Periodic, Exogenous Signals,* M.S. thesis, University of Illinois at Urbana-Champaign, 1998.

Chapter 5

Stability of SDRE Controlled Systems Using Vector Norms

The origin of an SDRE controlled system is an asymptotically stable equilibrium point. However, knowledge of its region of stability is of practical importance. Many methods have been proposed for estimating the region of attraction (ROA) of nonlinear systems [4]. Some of these, including Zubov's method, are Lyapunov-based while others are based on topological considerations and trajectory reversing.

Obtaining a good estimate of the ROA for general nonlinear systems—especially of higher order—is a challenging task in itself. For SDRE controlled systems, the task becomes even more difficult since the closed-loop matrix is usually not available in closed form.

About a decade ago, a new method for determining the stability of nonlinear systems was proposed by Borne, Richard et.al.[1, 2, 3, 5]. The method is based on defining an overvaluing system for the original one by using vector norms. In this chapter, an approach to estimate stability regions of SDRE controlled systems using this method is developed.

This chapter is organized as follows. First, basic concepts of vector norms and comparison systems are introduced. Theorems to determine global/local asymptotic stability and estimate the ROA are presented. Next, the application of this method to SDRE controlled systems is developed. Examples are given to illustrate the use of the method.

5.1 Overvaluing Systems of a Continuous System

The class of systems considered are general nonlinear time-varying systems described by the vector differential equation

$$\dot{x} = f(t, x) = A(t, x)x \tag{5.1}$$

It is supposed that x=0 is the unique equilibrium and adequate smoothness conditions are satisfied to assure existence of solutions. The overvaluing system will be defined by the use of the vector norm $p(x)$ of the state vector x. First, the notion of the time derivative of a vector

norm will be used. For this purpose it is necessary to use the right-hand derivative $D^+ p_i(x_i)$ [2]. $D^+ p_i(x_i)$ is taken along the motion of (5.1) in the subspace E_i and $D^+ p(x)$ along the motion of (5.1) in E, where $E_i \subset E$.

Definition 5.1: *Overvaluing system* [1, 3, 5]

The matrix $M : \tau_o x R^n \to R^{kxk}$ defines an overvaluing system of (5.1) with respect to the vector norm *p* if and only if the following inequality is verified for each corresponding component:

$$D^+ p(x) \le M(t,x)p(x) \qquad \forall x \in R^n, \forall t \in \tau_o \tag{5.2}$$

Let us denote

$$A_{ij}(t,x) = P_i A(t,x) P_j , \qquad \forall i, j = 1,2,...,k , \tag{5.3}$$

$$M(t,x) = \{\mu_{ij}(t,x)\}, \tag{5.4}$$

$$m_{ij}(t,x,y) = \frac{grad[p_i(y_i)^T A_{ij}(t,x) y_i]}{p_i(y_i)} \tag{5.5}$$

Definition 5.2: *Pseudo-overvaluing matrix* [1, 3, 5]

$$M(t,x) = \{\mu_{ij}(t,x)\}, \qquad \mu_{ij} : \tau_o x R^n \to R$$

is a pseudo-overvaluing matrix if:

$$\mu_{ii}(t,x) = \sup_{y \in E}\{m_{ii}(t,x,y)\}, \qquad \forall i = 1,2,...,k , \ \forall x \in E, \forall t \in \tau_o \tag{5.6}$$

$$\mu_{ij}(t,x) = \max(0, \sup_{y \in E}\{m_{ii}(t,x,y)\}), \qquad \forall i \ne j = 1,2,...,k , \ \forall x \in E, \forall t \in \tau_o \tag{5.7}$$

5.1.1 Application to Usual Norms

Let a partition of the space E define a block partition of a matrix A. I_i and I_j represent the sets of indices of rows and columns, respectively, of block A_{ij}. If $p_i(x_i)$ is the 'max' norm (maximum of the modulus of each component of x_i), then:

$$\mu_{ii}(t,x) = \max_{s\in I_i}[a_{ss} + \sum_{\substack{l\in I_i \\ l\neq s}} |\, a_{sl}\,|] \tag{5.8}$$

$$\mu_{ij}(t,x) = \max_{s\in I_i}[\sum_{l\in I_j} |\, a_{sl}\,|] \tag{5.9}$$

A simple interpretation of these formulae is possible: first, all off-diagonal elements of A are replaced by their absolute value, leading to an overvaluing matrix A^*. Then, μ_{ij} is the greatest sum of all components in each row of the block A^*_{ij} of A^*.

5.1.2 Properties of Pseudo-overvaluing Matrices

When a pseudo-overvaluing matrix $M(t,x)$ of $A(t,x)$ is defined with respect to a regular vector norm p, the following properties hold:

(a) The off-diagonal elements $\mu_{ij}(t,x)$, $(i \neq j)$, of matrix $M(t,x)$ are nonnegative.

(b) If $\lambda_A(t,x)$ denotes any of the n eigenvalues of $A(t,x)$, $\lambda_M(t,x)$ any of the k eigenvalues of $M(t,x)$ and $\lambda_m(t,x)$ the $\lambda_M(t,x)$ which represents the maximal real part, then the following holds:

$$\mathrm{Re}(\lambda_A(t,x)) \leq \mathrm{Re}(\lambda_M(t,x)) \leq \lambda_m(t,x) \in R, \qquad \forall x \in E, \forall t \in \tau_o \tag{5.10}$$

$\lambda_m(t,x)$ is called the importance eigenvalue of $M(t,x)$.

(c) When all the real parts of the $\lambda_M(t,x)$ are negative, the matrix $M(t,x)$ is the *opposite of an M-matrix*, i.e. it admits an inverse whose elements are all nonpositive.

(d) The pseudo-overvaluing matrix whose elements are defined from (5.4-5.7) is the smallest pseudo-overvaluing matrix of $A(t,x)$ elementwise with respect to the vector norm p.

(e) $M(t,x)$ verifies (c) iff the *Koteliansky conditions* are verified, i.e. its successive principal minors are sign alternate:

$$(-1)^i M(t,x)\begin{bmatrix}1,...,i\\1,...,i\end{bmatrix} > 0\,, \qquad \forall i = 1,2,...,k\,,\ \forall x \in E, \forall t \in \tau_o \tag{5.11}$$

(f) When the matrix $M(t,x)$ is the opposite of an *M*-matrix (or equivalently verifies (c) or (e)), and when its inverse is irreducible, then $M(t,x)$ admits an eigenvector $u_m(t,x)$, called the *importance vector* of $M(t,x)$, whose components are strictly positive and which is related to the real, maximal and negative eigenvalue $\lambda_m(t,x)$.

<u>*Remark:*</u> When the matrix $M(t,x)$ does not admit an irreducible inverse matrix, one can use another pseudo-overvaluing matrix deduced from $M(t,x)$ by adding a constant matrix B^* with $B^*=\{b_{ij}\}$:

$b_{ii}^* = 0\,,\ \ \forall i = 1,2,...,k$

$b_{ij}^* \geq 0\,,\ \ \forall i \neq j = 1,2,...,k$

5.2 Stability Study and Estimation of Stability Regions

<u>Comparison Lemma:</u>

Let there exist a vector norm p and a matrix $M(t,x)$ connected with (5.1) such that the off-diagonal elements of $M(t,x)$ are all nonnegative and that the following inequality is verified along the solution of (5.1):

$$D^+ p(x) \leq M(t,x)p(x) \qquad \forall x \in R^n, \forall t \in \tau_o \tag{5.2}$$

Then the system

$$\dot{z} = M(t,x)z\,, \qquad \forall z \in R_+^k \tag{5.12}$$

is a comparison system of (5.1) in the sense that $z(t) \geq p(x(t))$, $\forall t \in \tau_o$ holds as soon as $z(t_0) \geq p(x_0)$.

It is obvious from this lemma that the stability of a comparison system deduced from (5.1) implies the stability of the initial system (5.1).

Theorem 5.1: *Global asymptotic stability of an equilibrium point with a constant M-matrix* [1, 2, 3, 5]

If it is possible to define a pseudo-overvaluing system for (5.1) on R^n relative to a regular vector norm p:

$$D^+ p(x) \leq Mp(x) \qquad \forall x \in R^n \tag{5.13}$$

for which the constant matrix M is Hurwitz or, in the same way, is the opposite of an M-matrix, or even verifies the Koteliansky conditions, then x=0 is globally asymptotically stable for (5.1). If, moreover, $M + \varepsilon I_k$ (with ε>0) is stable, then x=0 is globally exponentially stable in $e^{-\varepsilon t}$.

Proof: Can be found in [5].

Expressed formally, exponential stability in $e^{-\varepsilon t}$ means that there exist two positive numbers, α and λ such that for sufficiently small $x(t_0)$,

$$\|x(t)\| \leq \alpha \|x_0\| e^{-\lambda(t-t_0)}\,, \quad \forall t > t_0 \tag{Δ}$$

In words, (Δ) means that the state vector x converges to the origin faster than an exponential function.

Example 5.1: Consider the following system [2]:

$$\dot{x} = A(t,x)x\ ,\ x = [x_1, x_2, x_3, x_4]^T, \qquad w \in [-0.5,1]$$

$$A(t,x) = \begin{bmatrix} -2 & \sin t & 0.2 & 0.7w \\ -0.5 & -1.5 & -0.4 & 0.6sat(x_3) \\ -0.1\,\mathrm{sgn}(x_1) & 0.3 & -1.3 & 0.3 \\ 0.1 & 0.4 & -1.1 & -2 \end{bmatrix} \tag{5.14}$$

The convergence of the trajectories towards the equilibrium x=0 will be studied through the convergence of the vector norm:

$$p(x) = [\max\{|x_1|,|x_2|\}, \max\{|x_3|,|x_4|\}]^T = [p_1(x), p_2(x)]^T \tag{5.15}$$

If D^+ designates the right-hand derivative operator, one can compute:

$$D^+ p_1(x) \le \max\{D^+|x_1|, D^+|x_2|\} \tag{5.16}$$

$$D^+|x_1| \le -2|x_1| + |\sin t||x_2| + |0.2||x_3| + |0.7w||x_4| \tag{5.17}$$

$$D^+|x_2| \le |-0.5||x_1| - 1.5|x_2| + |-0.4||x_3| + |0.6sat(x_3)||x_4| \tag{5.18}$$

Overvaluing |sin *t*|, |*w*|, and |*sat* (x_3)| by 1, the following is obtained:

$$D^+ p_1(x) \le \max\{-2 + |\sin t|, |0.5| - 1.5\} p_1(x) + \max\{|0.2| + |0.7w|, |-0.4| + |0.6sat(x_3)|\} p_2(x) \tag{5.19}$$

$$D^+ p_1(x) \le -p_1(x) + p_2(x) \tag{5.20}$$

Similarly for $p_2(x)$

$$D^+ p_2(x) \le 0.5 p_1(x) - 0.9 p_2(x) \tag{5.21}$$

In matrix writing,

$$D^{+} p(x) \le Mp(x), \qquad M = \begin{bmatrix} -1 & 1 \\ 0.5 & -0.9 \end{bmatrix} \tag{5.22}$$

M is a Hurwitz matrix with eigenvalues –0.241 and –1.659. It follows from Theorem 5.1 that x=0 is globally asymptotically stable. Moreover, by the importance eigenvalue of M, exponentiality of stability is ensured in $e^{-0.241t}$.

Remarks:

(i) If the vector $p(x)$ is of dimension 1, then this is like the second method of Lyapunov. If $p(x)$ is of higher dimension, then it is dealt with a vector-Lyapunov function.

(ii) Reduction of complexity: The matrix M, of order 2, allows the study of a system which is initially of order 4.

(iii) The comparison system M is linear time-invariant for an initially nonlinear and time-varying system.

(iv) Overvaluing is a loss of information. The more it is used, the less the obtained model is likely to be convergent.

Theorem 5.2: *Local asymptotic stability of an equilibrium point with a constant M-matrix matrix* [1, 2, 3, 5]

If it is possible to define a pseudo-overvaluing system for (5.1) in a neighborhood $S \subset R^n$ of the equilibrium point x=0 relative to a regular vector norm p:

$$D^{+} p(x) \le Mp(x) \qquad \forall t \in \tau_o, \ \forall x \in S \tag{5.23}$$

for which the constant matrix M is irreducible and of Hurwitz type, then x=0 is locally asymptotically stable, and its region of attraction (ROA) includes the largest domain D which is strictly included in S and defined by one of the following types of sets:

1. $D = D_1 = \{x \in R^n;\ p^T(x) u_m(M^T) \le \alpha \text{ (constant)}\} \subset S$,

2. $D = D_\infty = \{x \in R^n;\ p(x) \le \alpha u_m(M),\ \text{(constant scalar)}\} \subset S$,

3. $D = D_c = \{x \in R^n;\ p(x) \le c,\ \ c \in R_+^k,\ \ -Mc > 0\} \subset S$,

in which $u_m(A)$ denotes a positive importance vector of an *M*-matrix *A*, associated with its importance eigenvalue $\lambda_m(A)$ (<0). Moreover, the convergence in this domain is exponential in $e^{\lambda_m(M)t}$. If *S*=R^n, the property is global (Theorem 5.1).

Proof: Can be found in [5].

Example 5.2: [2] Consider a system $\dot{x} = A(x)x$ with

$$A(x) = \begin{bmatrix} -3 + x_1^2 & x_2 \sin x_2 \\ 4x_1^2 \cos x_2 & -3 + x_2^2 \end{bmatrix},\ x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{5.24}$$

Choosing $p(x) = [|x_1|, |x_2|]^T$, one obtains a pseudo-overvaluing *M*-matrix of the form

$$M(t,x) = \begin{bmatrix} -3 + x_1^2 & |x_2 \sin x_2| \\ |4x_1^2 \cos x_2| & -3 + x_2^2 \end{bmatrix} \tag{5.25}$$

which, for every *x* in the domain

$$S = \{x \in R^2, |x_1| < 1, |x_2| < 1\} \tag{5.26}$$

is overvalued by the matrix

$$M = \begin{bmatrix} -2 & 0.842 \\ 4 & -2 \end{bmatrix} \tag{5.27}$$

which is Hurwitz, with eigenvectors and eigenvalues as follows:

$$\lambda_1(M) = -0.165, \quad \lambda_2(M) = -3.835$$

$$\Rightarrow \quad \lambda_m(M) = -0.165\,, \quad u_m(M) = [0.459, 1]^T$$

$$\lambda_m(M^T) = -0.165\,, \; u_m(M^T) = [1, 0.459]^T \tag{5.28}$$

and the cone of vectors such that $Mc<0$ is

$$\{c = [c_1, c_2]^T : 2c_1 < c_2 < 2.375c_1\} \tag{5.29}$$

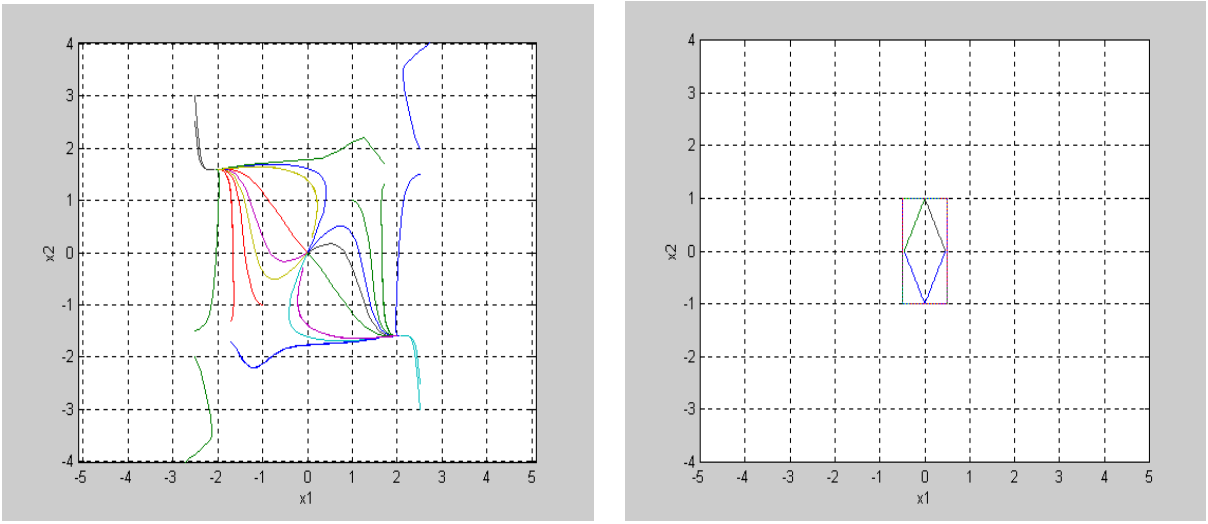
By Theorem 5.2, the three estimates for the region of attraction of the origin are:

$$D_1 = \{x \in R^2 : |x_1| + 0.459|x_2| < 0.459\} \subset S$$

$$D_\infty = \{x \in R^2 : |x_1| < 0.459, |x_2| < 1\} \subset S$$

$$D_c = \{x \in R^2 : |x_1| < 0.5, |x_2| < 1\} \subset S \tag{5.30}$$

It is seen that $D_\infty \subset D_c$. In Figure 5.1, The domains D_1 and D_c are shown together with the phase portrait of the system obtained by the trajectory reversing method. In conclusion, any motion starting in $D_1 \cup D_c$ exponentially tends to the origin (in $e^{-0.165t}$).



(a) (b)

Figure 5.1. (a) Stability region of Example 5.2 obtained by trajectory reversing.
(b) Estimates D_1 (the rhombus) and D_c (the rectangle).

The degree of conservativeness of the ROA estimate depends on the system equations as well as on the chosen $A(x)$ matrix. In the next example, it is shown that the ROA estimate obtained by using this method can recover the exact one.

Example 5.3: [4] Consider the system

$$\begin{aligned} \dot{x}_1 &= -x_1 + 2x_1^2 x_2 \\ \dot{x}_2 &= -x_2 \end{aligned} \tag{5.31}$$

This system is known to have an exact stability region bounded by $x_1x_2 < 1$. Let the state equations be parametrized as $\dot{x} = A(x)x$ with the following choice of $A(x)$:

$$A(x) = \begin{bmatrix} -1 + x_1 x_2 & x_1^2 \\ 0 & -1 \end{bmatrix} \tag{5.32}$$

Choosing $p(x) = [|x_1|, |x_2|]^T$, it can be seen that the matrix $A(x)$ itself would be pseudo-overvaluing matrix for this system. Choosing the domain S as

$$S = \{x \in R^2, |x_1| < \alpha, x_1 x_2 | < 0.99\} \tag{5.33}$$

a constant overvaluing matrix M for (5.30) in S for $\alpha = 4$ would be

$$M = \begin{bmatrix} -0.01 & 16 \\ 0 & -1 \end{bmatrix} \tag{5.34}$$

which is Hurwitz. The domain S was chosen a little smaller than $x_1x_2 < 1$, in order to avoid M having a zero eigenvalue, which is not allowed. In fact, choosing a domain S in which the overvaluing matrix M is Hurwitz is usually the most challenging part of this method, and it may require some iterations.

The importance eigenvectors and eigenvalues of M are as follows:

$$\lambda_m(M) = \lambda_m(M^T) = -0.01, \qquad u_m(M)=[1, 0]^T, \; u_m(M^T)=[0.062, 1]^T \tag{5.35}$$

The cone of vectors such that $Mc<0$ is

$$\{c = [c_1, c_2]^T : c_1 < 1600 c_2\} \tag{5.36}$$

By Theorem 5.2, the estimates for the region of attraction can be obtained as follows:

$$\begin{aligned} D_1 &= \{x \in R^2 : 0.062 \,|\, x_1 \,| + |\, x_2 \,| < 0.248\} \subset S \\ D_\infty &= \{x \in R^2 : |\, x_1 \,| < 4\} \subset S \\ D_c &= \{x \in R^2 : |\, x_1 \,| < 4, |\, x_2 \,| < 0.0025\} \subset S \end{aligned} \tag{5.37}$$

Although the regions D_1 and D_c are quite small, the largest estimate D_∞ for the ROA is in fact equal to the chosen domain S (5.32). It can easily be seen that this estimate can be made to match the exact ROA by choosing α in (5.32) arbitrarily large ($\alpha \to \infty$). In this case, the estimate D_∞ would be simply

$$D_\infty = \{x \in R^2, |\, x_1 x_2 \,| < 0.99\} \tag{5.38}$$

5.3 Estimation of Stability Regions of SDRE Controlled Systems

For SDRE controlled systems, the well known Lyapunov approaches to estimate the region of attraction cannot be used since the closed-loop system equations are usually not known explicitly. The other alternative would be to make time-domain simulations of the closed loop system; however, this would be too cumbersome and costly. The method presented so far in this chapter seems to be a good fit for this purpose, since it does not require that the closed-loop system equations be known explicitly and it eliminates the need for time-domain simulations. Only the maximum and the minimum values of the feedback gains over a chosen domain S in the state-space need be known.

The procedure to estimate the stability region of an SDRE controlled system using this method is as follows:

1) Choose $A(x)$. As a rule of thumb, a choice of $A(x)$ in which all states $x_i,...,x_n$ appear yields a larger stability region than an $A(x)$ in which only some of the states appear.
2) Choose a domain S in the state-space. To obtain the largest ROA estimate possible, S must be chosen as the largest region that would yield a Hurwitz M-matrix. The resulting ROA estimate will be a subset of S.
3) Find the closed-loop matrix $A_{CL}(x,k_1(x),...,k_n(x))$ explicitly in terms of the feedback gains.
4) Find the minimum and maximum values of each feedback gain $k_i(x)$ in S.
5) Now that the bounds on $k_i(x)$ are known, find an overvaluing matrix M for the closed-loop matrix $A_{CL}(x)$. Make sure that M is Hurwitz in the domain S. If M is not Hurwitz in S, either,
   a) Go to step 1 and start with a smaller domain S.
   or,
   b) Start over with another matrix parametrization (i.e., choose another $A(x)$ or $A_{CL}(x)$ representation for the same system equations).
6) Apply Theorem 5.2 to find the three region of attraction estimates, D_1, D_c and D_∞.
7) The region of attraction estimate is given by $D_1 \cup D_c \cup D_\infty$.

As mentioned before, choosing a system representation $A(x)$ or a domain S that would yield a Hurwitz M-matrix can be challenging and may require some iterations as mentioned in Step 5. Since the resulting ROA estimate is a subset of S, the estimate may turn out to be a conservative one if S is too small.

Some examples are given in the next section to illustrate the procedure.

5.4 Examples

Example 5.4: For the nonlinear system (5.31) of Example 5.3, let the $A(x)$ matrix be chosen as

$$A(x) = \begin{bmatrix} -1 & 2x_1^2 \\ 0 & -1 \end{bmatrix} \tag{5.39}$$

With B=[0 1]^T, $Q = I_{2x2}$, R=1; the feedback gains $k_1(x_1)$ and $k_2(x_1)$ depend on x_1 only. The closed-loop matrix $A_{CL}(x)$ such that $\dot{x} = A_{CL}(x)x$ is of the form

$$A_{CL}(x) = \begin{bmatrix} -1 & 2x_1^2 \\ -k_1(x_1) & -1-k_2(x_1) \end{bmatrix} \tag{5.40}$$

An overbounding matrix M for $A_{CL}(x)$ would be of the form

$$M = \begin{bmatrix} -1 & \max\limits_{x_1 \in S} |2x_1^2| \\ \max\limits_{x_1 \in S} |-k_1| & \max\limits_{x_1 \in S}(-1-k_2) \end{bmatrix} \tag{5.41}$$

Let the region S, in which the stability analysis will be done, be chosen as

$$S = \{x_1, x_2 \in R : |x_1| < 1.2, |x_2| < \infty\} \tag{5.42}$$

The variation of $k_1(x_1)$ and $k_2(x_1)$ within S is as shown in Figure 5.2.

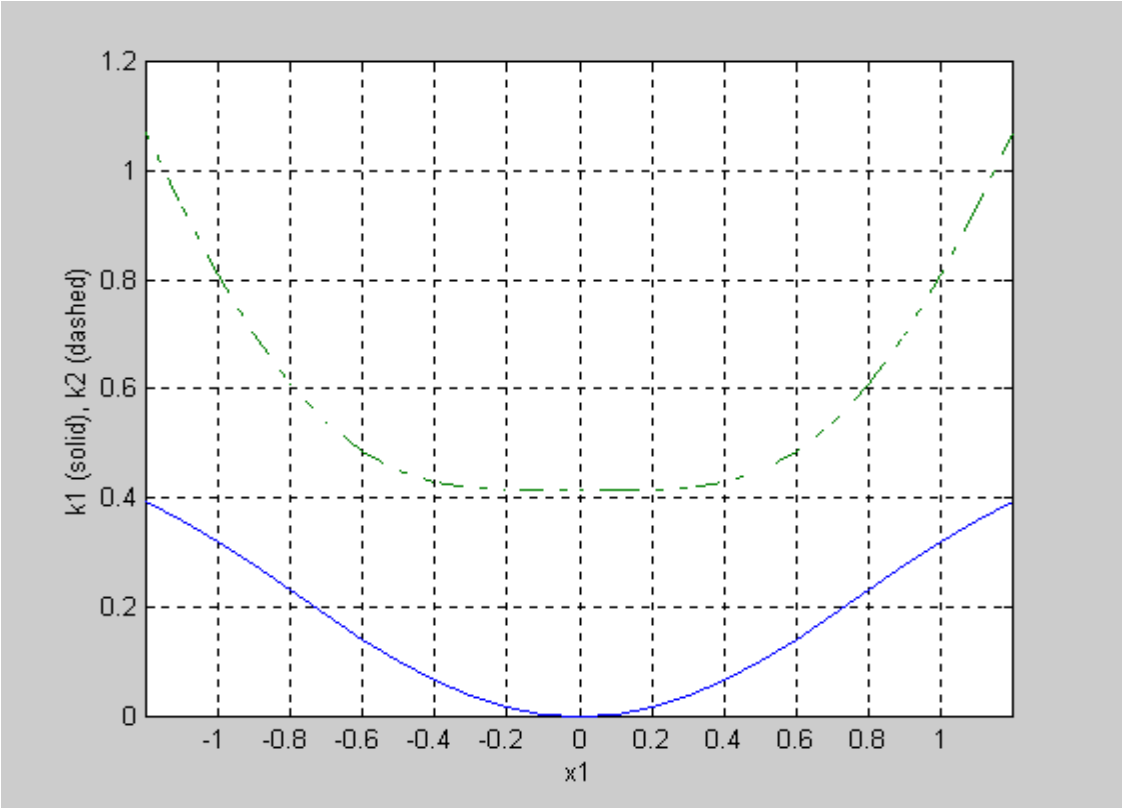


Figure 5.2. Variation of $k_1(x_1)$ and $k_2(x_1)$ for Example 5.4.

Then the M-matrix in (5.41) becomes

$$M = \begin{bmatrix} -1 & 2.88 \\ 0.396 & -1.4142 \end{bmatrix} \tag{5.43}$$

The related importance eigenvalues and eigenvectors are as follows:

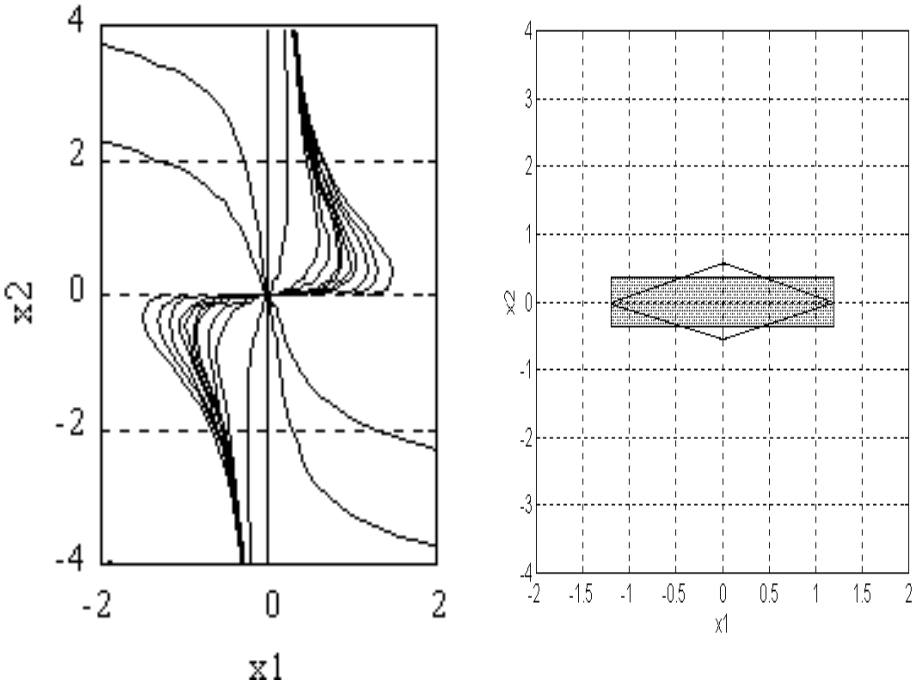
$$\lambda_m(M) = \lambda_m(M^T) = -0.1193$$
$$u_m(M)=[1,\ 0.306]^T, \quad u_m(M^T)=[1,\ 2.224]^T \tag{5.44}$$

By using the vector norm $p(x)=[\ |x_1|,\ |x_2|\ ]^T$, and applying Theorem 5.2, the three region of attraction estimates are found as

$$D_1 = \{x \in R^2 : |x_1| + 2.224\,|x_2| < 1.2\}$$
$$D_\infty = \{x \in R^2 : |x_1| < 1.2, |x_2| < 0.367\}$$

$$D_c = \{x \in R^2 : |x_1| < 1.2, |x_2| < 0.417\} \tag{5.45}$$

Since $D_c \supset D_\infty$, an estimate of the region of attraction of the closed-loop system would be $D_1 \cup D_c$. In Figure 5.3 (a), the simulated trajectories of the closed-loop system are shown. The region of attraction estimates D_1 and D_c are shown in Figure 5.3 (b).



(a) (b)

Figure 5.3 For Example 5.4, (a) simulated trajectories of the closed-loop system, (b) region of attraction estimates D_1 (the rhombus) and D_c (the rectangle).

Example 5.5:

For the same dynamical system (5.31) in Examples 5.3 and 5.4, let the $A(x)$ matrix be now chosen as

$$A(x) = \begin{bmatrix} -1 + x_1 x_2 & x_1^2 \\ 0 & -1 \end{bmatrix} \tag{5.32}$$

Again, with B=[0 1]^T, $Q = I_{2x2}$, R=1, the closed-loop matrix $A_{CL}(x)$ is of the form

$$A_{CL}(x)=\begin{bmatrix} -1+x_1x_2 & x_1^2 \\ -k_1(x_1,x_2) & -1-k_2(x_1,x_2) \end{bmatrix} \tag{5.46}$$

The feedback gains k_1 and k_2 now depend on both x_1 and x_2. Their variations are shown in Figure 5.4.

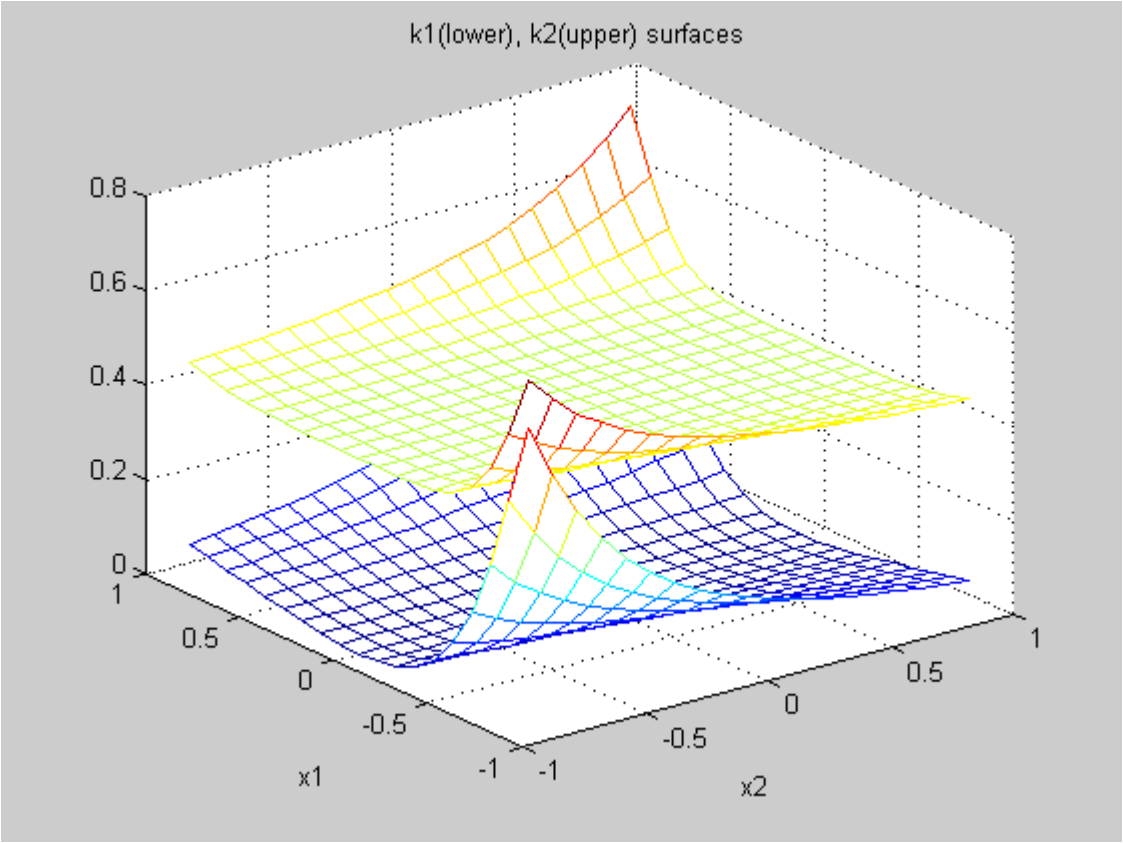


Figure 5.4 Variations of $k_1(x_1,x_2)$ and $k_2(x_1,x_2)$ for Example 5.5.

An overbounding matrix M for $A_{CL}(x)$ would be of the form

$$M=\begin{bmatrix} \max\limits_{x_1,x_2\in S}(-1+x_1x_2) & \max\limits_{x_1,x_2\in S}|x_1^2| \\ \max\limits_{x_1,x_2\in S}|-k_1| & \max\limits_{x_1,x_2\in S}(-1-k_2) \end{bmatrix} \tag{5.47}$$

Choosing the region S be chosen as

$$S = \{x_1, x_2 \in R : | x_1 | < 0.9, x_1 x_2 < 0.63\} \tag{5.48}$$

the overvaluing M-matrix is obtained as

$$M = \begin{bmatrix} -0.37 & 0.81 \\ 0.644 & -1.4142 \end{bmatrix} \tag{5.49}$$

with the related importance eigenvalues and eigenvectors as follows:

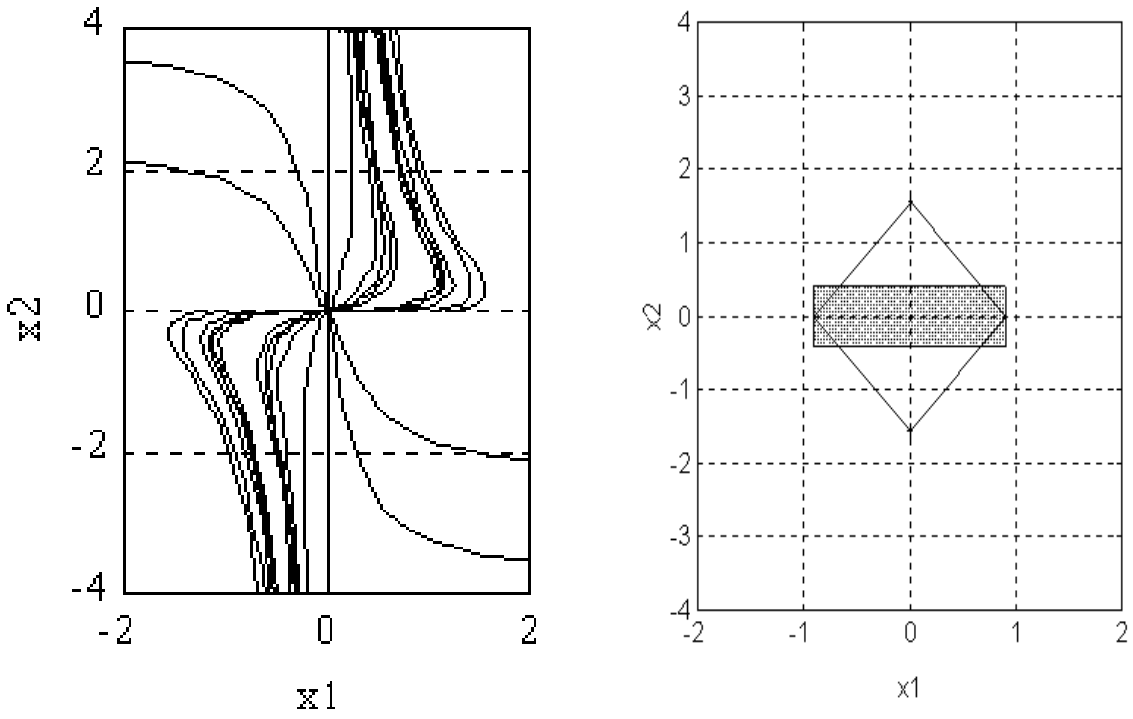
$$\lambda_m(M) = \lambda_m(M^T) = -0.0009$$
$$u_{\mathrm{m}}(M)=[1, 0.456]^T, \quad u_{\mathrm{m}}(M^{\mathrm{T}})=[1, 0.573]^T \tag{5.50}$$

The barely negative eigenvalue –0.0009 of the *M*-matrix shows that the region *S* is indeed as large as can possibly be chosen to yield a Hurwitz *M*-matrix.

By using the vector norm $p(x)=[\ |x_1|,\ |x_2|\ ]^T$, and applying Theorem 5.2, the three region of attraction estimates are found as

$$D_1 = \{x \in R^2 : | x_1 | + 0.573 | x_2 | < 0.9\}$$
$$D_\infty = \{x \in R^2 : | x_1 | < 0.9, | x_2 | < 0.41\}$$
$$D_c = \{x \in R^2 : | x_1 | < 0.9, | x_2 | < 0.41\} \tag{5.51}$$

Since $D_c = D_\infty$, an estimate of the region of attraction of the closed-loop system would be $D_1 \cup D_\infty$. In Figure 5.5 (a), the simulated trajectories of the closed-loop system are shown. The region of attraction estimates D_1 and D_∞ are shown in Figure 5.5 (b).



(a) (b)

Figure 5.5 For Example 5.5, (a) simulated trajectories of the closed-loop system, (b) region of attraction estimates D_1 (the rhombus) and D_∞ (the rectangle).

Example 5.6: Consider the 3rd order system with coefficient matrix

$$A(x)=\begin{bmatrix} -1 & 0 & 1.5 \\ 0 & -3 & 1.5 \\ 1 & 1 & -2-\sin(x_3)/x_3 \end{bmatrix} \tag{5.52}$$

With B=[0 0 1]^T, $Q = I_{3x3}$, R=1, the closed-loop matrix $A_{CL}(x)$ is of the form

$$A(x)=\begin{bmatrix} -1 & 0 & 1.5 \\ 0 & -3 & 1.5 \\ 1-k_1 & 1-k_2 & -2-\sin(x_3)/x_3-k_3 \end{bmatrix} \tag{5.53}$$

Let the domain S be chosen as the open set

$$S=\{x\in R^3, |\, x_3 \,| < \pi\} \tag{5.54}$$

The maximum and minimum values of the feedback gains in S are as follows:

Min(k1)=0.4199 max(k1)=0.6567
Min(k2)=0.1441 max(k2)=0.2459
Min(k3)=0.4194 max(k3)=0.7815

Then an overvaluing M-matrix for this system is

$$M = \begin{bmatrix} -1 & 0 & 1.5 \\ 0 & -3 & 1.5 \\ 0.85 & 0.856 & -2.4194 \end{bmatrix} \tag{5.55}$$

The importance eigenvalues and eigenvectors of M are as follows:

$$\lambda_m(M) = \lambda_m(M^T) = -0.4205$$
$$u_{\mathrm{m}}(M)=[0.5815, 2.5886, 1]^T, \qquad u_{\mathrm{m}}(M^{\mathrm{T}})=[1.0009, 0.3318, 1]^T \tag{5.56}$$

By using the vector norm $p(x)=[\,|x_1|, |x_2|, |x_3|]^T$, and applying Theorem 5.2, the three region of attraction estimates are found as

$$D_1 = \{x \in R^2 : |x_1| + 0.332\,|x_2| + |x_3| < \pi\}$$
$$D_\infty = \{x \in R^2 : |x_1| < 0.581\pi, |x_2| < 2.589\pi, |x_3| < \pi\} \tag{5.57}$$

$D_1 \cup D_\infty \cup D_c$ is an estimate of the region of attraction.

5.5 References

[1] P. Borne, J.P. Richard, “Local and Global Stability of Attractors by use of Vector Norms”, *The Lyapunov Functions Method and Applications*, P. Borne and V. Matrosov (editors), pp.53-62, IMACS 1990.

[2] P. Borne, J.P. Richard, N.E. Radhy, “Stability, stabilization, regulation using vector norms”, *Chapter 2 in Nonlinear systems Vol.2: Stability and Stabilization,* A.J. Fossard and D. Normand-Cyrot (editors), Chapman & Hall, 1996.

[3] P. Borne, J.P. Richard, M. Tahiri, “Estimation of Attractive Domains for Locally Stable or Unstable Systems”, *Systems Analysis Modelling Simulation*, vol.7, no.8, pp.595-610, 1990.

[4] R. Genesio, R. Tartaglia, A. Vicino, “On the estimation of Asymptotic Stability Regions: State of the Art and New Proposals”, *IEEE Transactions on Automatic Control*, vol.30, no.8, pp.747-755, 1985.

[5] J.P. Richard, P. Borne, J.C. Gentina, “Estimation of Stability Domains by Use of Vector Norms”, *Information and Decision Technologies*, vol.14, pp.241-251, 1988.

Chapter 6

A Real-Time Control Experiment Using SDRE

In the pioneering work of Langson [1, 2] the design and hardware-in-the-loop experimental implementation of an SDRE controller for a third order nonlinear system was realized. Two 75 MHz Pentium computers were interfaced where one computer simulated the 3rd order plant dynamics, and the other acted as the SDRE controller. A sampling rate of 100 Hz was used. A hand-coded C program was used in the controller PC to solve the Riccati Equation and compute the gains. The solution was based on finding the eigenvalues of the Hamiltonian matrix and calculating the gains by pole placement [1]. This was the most computationally demanding part of the experiment. Despite the relatively slow speed of the computers, the results were quite good which suggested that the associated computational burden was not so much to rule out SDRE from real-time control applications.

With this motivation, coupled with the handiness of dSPACE digital signal processor, the experiment which is the subject of this chapter was initiated. The novelty in this experiment is that SDRE control is applied to a *physical* plant that is highly nonlinear. Thus, it is the first time in literature that SDRE control is applied i) on a physical plant, ii) in real-time, iii) by computing the SDRE gains online.

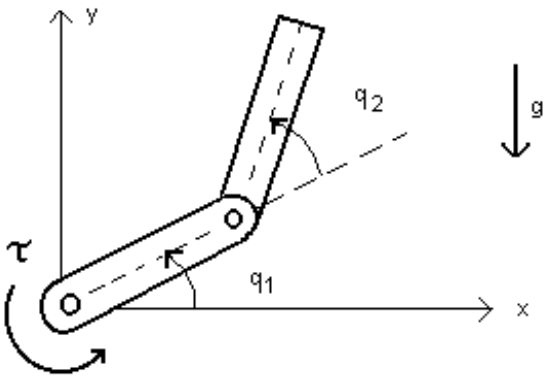


Figure 6.1 Pictorial of the Pendubot.

6.1 The Plant: The Pendubot

The Pendubot, shown schematically in Figure 6.1, is a two-link, underactuated robotic mechanism that was built at the University of Illinois [5].

The equations of motion for the Pendubot in matrix form are

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau \tag{6.1}$$

where τ is the torque vector and q is the vector of joint angle positions. The system is underactuated since torque is applied only on the first link.

The masses and link lengths are grouped into five system parameters $\theta_1,\ldots,\theta_5$ as follows:

$$\begin{aligned}
\theta_1 &= m_1 l_{c1}^2 + m_2 l_1^2 + I_1 \\
\theta_2 &= m_2 l_{c2}^2 + I_2 \\
\theta_3 &= m_2 l_1 l_{c2} \\
\theta_4 &= m_1 l_{c1} + m_2 l_1 \\
\theta_5 &= m_1 l_{c2}
\end{aligned} \tag{6.2}$$

where

m_1: the total mass of link 1.
l_1: the length of link 1.
l_{c1}: the distance to the center of mass of link 1.
I_1: the moment of inertia of link 1 about its centroid.
m_2: the total mass of link 2.
l_{c2}: the distance to the center of mass of link 2.
I_2: the moment of inertia of link 2 about its centroid.
g: gravitational acceleration.

The terms in (6.1) in terms of $\theta_1,\ldots,\theta_5$ are as follows:

$$D(q) = \begin{bmatrix} \theta_1 + \theta_2 + 2\theta_3 \cos q_2 & \theta_2 + \theta_3 \cos q_2 \\ \theta_2 + \theta_3 \cos q_2 & \theta_2 \end{bmatrix}$$

$$C(q,q) = \begin{bmatrix} -\theta_3 \sin(q_2)\dot{q}_2 & -\theta_3 \sin(q_2)\dot{q}_2 - \theta_3 \sin(q_2)\dot{q}_1 \\ \theta_3 \sin(q_2)\dot{q}_1 & 0 \end{bmatrix} \tag{6.3}$$

$$g(q) = \begin{bmatrix} \theta_4 g \cos(q_1) + \theta_5 g \cos(q_1 + q_2) \\ \theta_5 g \cos(q_1 + q_2) \end{bmatrix}$$

where the values of $\theta_1,\ldots,\theta_5$ are

θ_1= 0.0308 V*s^2
θ_2= 0.0106 V*s^2
θ_3= 0.0095 V*s^2 (6.4)
θ_4= 0.2097 V*s^2/m
θ_5= 0.063 V*s^2/m

The objective of this experiment is to regulate the Pendubot at vertical equilibrium position with $q_1=\pi/2$ and $q_2=0$. By choosing the state variables as

$$x_1 = q_1 - \pi/2\,, \qquad x_2 = \dot{q}_1\,, \qquad x_3 = q_2\,, \qquad x_4 = \dot{q}_2 \tag{6.5}$$

the equations of motion in state-space form become

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= [-\sin x_3(-2\theta_2\theta_3 x_2 x_4 - \theta_2\theta_3 x_4^2 - \theta_2\theta_3 x_2^2 - {\theta_3}^2 x_2^2 \cos x_3 + \cos x_1 \cos x_3 \theta_3\theta_5 g) \\
&\quad - \sin x_1(\cos^2 x_3\theta_3\theta_5 g - \theta_2\theta_4 g) + \theta_2\tau]/\det \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= [-\sin x_3(2\theta_2\theta_3 x_2 x_4 + \theta_2\theta_3 x_4^2 + (\theta_1 + \theta_2)\theta_3 x_2^2 + 2{\theta_3}^2 x_2(x_2 + x_4)\cos x_3 \\
&\quad + {\theta_3}^2 x_4^2 \cos x_3 - \cos x_1\theta_1\theta_5 g - \cos x_1 \cos x_3\theta_3\theta_5 g) - \sin x_1(\cos^2 x_3\theta_3\theta_5 g \\
&\quad - \theta_2\theta_4 g + (\theta_1\theta_5 - \theta_3\theta_4) g \cos x_3) + (\theta_2 + \theta_3 \cos x_3)\tau]/\det
\end{aligned} \tag{6.6}$$

where

$$\det = \theta_1\theta_2 - \theta_3^2 \cos x_3 \tag{6.7}$$

is the determinant of the inertia matrix and is never zero.

6.2 Parametrization for SDRE Control

The highly nonlinear state equations (6.6) are parametrized in state-dependent coefficient form $\dot{x} = A(x)x + B(x)u$ with the following choice of $A(x)$ and $B(x)$:

$$A(x) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ a_{21}(x) & 0 & a_{23}(x) & 0 \\ 0 & 0 & 0 & 1 \\ a_{41}(x) & 0 & a_{43}(x) & 0 \end{bmatrix}, \quad B(x) = \begin{bmatrix} 0 \\ b_2(x) \\ 0 \\ b_4(x) \end{bmatrix} \tag{6.8}$$

where

$$\mathrm{a}_{21}(\mathrm{x}) = -\frac{1}{\det}(\cos^2 x_3\theta_3\theta_5 g - \theta_2\theta_4 g)\frac{\sin x_1}{x_1}$$

$$a_{23}(x) = -\frac{1}{\det}(-2\theta_2\theta_3 x_2 x_4 - \theta_2\theta_3 x_4^2 - \theta_2\theta_3 x_2^2 - \theta_3{}^2 x_2^2 \cos x_3 + \cos x_1 \cos x_3\theta_3\theta_5 g)\frac{\sin x_3}{x_3}$$

$$a_{41}(x) = -\frac{1}{\det}(\cos^2 x_3\theta_3\theta_5 g - \theta_2\theta_4 g + (\theta_1\theta_5 - \theta_3\theta_4)g\cos x_3)\frac{\sin x_1}{x_1}$$

$$a_{43}(x) = -\frac{1}{\det}(2\theta_2\theta_3 x_2 x_4 + \theta_2\theta_3 x_4^2 + (\theta_1 + \theta_2)\theta_3 x_2^2 + 2\theta_3{}^2 x_2(x_2 + x_4)\cos x_3$$
$$+ \theta_3{}^2 x_4^2 \cos x_3 - \cos x_1\theta_1\theta_5 g - \cos x_1 \cos x_3\theta_3\theta_5 g)\frac{\sin x_3}{x_3}$$

$$b_2(x) = \theta_2 / \det$$

$$b_4(x) = (-\theta_2 - \theta_3 \cos x_3)/\det \tag{6.9}$$

With this parametrization, the entries of $A(x)$ are always continuous in x and bounded, since $\sin(x)/x \to 1$ as $x \to 0$. Before proceeding further to design the control, it should be checked whether $(A(x), B(x))$ is pointwise stabilizable. The determinant of the controllability matrix is

$$\det W_{contr}(x) = (a_{43}b_4b_2 - a_{23}b_4^2 + a_{41}b_2^2 - a_{21}b_2b_4)^2 \tag{6.10}$$

The region of the state-space in which the system is NOT controllable ($\det W_{contr}=0$) is the union of the following two regions:

1) $x_1=\pm\pi$, $x_3=\pm\pi$, $\forall x_2, x_4$.
2) $x_1=\pm\pi/2$, $x_3=\pm\pi/2$ AND $x_2=0$, $\forall x_4$.

The first set of configurations (*1*) corresponds to the first link being horizontal and the second link at a 90° angle from the first one. These coincide with the physically uncontrollable positions. The second configuration (*2*) corresponds to the links folded over each other, with the first link hanging downward. This is not a physically uncontrollable position, meaning that another control exists that would render the system controllable at this particular position. It just indicates that there does not exist an SDRE control for the particular choice of parametrization $A(x)$.

Thus, the particular parametrization $(A(x), B(x))$ is pointwise stabilizable in the practical region of operation of this experiment. In Figure 6.2, the value of $\det W_{contr}$ on the x_1-x_3 plane with $x_2=0$, $x_4= 0$ is shown. In Figure 6.3, the value of $\det W_{contr}$ on the x_2-x_4 plane with $x_1=\pi/2$, $x_3=\pi/2$ rad is shown. In Figure 6.4, the uncontrollable positions on the x_1-x_3 plane with $x_2=0$, $x_4= 0$ are shown, marked '*' along with the contours of the determinant of the controllability matrix, $\det W_{contr}$.

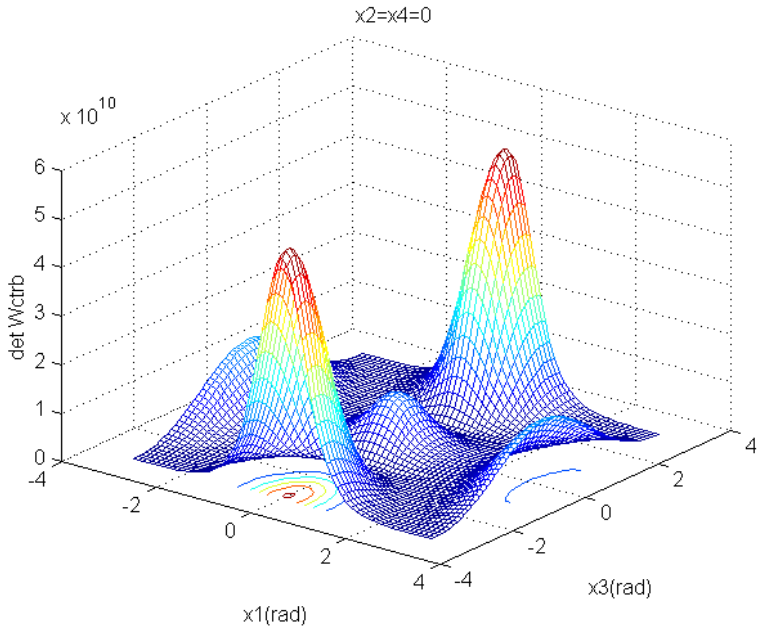


Figure 6.2 Determinant of the controllability matrix on the x_1-x_3 plane with x_2=0, x_4= 0.

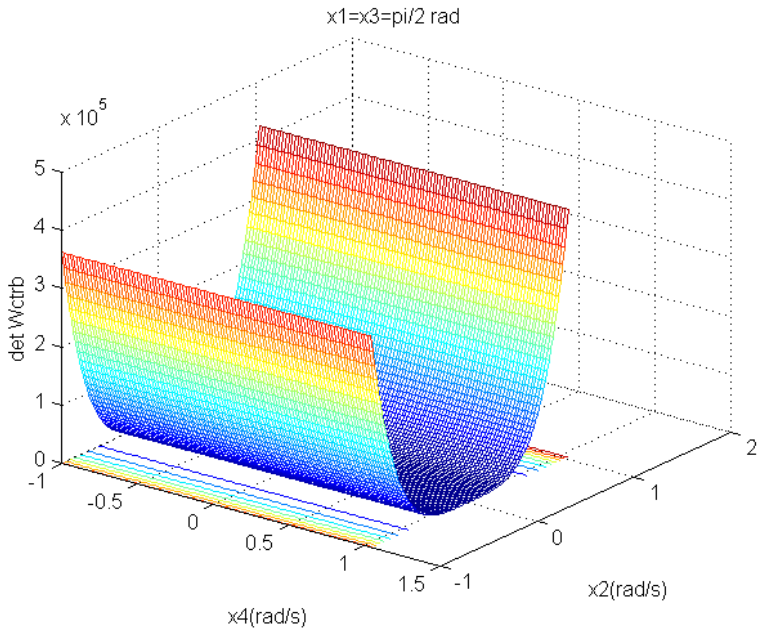


Figure 6.3 Determinant of the controllability matrix on the x_2-x_4 plane.

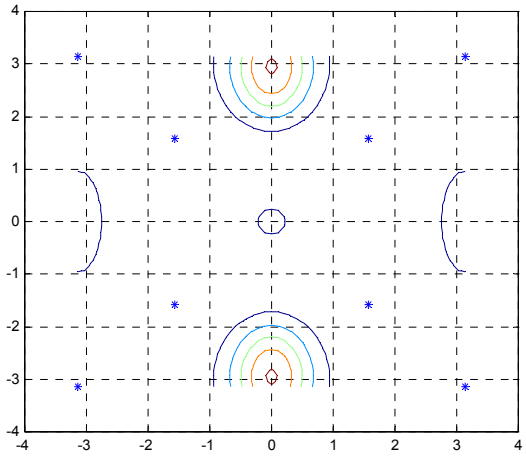


Figure 6.4 Contours of the determinant of the controllability matrix and the uncontrollable positions (marked '*') on the x_1-x_3 plane with x_2=0, x_4= 0.

6.3 Simulation Results

The plant model used in the simulations in this section is from [1], which is more realistic than the model used for controller design, since it includes friction as well. The first controller tried in simulation is LQR obtained by using the linearized state equations around x=0 and Q=diag([1 0 1 0]), R=1. The fixed gains for this controller are K=[16.462 3.129 16.242 2.063]. In simulations it was seen that with zero initial link velocities, this controller fails to stabilize if the link angles are greater than 0.6 rad in magnitude (opposite signs).

The second controller tried is SDRE, again with Q=diag([1 0 1 0]), R=1. This controller outperforms the LQR one by stabilizing from initial link angles as high as 0.9 rad. This result directly reflects the advantage of using the nonlinear state equations in controller design over using the linearized equations. Since the input magnitude is constrained to be 30 (Nm), the SDRE control, too, goes unstable after 0.9 rad due to lack of actuator input. In Figure 6.5, the responses to the two controllers mentioned above are shown with initial conditions x_0=[-0.6, 0.001, 0.6, 0.001]^T. It is seen that the SDRE controller yields smaller state norm, smaller control input and smoother response than the LQR controller. It is also seen that the SDRE controller responds faster than the LQR controller. This is expected, since the SDRE control is calculated based on the dynamics at x_0=[-0.6, 0.001, 0.6, 0.001]^T initially, while the LQR control is calculated based on the dynamics at the origin. It is also

seen that the LQR controller exhibits a peaking phenomenon that eventually diverges when $|x_1| = |x_3| > 0.6$ rad. The SDRE controller avoids peaking.

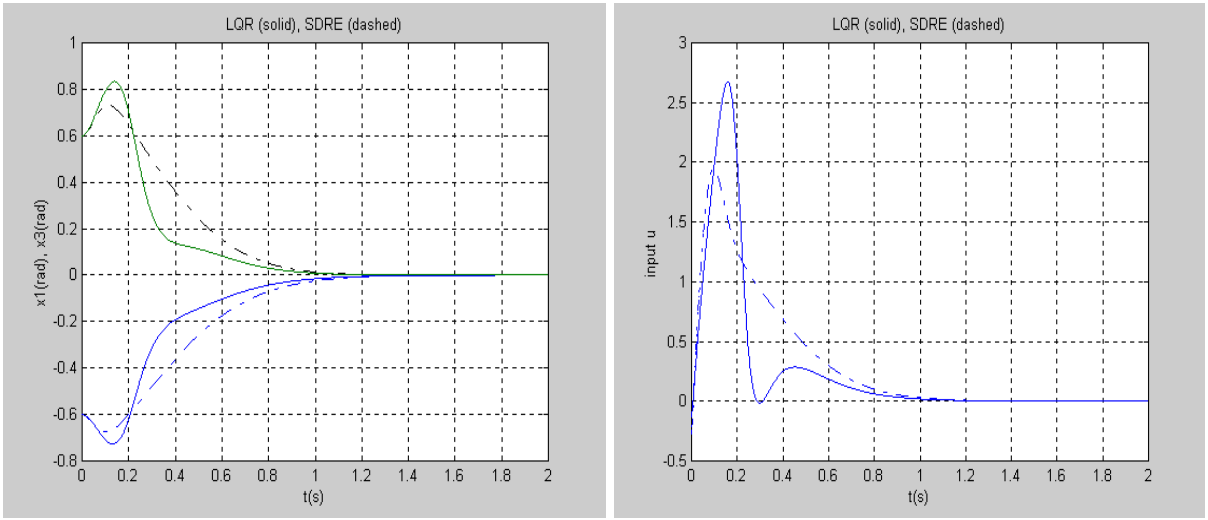


Figure 6.5. Response to LQR and SDRE controllers with Q=diag([1 0 1 0]), R=1 from initial conditions x_0=[-0.6, 0.001, 0.6, 0.001]^T.

In Figure 6.6, the responses to the same two controllers are shown with initial conditions x_0=[-0.62, 0.001, 0.62, 0.001]^T. It is seen that while the SDRE controller easily stabilizes the system, the LQR controlled system becomes unstable. These results clearly illustrate how using the nonlinear state equations instead of the linearized ones can increase the performance and the stability region of the closed-loop system.

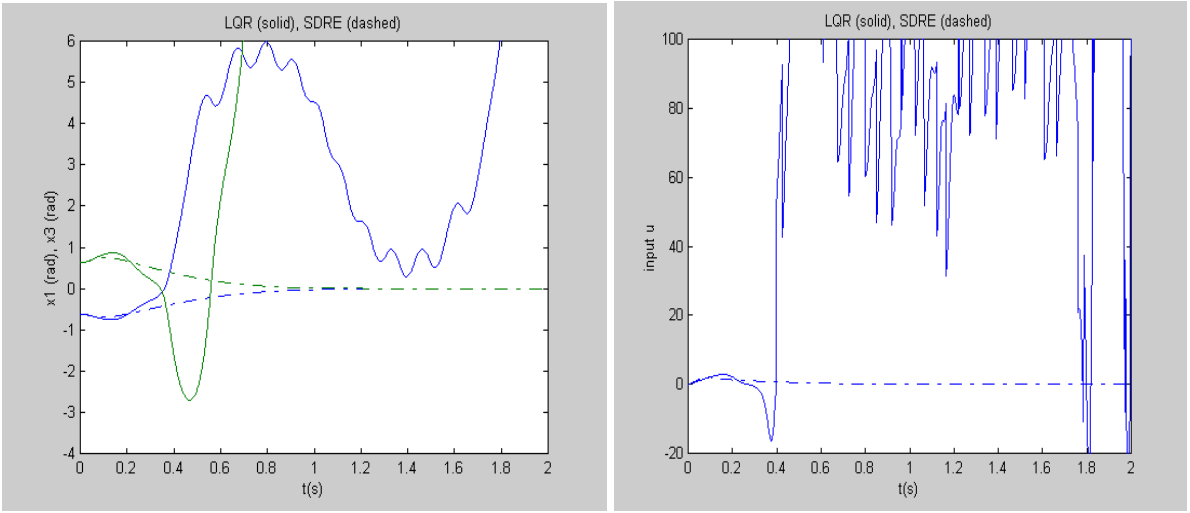


Figure 6.6 Response to LQR and SDRE controllers with Q=diag([1 0 1 0]), R=1 from initial conditions x_0=[-0.62, 0.001, 0.62, 0.001]^T.

Another SDRE controller with state-dependent $Q(x)$ and $R(x)$ matrices was also simulated on the model. Figure 6.8 shows the response to the two SDRE controllers with initial conditions x_0=[-0.9, 0.001, 0.9, 0.001]^T. Case 1, which is shown with dashed lines, corresponds to the controller with Q=diag([1 0 1 0]), R=1. Case 2, shown with solid lines corresponds to the controller with $Q(x) = diag[(1+0.5x_1^2),0,(1+0.5x_3^2),0]$ and $R(x)$=1/(0.05+2||x||). This example illustrates how in SDRE control the weighting matrices Q and R can be chosen as functions of the states so as to obtain the desired system response.

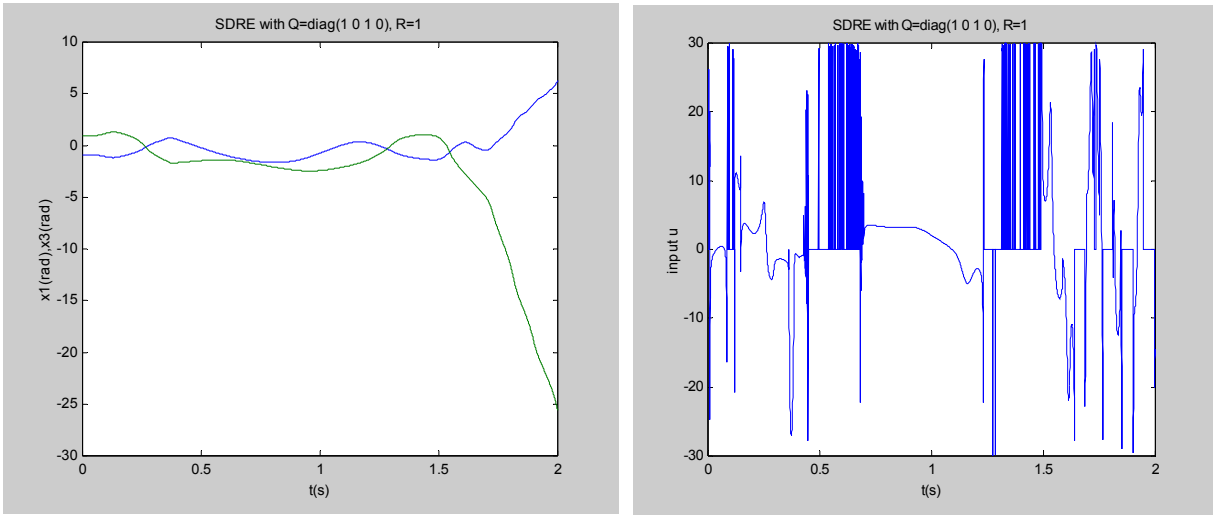


Figure 6.7. SDRE control becomes unstable at x_0=[-0.91, 0.001, 0.91, 0.001]^T.

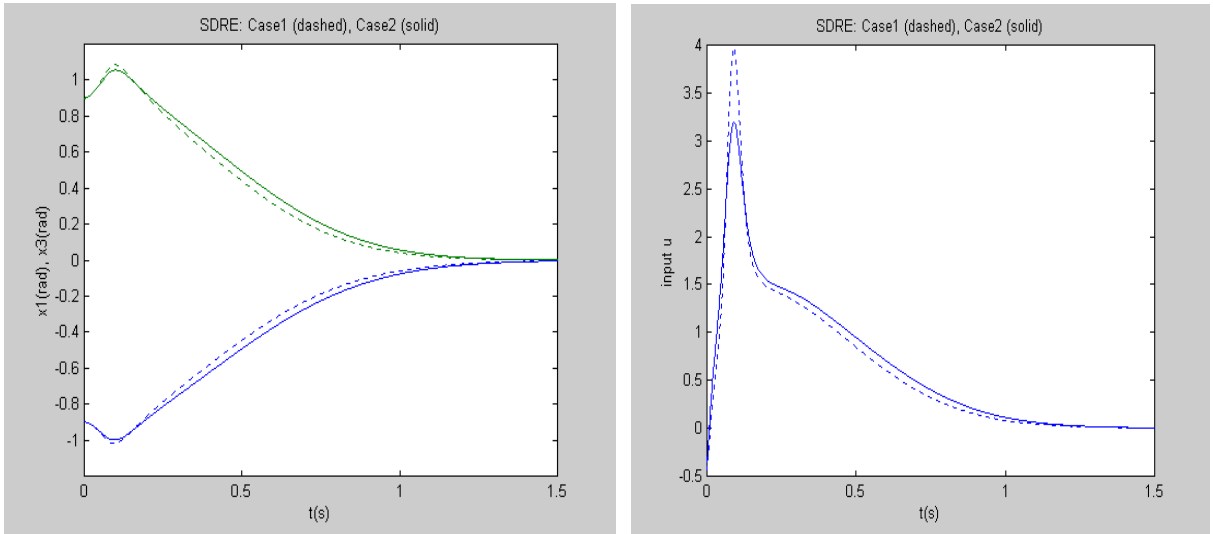


Figure 6.8. Response to SDRE controllers: *Case 1:* Q=diag([1 0 1 0]), R=1.

Case 2: $Q(x) = diag[(1+0.5x_1^2),0,(1+0.5x_3^2),0]$, $R(x)$=1/(0.05+2||x||).

6.3.1 Region of Attraction of the Pendubot

The computation of the region of attraction (ROA) of the Pendubot using the overvaluing systems method discussed in Chapter 5 was attempted. It turned out that the particular $A(x)$ choice would not yield a Hurwitz M-matrix, even at x=0. Thus, for the SDRE controlled Pendubot with this $A(x)$ choice, the method of overvaluing matrices to find the ROA estimate cannot be applied. The relevant computations are shown in the sequel. For ease of notation, x-dependence of the variables has been omitted. It should be mentioned that another $A(x)$ may be found that would yield a Hurwitz M-matrix.

The closed-loop coefficient matrix $A_{\mathrm{CL}}(x)$ is of the form

$$A_{CL}(x)=\begin{bmatrix} 0 & 1 & 0 & 0 \\ a_{21}-b_2k_1 & -b_2k_2 & a_{23}-b_2k_3 & -b_2k_4 \\ 0 & 0 & 0 & 1 \\ a_{41}-b_4k_1 & -b_4k_2 & a_{43}-b_4k_3 & -b_4k_4 \end{bmatrix} \tag{6.11}$$

The M-matrix would be in the form

$$M=\begin{bmatrix} 0 & 1 & 0 & 0 \\ \max\limits_{x\in S}|a_{21}-b_2k_1| & \max\limits_{x\in S}(-b_2k_2) & \max\limits_{x\in S}|a_{23}-b_2k_3| & \max\limits_{x\in S}|-b_2k_4| \\ 0 & 0 & 0 & 1 \\ \max\limits_{x\in S}|a_{41}-b_4k_1| & \max\limits_{x\in S}|-b_4k_2| & \max\limits_{x\in S}|a_{43}-b_4k_3| & \max\limits_{x\in S}(-b_4k_4) \end{bmatrix} \tag{6.12}$$

Unfortunately, due to the absolute valuing of the off-diagonal elements of M, it is not possible for M to be Hurwitz. For x=0 in particular with Q=diag([1 0 1 0]) and R=1, the M-matrix is

$$M(x=0)=\begin{bmatrix} 0 & 1 & 0 & 0 \\ 872.2 & 151.4 & 760.3 & 99.4 \\ 0 & 0 & 0 & 1 \\ 1595.7 & 287.2 & 1383.3 & -188.4 \end{bmatrix} \tag{6.13}$$

with eigenvalues [228.73, -260.14, -5.55, -0.02]. Thus, the method in Chapter 5 cannot be applied to find the ROA estimate.

6.4 Real-Time Implementation

6.4.1 The Experimental System

A 60 MHz dSPACE DS1102 digital signal processor was used for the real-time experiments. The dSPACE Real-Time Interface (RTI) connects the MathWorks' software MATLAB, SIMULINK and the Real-Time Workshop (RTW) with dSPACE's real-time systems to form an integrated and ready-to-use development environment for real-time applications. It is widely used for controller prototyping and hardware-in-the-loop simulation in the automotive industry. The working principle is as follows.

The RTW generates C-code from the SIMULINK graphical model in MATLAB. This code is the loaded to the DSP by means of the RTI. Any kind of SIMULINK model for which code can be generated by means of the RTW can be implemented by the RTI, including continuous-time, discrete-time and hybrid systems. Time histories of the variables can be recorded and monitored by the TRACE module. These data can be transferred to MATLAB workspace for further analysis and visualization. The COCKPIT module allows experiment control, real-time parameter tuning and signal monitoring.

The DS1102 board has 4 analog output, 4 analog input and 2 encoder input channels. In this experiment, the positions of the links were read from optical encoders. The velocities were found by running the position signals through a transfer function in the SIMULINK model that served as both a differentiator and a low-pass filter. The input torque signal, after having passed through the DAC of ±10 V limits, was sent to the DC motor driving the first link. In order not to overload the equipment, a safety switching was incorporated such that the control input signal would be set to zero in case the DAC output was larger than 3V in magnitude.

6.4.2 Computation of SDRE Gains in Real-Time

The SDRE control is computed by a MATLAB executable S-function written in C. This code, *sdrecode.c*, is presented in Appendix A. The inputs to this program are the four states as well as the penalty matrices $Q(x)$ and $R(x)$, which can be functions of the states. The outputs are the four gains and the input signal, u.

At a particular state x, the problem of computing the SDRE gains boils down to solving the Algebraic Riccati Equation (ARE). Various approaches exist for the solution of the ARE. Most of them are based on finding the eigenvalues and eigenvectors of the associated Hamiltonian matrix (6.14).

$$H(x) = \begin{bmatrix} A(x) & -B(x)R^{-1}(x)B^{T}(x) \\ -Q(x) & -A^{T}(x) \end{bmatrix} \tag{6.14}$$

The stable eigenvalues of the Hamiltonian matrix are also the closed loop poles. For single input systems, a pole placement algorithm can be used to find the feedback gains K [3] in terms of the stable eigenvalues of H. For multi-input systems, the eigenvectors of H need also be computed which can be computationally costly especially for high order systems.

Since the Pendubot is a single input system, it is sufficient that only the eigenvalues of H be computed. The controller algorithm includes four subroutines: the first subroutine balances the Hamiltonian matrix. The second one puts it into Hessenberg form. Then the third subroutine finds the eigenvalues by performing a QR-decomposition. Finally the last subroutine sorts the stable eigenvalues (closed-loop poles) in ascending order according to their real parts:

$$\lambda_j(x) = a_j(x) + ib_j(x), \qquad j = 1,\ldots,4. \tag{6.15}$$

The codes for these subroutines were taken from [10, 11]. Some modifications were made in order to run in real-time and to yield the shortest computation time possible.

The feedback gains were obtained by Ackerman's formula (6.16). For ease of notation, state-dependence of the variables has been suppressed in the sequel.

$$K = [0 \quad 0 \quad 0 \quad 1](W_{ctrb})^{-1}\alpha_c(A) \tag{6.16}$$

where W_{ctrb} is the controllability matrix and $\alpha_c(A)$ is a matrix defined as

$$\alpha_c(A) = A^4 + \alpha_3 A^3 + \alpha_2 A^2 + \alpha_1 A + \alpha_0 I \tag{6.17}$$

where α_k are the coefficients of the desired closed loop characteristic polynomial:

$$\alpha(s) = \prod_{j=1}^{4}(s - a_j - ib_j) \tag{6.18}$$

Comparing (6.17) and (6.18), the coefficients α_k are found as

$$\alpha_3 = -a_1-a_2-a_3-a_4 \tag{6.19}$$

$$\alpha_2 = a_1a_3-b_1b_3+a_1a_2-b_3b_4-b_2b_4+a_2a_3-b_1b_4-b_2b_3+a_3a_4+a_2a_4+a_1a_4-b_1b_2 \tag{6.20}$$

$$\alpha_1 = a_1b_3b_4+b_1a_2b_3+a_2b_3b_4+b_1b_3a_4-a_1a_2a_3+b_1a_2b_4+b_2b_3a_4+b_1b_2a_4+b_2a_3b_4-a_2a_3a_4 + a_1b_2b_4+b_1b_2a_3-a_1a_2a_4-a_1a_3a_4+a_1b_2b_3+b_1a_3b_4 \tag{6.21}$$

$$\alpha_0 = -b_1a_2b_3a_4-b_1b_2a_3a_4-a_1b_2b_3a_4-a_1a_2b_3b_4+a_1\,a_2\,a_3\,a_4+b_1\,b_2\,b_3\,b_4-b_1\,a_2\,a_3\,b_4-a_1\,b_2\,a_3\,b_4 \tag{6.22}$$

The input signal is obtained by $u=-Kx$ where K is found using (6.16).

6.4.3 Experimental Results

The Pendubot was made to regulate at the top open-loop unstable equilibrium, subject to manual disturbances as shown in Figure 6.9. The state penalty matrix was chosen in the form $Q(x)$=diag ($q_1(x)$, 0, $q_3(x)$, 0), (i.e., no penalty on link velocities) and the input penalty matrix as simply $r(x)$. The real-time implementation could be achieved at a sampling rate of 100 Hz. With its open-loop eigenvalues being 1.83 Hz and 1.01 Hz, it can safely be concluded that this is a fast enough sampling time for the Pendubot.

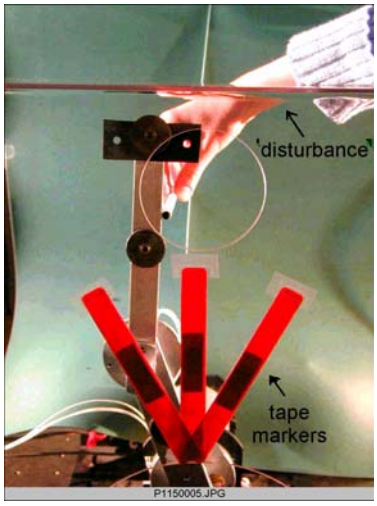


Figure 6.9 Manually disturbing the Pendubot
away from its equilibrium position.

For the choice of $q_1=q_3=r=1$, Figure 6.10 shows the experimental LQR control results and Figure 6.11 shows the experimental SDRE control results. It can be seen that for this choice of the weights the performances of both controllers are functionally similar.

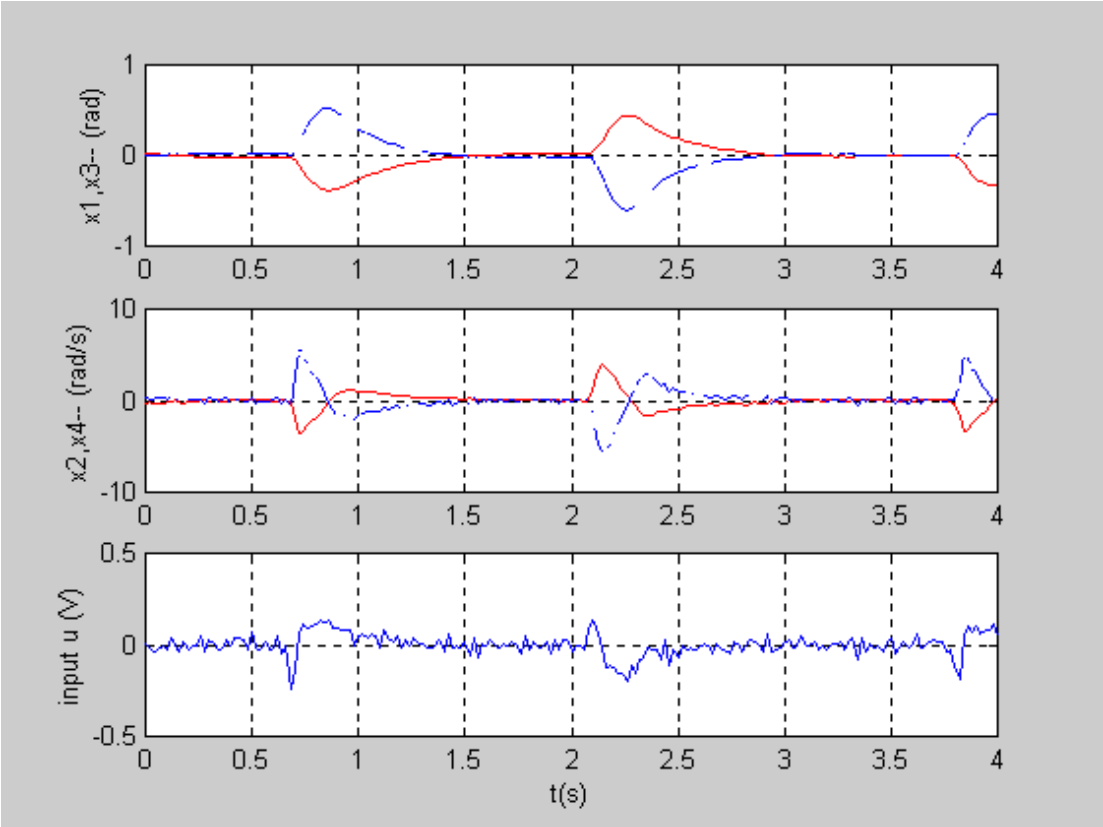


Figure 6.10 Response to LQR with $q_1=q_3=r=1$.

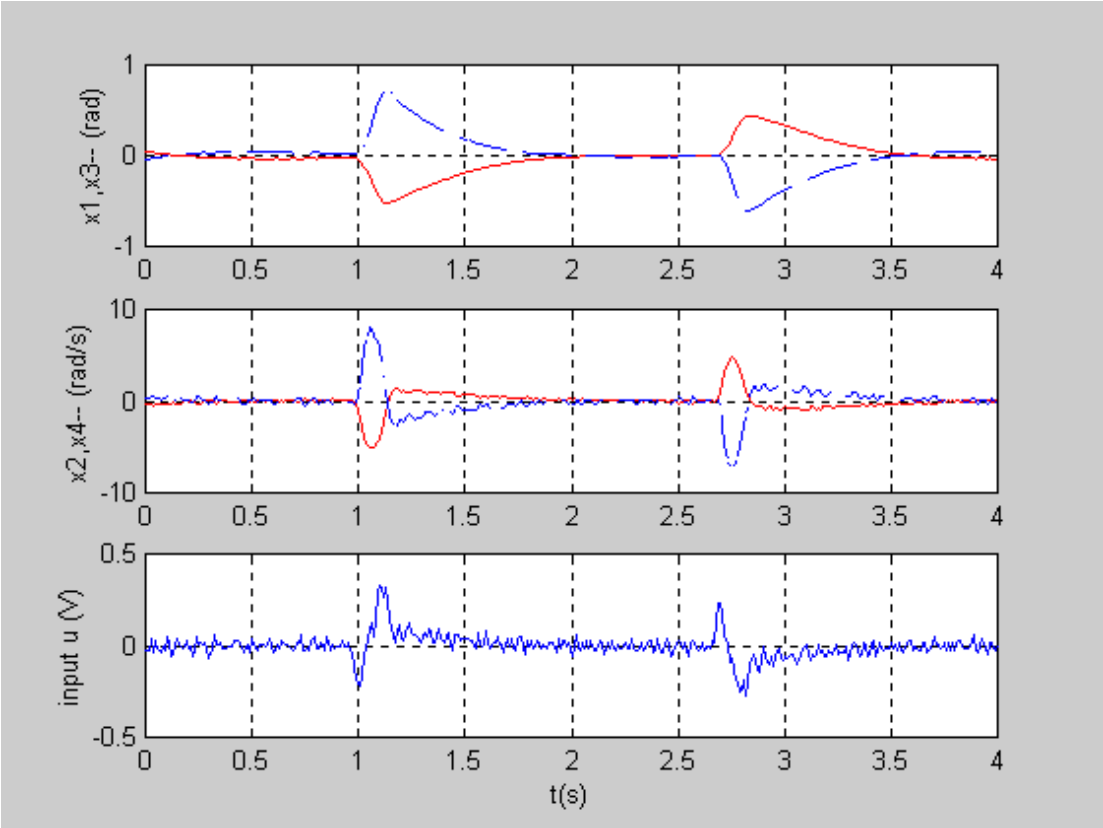


Figure 6.11 (a) Response to SDRE with q_1=q_3=r =1.

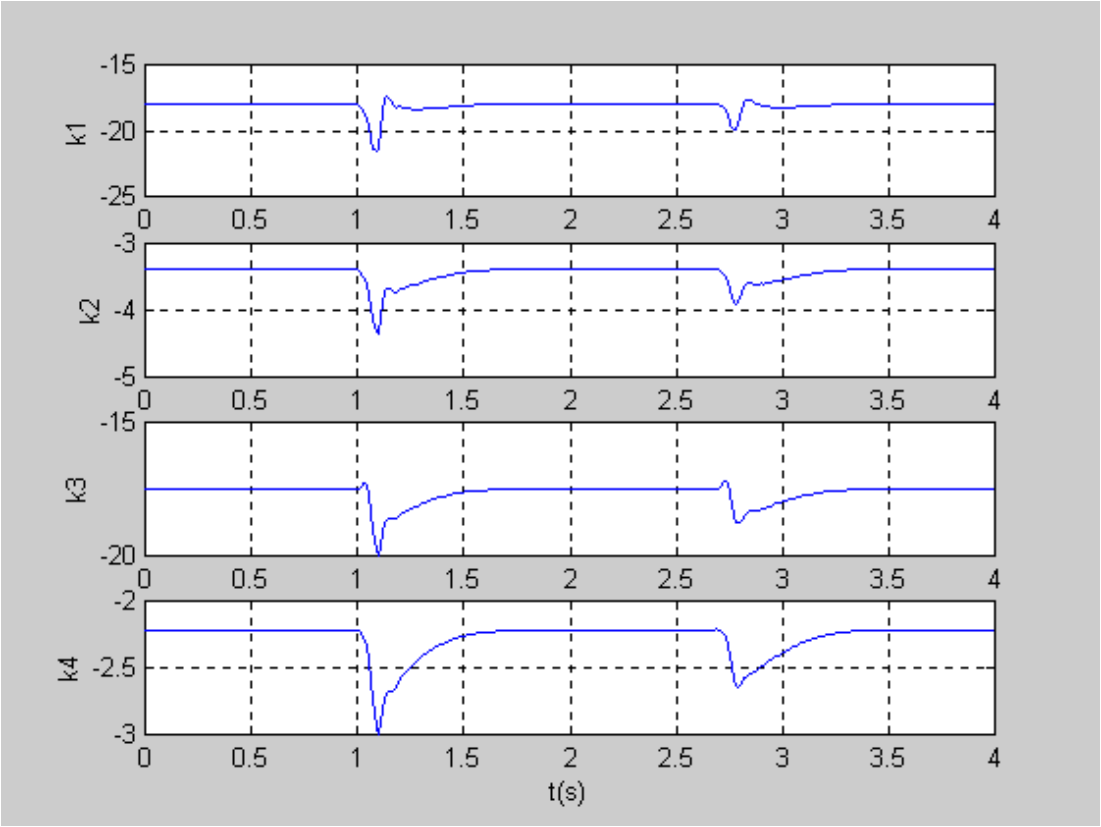


Figure 6.11 (b) Corresponding SDRE feedback gains.

In the next trial, in order to take advantage of the design flexibility offered by SDRE, the weightings were chosen as $q_1(x) = 1 + 0.2x_1^2$, $q_3(x) = 1 + 0.2x_3^2$, and $r(x) = 1/(0.2\|x\| + 0.005)$. This choice ensures that generous control effort is applied when further from the origin and less is applied when near the origin. The results are shown in Figures 6.12. It can be seen that equilibrium can be reached from larger deviations, as large as q_1= -0.9 rad and q_3=1.01 rad. The reason why the plots in Figures 6.11 are not as smooth as those in the previous figures is because a larger downsampling factor was used in monitoring the data.

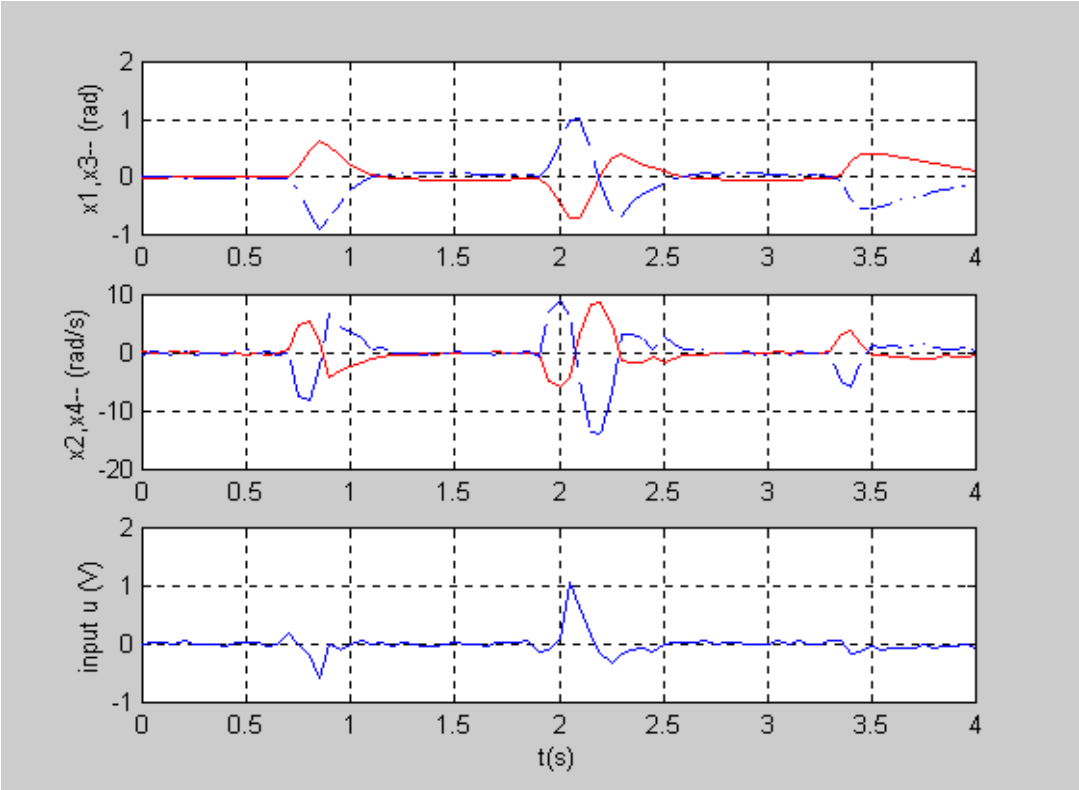


Figure 6.12 (a) Response to SDRE with $q_1(x) = 1 + 0.2x_1^2$, $q_3(x) = 1 + 0.2x_3^2$, $r(x) = 1/(0.2\|x\| + 0.005)$.

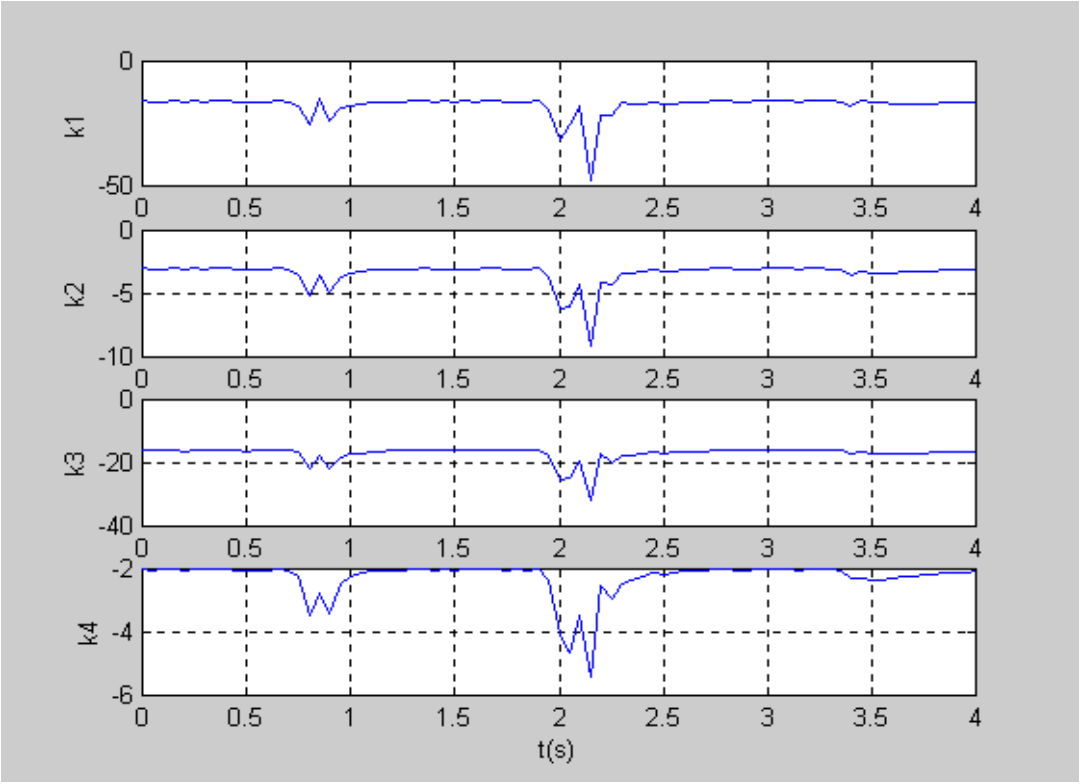


Figure 6.12 (b) Corresponding SDRE feedback gains.

6.5 Conclusions

SDRE nonlinear control method offers great design flexibility that can be exploited to meet the desired performance characteristics. Since the formulation is implicitly time-dependent, it is also very suitable for on-line implementation. Unfortunately, the "myth" against the computational complexity associated with solving the Algebraic Riccati Equation on-line has kept the number of SDRE applications from flourishing.

In this experiment, a highly nonlinear 4^{th} order system could be controlled in real-time at a sampling rate of 100 Hz around its unstable equilibrium position. It is seen that SDRE can perform as well as the classical LQR formulation with constant gains, and possibly better if the SDRE design flexibility is fully utilized. The advantages offered here outweigh the computational complexity, which is bound to cease to be an issue with the development of faster computing devices.

An alternative shortcut way for this experiment that was tried prior to the approach given in this section is presented in Appendix B. The alternative approach failed due to

hardware limitations at the time but hopefully can work in the future with increased technological capabilities.

6.6 References

[1] D.J. Block, "Mechanical Design and Control of the Pendubot", M.S. Thesis, University of Illinois at Urbana-Champaign, 1991.

[2] J.R. Cloutier, C.N. D'Souza, C.P. Mracek, "Nonlinear regulation and Nonlinear H_∞ Control via the State-Dependent Ricatti Equation Technique: Part 1, Theory; Part 2, Examples", *Proc. 1st International Conference on Nonlinear Problems in Aviation and Aerospace,* May 1996.

[3] P. Dorato, C. Abdallah, V. Cerone, "Linear Quadratic Control: An Introduction", Prentice-Hall, 1995.

[4] dSPACE User's Guide Reference Guide*,* "RTI 31: Real-Time Interface to SIMULINK", dSPACE GmbH.

[5] E.B. Erdem, A.G. Alleyne, "Globally Stabilizing Second Order Nonlinear Systems by SDRE Control", *Proc. American Control Conference,* pp.2501-2505, June 1999.

[6] Y. Huang, W-M. Lu, "Nonlinear Optimal Control: Alternatives to Hamilton-Jacobi Equation", *Proc. 35th Conference on Decision and Control,* pp.3942-3947, December 1996.

[7] M. Innocenti, F. Barolli, F. Salotti, A. Caiti, "Manipulator path control using SDRE", *Proc. American Control Conference,* pp. 3348-3352, June 2000.

[8] W. Langson, "Optimal and Suboptimal Control of a Class of Nonlinear Systems", M.S. Thesis, University of Illinois at Urbana-Champaign, 1997.

[9] W. Langson, A.G. Alleyne, "A Stability Result with Application to Nonlinear Regulation: Theory and Experiments", *Proc. American Control Conference,* pp.3051-3056, June 1999.

[10] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, "Numerical Recipes in C, The Art of Scientific Computing", 2nd Edition, Cambridge University Press, 1999.

[11] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, "Numerical Recipes in C, Example Book ", 2nd Edition, Cambridge University Press, 1999.

Chapter 7

Conclusions

In this chapter, a summary of the results of this study is given and related future research topics are recommended.

7.1 Summary of Results

Initially, a summary of mainstream nonlinear control methods was given. The methods were compared in terms of their stability, robustness, optimality, computational power requirement and intuitiveness. The main point of this summary was that each nonlinear control method had its advantages and disadvantages, and that there did not exist a unified approach that would work well for a wide variety of nonlinear systems.

Next, the State-Dependent Riccati Equation (SDRE) control method was introduced. The design flexibility offered by the method was illustrated in examples. A comprehensive survey of SDRE literature was also given. Most of the published SDRE research is simulation results, and there is a lack of solid theoretical framework. In a nutshell, the current state of SDRE research is this: simulation results show that the method works well, but it cannot be fully proved why it works.

With this motivation, two main SDRE control-related issues were explored in this thesis: stability and real-time implementation. The found results can be itemized as in the following.

(1) Conditions to globally asymptotically stabilize second order nonlinear system using SDRE were derived. The coefficient matrix $A(x)$ was chosen such that the State-Dependent Ricatti Equation could be solved and the control law could be obtained analytically. With this parametrization, it proved sufficient that the derivative of the Lyapunov function be negative semi-definite (instead of strictly negative definite) for global stability, due to LaSalle's Invariance Theorem. A condition to achieve global asymptotic stability was derived by using a quadratic Lyapunov function. For second

order systems in canonical form with constant $g(x)$, the origin can be globally asymptotically stabilized by arbitrary, constant, positive choices of q_1, q_2 and r. For more general nonlinear systems, these weighting parameters can be tuned as functions of the states to achieve global stability. Even if the weighting parameters are specific functions of the states, they can still be tuned by a (multiplicative) constant so as to achieve the desired performance, as in the linear case. The class of systems considered are feedback linearizable, so a comparison of SDRE with feedback linearization (F/L) was made. It was seen that unlike F/L, SDRE required less input and did not cancel beneficial nonlinearities. Also, F/L was unable to steer the states to the origin from large initial conditions. The theoretical stability result was successfully demonstrated on a magnetically levitated ball example, using a state-dependent state-penalty matrix.

(2) A method using overvaluing matrices was developed for estimating the region of attraction of SDRE controlled systems. The method does not require that the closed-loop system equations be known explicitly and it eliminates the need for time-domain simulations. Only the maximum and the minimum values of the feedback gains over a chosen domain S in the state-space need to be known. With this method, it is possible to reduce the complexity of the stability analysis by overvaluing higher order systems with lower order comparison systems. However, the more a system is overvalued, the more conservative the stability estimate will be.

The region of attraction estimates obtained by this method depend very much on the choice of the $A(x)$ matrix. Also, the requirement that the overvaluing matrix M must be Hurwitz in the stability region makes the estimates more conservative. Because it is known that for nonlinear systems, a point in the state-space can be stable even if the eigenvalues of the $A(x)$ matrix are not all stable. This is directly related to the choice of the parametrization $\dot{x} = A(x)x$ and is a topic for further research.

(3) Since the SDRE formulation is implicitly time-dependent, it is also very suitable for on-line implementation. Related issues were explored by a real-time control experiment using a 4^{th} order highly nonlinear plant, namely the Pendubot. The control objective was to regulate the system at one of its unstable equilibrium positions. Simulation results with the same Q and R matrices for both controllers showed that SDRE not only exhibited better

performance characteristics than LQR, but also had a larger region of attraction. Then, a computation of the ROA estimate by using the overvaluing systems method discussed in Chapter 5 was attempted. It was found that this method would not yield a nonzero ROA due to the particular choice of $A(x)$.

A 60 MHz dSPACE digital signal processor was used for the experiment. Although the hardware used for control was quite outdated, control at a sampling rate of 100 Hz could be achieved. Given that the open-loop eigenvalues of the system are 1.83 Hz and 1.01 Hz, this sampling rate is fast enough for online control. Performance-wise, the experimental results showed that SDRE could perform as well as the classical LQR formulation with constant gains, and possibly better if the SDRE design flexibility was utilized. One of the main results of this part of this study was the realization that the computational cost associated with SDRE real-time control was not as high as anticipated and that this method should not be ruled out from real-time control applications.

7.2 Recommendations for Future Work

As stated previously, many issues related to SDRE control call for more research. However, some topics that would be a direct continuation of this study can be itemized as follows:

- Explore the effect of different parametrizations $\dot{x} = A(x)x$ on the stability of the nonlinear system. This, in itself, is a general nonlinear systems problem.
- More specifically, explore the effect of different choices of the $A(x)$ matrix on the (stability) characteristics of the SDRE controlled closed-loop system.
- Explore if the region of attraction estimates obtained by the method proposed in Chapter 5 can be made less conservative.
- Find an estimate of the region of attraction of the Pendubot to match the experimental results.
- Run the Pendubot SDRE real-time control experiment with a faster processor. The code that runs on the processor is a stand-alone C-code and can be easily implemented on other processors. This would be a good measure of the efficiency of SDRE real-time control with increasing processor speed.

- In the Pendubot experiment, use a regulated disturbance of known magnitude rather than manual disturbances. This would give a more quantitative insight of the performance of both the LQR and SDRE controllers and would make a good basis for comparison.
- See if the real-time control C-code could be further optimized for processing time.

Appendix A

Computer Codes for the Pendubot Experiment

A.1 The SIMULINK Model

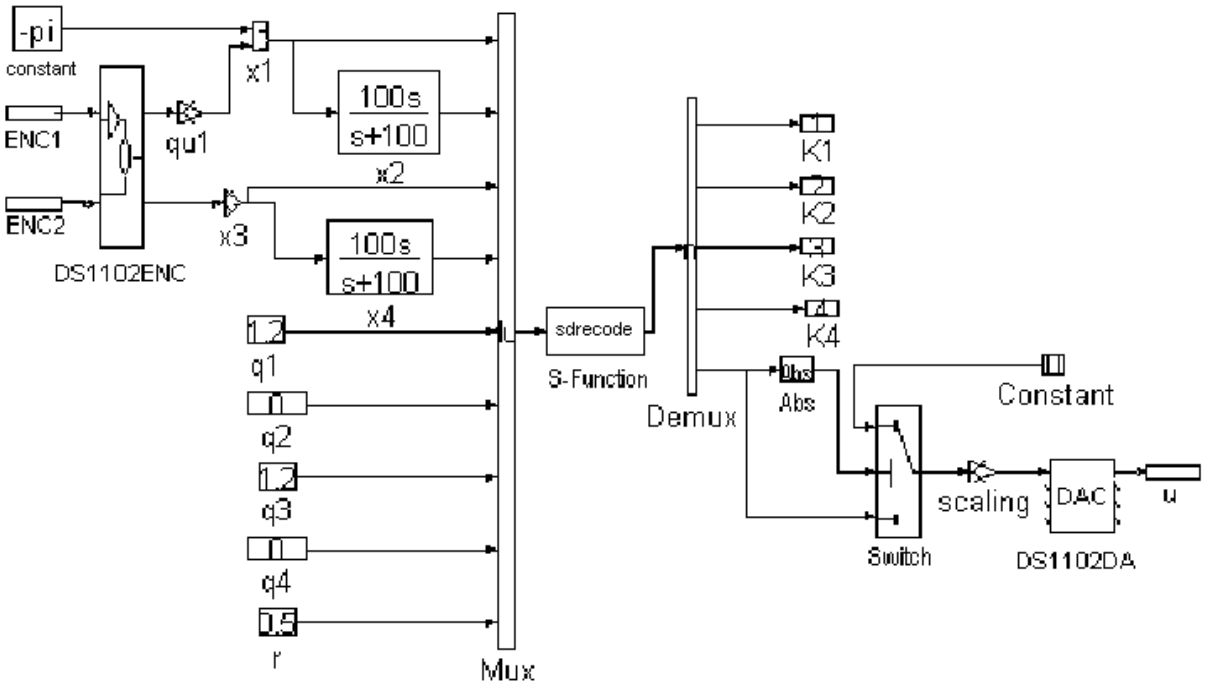


Figure A.1. The SIMULINK model.

A.2 The S-function: "sdrecode.c"

The SDRE feedback gains and control are computed in the S-function "sdrecode.c" in the SIMULINK model shown in Figure A.1. This code is presented in the following two subsections.

A.2.1 The Main Program

```
/*
 * sdrecode.c: C-MEX S-function code for controlling the Pendubot by SDRE
 * compatible with Matlab 4, to be run on DSP
```

```
* Written by Evrin Erdem, last modified 1/13/2001
* inputs are
* states: x1,x2,x3,x4; diagonal Q(x) entries q1,q2,q3,q4 and R(x)=r
* outputs are
* gains: k1,k2,k3,k4 and input: U
*
*/

#define S_FUNCTION_NAME sdrecode
#define NRANSI
#define NP 8

#include "simstruc.h"
#include <math.h>
#include "nr2.h"
#include "balanc.c"
#include "elmhes.c"
#include "hqr.c"
#include "piksr2.c"

float x1,x2,x3,x4;
float q1,q2,q3,q4,r,k1,k2,k3,k4,detM,g,o1,o2,o3,o4,o5,U;
float a21,a23,a41,a43,b21,b41,b41dw,b21dw,f1,f3,m1,m2;
float a1,a2,a3,a4,b1,b2,b3,b4,det_C,alfa1,alfa2,alfa3,alfa0,coef3;
int i,j;

float A[5][5];
float H [9][9];
float wr[NP];
float wi[NP];
float I4[5][5],T1[5][5],T2[5][5],T3[5][5],T4[5][5],T0[5][5],alfac[5][5];
```

```
/* mdlInitializeSizes - initialize the sizes array
 *
 * The sizes array is used by SIMULINK to determine the S-function block's
 * characteristics (number of inputs, outputs, states, etc.).
 */

static void mdlInitializeSizes(S)
        SimStruct *S;
{
   ssSetNumContStates(    S, 0);      /* number of continuous states */
   ssSetNumDiscStates(    S, 0);      /* number of discrete states */
   ssSetNumInputs(        S, 9);      /* number of inputs */
   ssSetNumOutputs(       S, 5);      /* number of outputs */
   ssSetDirectFeedThrough(S, 1);      /* direct feedthrough flag */
   ssSetNumSampleTimes(   S, 1);      /* number of sample times */
   ssSetNumInputArgs(     S, 0);      /* number of input arguments */
   ssSetNumRWork(         S, 0);      /* number of real work vector elements */
   ssSetNumIWork(         S, 0);      /* number of integer work vector elements */
   ssSetNumPWork(         S, 0);      /* number of pointer work vector elements */
}
/*
 * mdlInitializeSampleTimes - initialize the sample times array
 *
 * This function is used to specify the sample time(s) for your S-function.
 * If your S-function is continuous, you must specify a sample time of 0.0.
 * Sample times must be registered in ascending order.  If your S-function
 * is to acquire the sample time of the block that is driving it, you must
 * specify the sample time to be INHERITED_SAMPLE_TIME.
 */

static void mdlInitializeSampleTimes(S)
        SimStruct *S;
```

```
{
   ssSetSampleTimeEvent(S, 0, 0.0);
   ssSetOffsetTimeEvent(S, 0, 0.0);
}

static void mdlInitializeConditions(x0, S)
        double *x0;
        SimStruct *S;
{
}
/*
 * mdlOutputs - compute the outputs
 *
 * In this function, you compute the outputs of your S-function
 * block.  The outputs are placed in the y variable.
 */

static void mdlOutputs(y,x,u,S,tid)
        double *y, *x,*u;
        SimStruct *S;
        int tid;
{
        x1=(float)u[0];
        x2=(float)u[1];
        x3=(float)u[2];
        x4=(float)u[3];
        q1=(float)u[4];
        q2=(float)u[5];
        q3=(float)u[6];
        q4=(float)u[7];
        r=(float)u[8];
```

```
        g=9.804F;/* gravitational acceleration*/
/*      o1,...,o5 : system parameters compensating for masses and link lengths */
        o1=0.0308F;o2=0.0106F;o3=0.0095F;o4=0.2097F;o5=0.063F;

/*      determinant of the inertia matrix*/
        detM=o1*o2-o3*o3*(float)cos(x3)*(float)cos(x3);

/*****************************************
/*  state equations xdot=A(x)x+B(x)u(x)    *
/*****************************************

/*  compute nonzero entries of B(x) */
        b21=o2/detM;
        b41=(-o2-o3*(float)cos(x3))/detM;

/*  compute nonzero entries of A(x) and build A(x)*/
        f1=(float)sin((float)fabs(x1)+0.005F)/(float)(((float)fabs(x1)+0.005F)*detM);
        f3=-(float)sin((float)fabs(x3)+0.005F)/(((float)fabs(x3)+0.005F)*detM);
        m1=o3*g*(float)cos(x3);
        m2=m1*o3/g;

        a21=f1*(o2*o4*g-o5*m1*(float)cos(x3));
        a41=-a21+f1*m1*(o1*o5/o3-o4);
        a23=f3*(-o2*o3*(x2+x4)*(x2+x4)-o3*o3*x2*x2*(float)cos(x3)
        +m1*o5*(float)cos(x1));
        a43=-a23+f3*( m2*(-x2*x2+x4*x4+2*x2*x4)+o1*(o3*x2*x2-o5*g*(float)cos(x1))
                +0.*2*m2*x2*x2/o3);

        A[1][1]=0.0;
        A[1][2]=1.0;
        A[1][3]=0.0;
        A[1][4]=0.0;
```

```
A[2][1]=a21;
A[2][2]=0.0;
A[2][3]=a23;
A[2][4]=0.0;
A[3][1]=0.0;
A[3][2]=0.0;
A[3][3]=0.0;
A[3][4]=1.0;
A[4][1]=a41;
A[4][2]=0.0;
A[4][3]=a43;
A[4][4]=0.0;

/*      construct the Hamiltonian matrix H(x) */

        for (i=1;i<=NP;i++) {
                for (j=1;j<=NP;j++)    {
                        H[i][j]=0.0F;
                }
        }

        H[1][2]=1.0;
        H[2][1]=a21;
        H[2][3]=a23;
        H[2][6]=-b21*b21/r;
        H[2][8]= -b21*b41/r;
        H[3][4]=1.0;
        H[4][1]=a41;
        H[4][3]=a43;
        H[4][6]=H[2][8];
        H[4][8]=-b41*b41/r;
        H[5][1]=-q1;
```

```
        H[5][6]=-a21;
        H[5][8]=-a41;
        H[6][2]=-q2;
        H[6][5]=-1.0;
        H[7][3]=-q3;
        H[7][6]=-a23;
        H[7][8]=-a43;
        H[8][4]=-q4;
        H[8][7]=-1.0;

/*      begin routine for eigenvalue determination*/
        balanc(H,NP); /*balance Hamiltonian matrix*/
        elmhes(H,NP); /*put (balanced) matrix into Hessenberg form*/
        hqr(H,NP,wr,wi); /*    find eigenvalues of H by QR decomposition */
/*      wr: vector of real parts, wi: vector of imag parts of e-values*/
        piksr2(NP,wr,wi); /*sort wr and corresponding wi in ascending order*/

/*****************************************
/*   find gains k1...k4 by pole placement   *
/*****************************************

/*      calculate coefficients of closed loop polynomial alfac */
/*      alfac(s)=(s-a1-jb1)*...*(s-a4-jb4)                                  */

a1=wr[1];b1=wi[1];a2=wr[2];b2=wi[2];a3=wr[3];b3=wi[3];a4=wr[4];b4=wi[4];
alfa3=-a1-a2-a3-a4;
alfa2=a1*a3-b1*b3+a1*a2-b3*b4-b2*b4+a2*a3-b1*b4-b2*b3+a3*a4+a2*a4+a1*a4-b1*b2;
alfa1=a1*b3*b4+b1*a2*b3+a2*b3*b4+b1*b3*a4-a1*a2*a3+b1*a2*b4+b2*b3*a4+b1*b2*a4
+b2*a3*b4-a2*a3*a4+a1*b2*b4+b1*b2*a3-a1*a2*a4-a1*a3*a4+a1*b2*b3+b1*a3*b4;
alfa0=-b1*a2*b3*a4-b1*b2*a3*a4-a1*b2*b3*a4-a1*a2*b3*b4+a1*a2*a3*a4+b1*b2*b3*b4
-b1*a2*a3*b4-a1*b2*a3*b4;
```

```
/*      define the 4x4 identity matrix*/
        for (i=1;i<=4;i++) {
                 for (j=1;j<=4;j++)      {
                          I4[i][j]=0.0;
                 }
        }
        I4[1][1]=1.0;I4[2][2]=1.0;I4[3][3]=1.0;I4[4][4]=1.0;

/*      find matrix alfac=A^4+alfa3*A^3+alfa2*A^2+alfa1*A+alfa3*I4    */

        for (j=1;j<=4;j++){
                 for (i=1;i<=4;i++){
                          T0[i][j]=alfa0*I4[i][j];
                 }
        }

        for (j=1;j<=4;j++){
                 for (i=1;i<=4;i++){
                          T1[i][j]=alfa1*A[i][j];
                 }
        }


        for (j=1;j<=4;j++){
                 for (i=1;i<=4;i++){
                          T2[i][j]=alfa2*(A[i][1]*A[1][j]+A[i][2]*A[2][j]+A[i][3]*A[3][j]
                                           +A[i][4]*A[4][j]);
                 }
        }

        coef3=alfa3/alfa2;
```

```
for (j=1;j<=4;j++){
        for (i=1;i<=4;i++){
                T3[i][j]=coef3*(A[i][1]*T2[1][j]+A[i][2]*T2[2][j]+A[i][3]*T2[3][j]
                        +A[i][4]*T2[4][j]);
        }
}


for (j=1;j<=4;j++){
        for (i=1;i<=4;i++){
                T4[i][j]=(1.0F/alfa3)*(A[i][1]*T3[1][j]+A[i][2]*T3[2][j]
                        +A[i][3]*T3[3][j]+A[i][4]*T3[4][j]);
        }
}

for (i=1;i<=4;i++) {
         for (j=1;j<=4;j++)     {
                alfac[i][j]=T0[i][j]+T1[i][j]+T2[i][j]+T3[i][j]+T4[i][j];
        }
}
/*      calculate gains by K=[0 0 0 1]*inv(C)*alfac(A)
        where C is the controllability matrix*/

det_C=a41*b21*b21+b21*a43*b41-b41*a21*b21-a23*b41*b41;
b41dw=-b41/det_C;
b21dw=b21/det_C;
k1=b41dw*alfac[1][1]+b21dw*alfac[3][1];
k2=b41dw*alfac[1][2]+b21dw*alfac[3][2];
k3=b41dw*alfac[1][3]+b21dw*alfac[3][3];
k4=b41dw*alfac[1][4]+b21dw*alfac[3][4];

U=-(k1*x1+k2*x2+k3*x3+k4*x4); /*control input*/
```

```
        y[0]=k1;
        y[1]=k2;
        y[2]=k3;
        y[3]=k4;
        y[4]=U;

}
#undef NRANSI

static void mdlUpdate(x, u, S, tid)
   double *x, *u;
   SimStruct *S;
   int tid;
{
}

static void mdlDerivatives(dx, x, u, S, tid)
   double *dx, *x, *u;
   SimStruct *S;
   int tid;
{
}

/*
 * mdlTerminate - called when the simulation is terminated.
 *
 * In this function, you should perform any actions that are necessary
 * at the termination of a simulation.  For example, if memory was allocated
 * in mdlInitializeConditions, this is the place to free it.
 */
```

```
static void mdlTerminate(S)
   SimStruct *S;
{
}

/*=============================*
 * Required S-function trailer *
 *=============================*/

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

A.2.2 Functions Called in "sdrecode.c"

The functions *balanc.c, elmhes.c, hqr.c* and *piksr2.c* were taken from [1, 2] and modified to run in real-time.

```
//balanc.c  : balances a matrix
#include <math.h>
#define RADIX 2.0F
void balanc(float a[9][9], int n)
{
        int last,j,i;
        float s,r,g,f,c,sqrdx;
        sqrdx=RADIX*RADIX;
        last=0;
        while (last == 0) {
                last=1;
                for (i=1;i<=n;i++) {
```

```
                    r=c=0.0F;
                    for (j=1;j<=n;j++)
                        if (j != i) {
                            c += (float)fabs(a[j][i]);
                            r += (float)fabs(a[i][j]);
                        }
                    if (c && r) {
                        g=(float)r/RADIX;
                        f=1.0F;
                        s=c+r;
                        while (c<g) {
                            f *= RADIX;
                            c *= sqrdx;
                        }
                        g=(float)r*RADIX;
                        while (c>g) {
                            f /= RADIX;
                            c /= sqrdx;
                        }
                        if ((c+r)/f < 0.95*s) {
                            last=0;
                            g=(float)1.0/f;
                            for (j=1;j<=n;j++) a[i][j] *= g;
                            for (j=1;j<=n;j++) a[j][i] *= f;
                        }
                    }
                }
            }


}
#undef RADIX
```

```
/* (C) Copr. 1986-92 Numerical Recipes Software . */
/*  modified by Evrin Erdem 1/13/2001                              */

//elmhes.c : puts a matrix into Hessenberg form
#include <math.h>
#define SWAP(g,h) {y=(g);(g)=(h);(h)=y;}
void elmhes(float a[9][9], int n)
{
        int m,j,i;
        float y,x;
        for (m=2;m<n;m++) {
                x=0.0F;
                i=m;
                for (j=m;j<=n;j++) {
                        if ((float)fabs(a[j][m-1]) > (float)fabs(x)) {
                                x=a[j][m-1];
                                i=j;
                        }
                }
                if (i != m) {
                        for (j=m-1;j<=n;j++) SWAP(a[i][j],a[m][j])
                        for (j=1;j<=n;j++) SWAP(a[j][i],a[j][m])
                }
                if (x) {
                        for (i=m+1;i<=n;i++) {
                                if ((y=a[i][m-1]) != 0.0) {
                                        y /= x;
                                        a[i][m-1]=y;
                                        for (j=m;j<=n;j++)
                                                a[i][j] -= y*a[m][j];
                                        for (j=1;j<=n;j++)
```

```
                                        a[j][m] += y*a[j][i];
                                }
                        }
                }
        }

}

#undef SWAP

/* (C) Copr. 1986-92 Numerical Recipes Software . */
/*   modified by Evrin Erdem 1/13/2001                              */

//hqr.c : finds e-values of a matrix by QR decomposition

#include <math.h>
#define NRANSI
#define SIGN(a,b) ((b) >= 0.0 ? fabs(a) : -fabs(a))
void nrerror(char error_text[])
{
}
void hqr(float a[9][9], int n, float wr[], float wi[])
{
        int nn,m,l,k,j,its,i,mmin;
        float z,y,x,w,v,u,t,s,r,q,p,anorm;

        anorm=(float)fabs(a[1][1]);
        for (i=2;i<=n;i++)
                for (j=(i-1);j<=n;j++)
                        anorm += (float)fabs(a[i][j]);
        nn=n;
        t=0.0F;
```

```
while (nn >= 1) {
        its=0;
        do {
                for (l=nn;l>=2;l--) {
                        s=(float)fabs(a[l-1][l-1])+(float)fabs(a[l][l]);
                        if (s == 0.0) s=anorm;
                        if (((float)fabs(a[l][l-1]) + s) == s) break;
                }
                x=a[nn][nn];
                if (l == nn) {
                        wr[nn]=x+t;
                        wi[nn--]=0.0F;
                } else {
                        y=a[nn-1][nn-1];
                        w=a[nn][nn-1]*a[nn-1][nn];
                        if (l == (nn-1)) {
                                p=0.5F*(y-x);
                                q=p*p+w;
                                z=(float)sqrt((float)fabs(q));
                                x += t;
                                if (q >= 0.0) {
                                        z=p+(float)SIGN(z,p);
                                        wr[nn-1]=wr[nn]=x+z;
                                        if (z) wr[nn]=x-w/z;
                                        wi[nn-1]=wi[nn]=0.0F;
                                } else {
                                        wr[nn-1]=wr[nn]=x+p;
                                        wi[nn-1]= -(wi[nn]=z);
                                }
                                nn -= 2;
                        } else {
                                if (its == 30) nrerror("Too many iterations in hqr");
```

```
                if (its == 10 || its == 20) {
                        t += x;
                        for (i=1;i<=nn;i++) a[i][i] -= x;
                        s=(float)fabs(a[nn][nn-1])+(float)fabs(a[nn-
1][nn-2]);
                        y=x=0.75F*s;
                        w = -0.4375F*s*s;
                }
                ++its;
                for (m=(nn-2);m>=l;m--) {
                        z=a[m][m];
                        r=x-z;
                        s=y-z;
                        p=(r*s-w)/a[m+1][m]+a[m][m+1];
                        q=a[m+1][m+1]-z-r-s;
                        r=a[m+2][m+1];
                        s=(float)fabs(p)+(float)fabs(q)+(float)fabs(r);
                        p /= s;
                        q /= s;
                        r /= s;
                        if (m == l) break;
                        u=(float)fabs(a[m][m-
1])*((float)fabs(q)+(float)fabs(r));
                        v=(float)fabs(p)*((float)fabs(a[m-1][m-
1])+(float)fabs(z)+(float)fabs(a[m+1][m+1]));
                        if ((float)(u+v) == v) break;
                }
                for (i=m+2;i<=nn;i++) {
                        a[i][i-2]=0.0F;
                        if (i != (m+2)) a[i][i-3]=0.0F;
                }
                for (k=m;k<=nn-1;k++) {
```

```
                if (k != m) {
                        p=a[k][k-1];
                        q=a[k+1][k-1];
                        r=0.0F;
                        if (k != (nn-1)) r=a[k+2][k-1];
                        if
((x=(float)fabs(p)+(float)fabs(q)+(float)fabs(r)) != 0.0) {
                                p /= x;
                                q /= x;
                                r /= x;
                        }
                }
                if ((s=(float)SIGN(sqrt(p*p+q*q+r*r),p)) != 0.0)
{
                        if (k == m) {
                                if (l != m)
                                a[k][k-1] = -a[k][k-1];
                        } else
                                a[k][k-1] = -s*x;
                        p += s;
                        x=p/s;
                        y=q/s;
                        z=r/s;
                        q /= p;
                        r /= p;
                        for (j=k;j<=nn;j++) {
                                p=a[k][j]+q*a[k+1][j];
                                if (k != (nn-1)) {
                                        p += r*a[k+2][j];
                                        a[k+2][j] -= p*z;
                                }
                                a[k+1][j] -= p*y;
```

```
                                        a[k][j] -= p*x;
                                }
                                mmin = nn<k+3 ? nn : k+3;
                                for (i=l;i<=mmin;i++) {
                                        p=x*a[i][k]+y*a[i][k+1];
                                        if (k != (nn-1)) {
                                                p += z*a[i][k+2];
                                                a[i][k+2] -= p*r;
                                        }
                                        a[i][k+1] -= p*q;
                                        a[i][k] -= p;
                                }
                        }
                }
        }
    } while (l < nn-1);
  }

}

#undef NRANSI

/* (C) Copr. 1986-92 Numerical Recipes Software . */
/*   modified by Evrin Erdem 1/13/2001                    */

//piksr2.c
void piksr2(int n, float arr[], float brr[])
{
        int i,j;
        float a,b;
```

```
        for (j=2;j<=n;j++) {
                a=arr[j];
                b=brr[j];
                i=j-1;
                while (i > 0 && arr[i] > a) {
                        arr[i+1]=arr[i];
                        brr[i+1]=brr[i];
                        i--;
                }
                arr[i+1]=a;
                brr[i+1]=b;
        }
}
/* (C) Copr. 1986-92 Numerical Recipes Software . */
```

A.3 References

[1] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, "Numerical Recipes in C, The Art of Scientific Computing", 2nd Edition, Cambridge University Press, 1999.

[2] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, "Numerical Recipes in C, Example Book ", 2nd Edition, Cambridge University Press, 1999.

Appendix B

An SDRE Real-Time Control Attempt Using Existing dSPACE Software

The main motivation of this approach was the idea of solving the Algebraic Riccati Equation in MATLAB by commands like "LQR" or "ARE" during the real-time experiment. This would save a lot of coding effort compared to the alternative approach which was to hand-code the solution of the Algebraic Riccati Equation in C which is the approach presented in Chapter 6. The control gains would be calculated in MATLAB environment and would then be sent back to the DSP environment where they would be multiplexed with the states to compute the input. However, this approach to do real-time control failed due to the reasons explained in the sequel.

B.1 The Working Principle

The dSPACE TRACE and COCKPIT modules to monitor data and tune parameters, respectively, both work in the DSP environment. Data can be transferred between DSP and MATLAB environments by using MLIB [1] and MTRACE [2] modules of dSPACE.

The MATLAB-DSP Interface Library (MLIB) allows access to the dSPACE DSP hardware from within MATLAB's workspace. MLIB provides basic functions for reading and writing data to dSPACE processor boards. These functions are suited to modify parameters in the DSP program currently running, or to send test sequences of data to the DSP application. They can be called from standard MATLAB M-files. Thus the advantages of numerical tools running under MATLAB can be integrated into the real-time control program.

The Real-Time Trace Module for MATLAB (MTRACE) provides real-time data capture capabilities for the dSPACE DSP processor board. Via MTRACE, direct data transfer to MATLAB workspace is possible. MTRACE functions can be called from standard MATLAB M-files. Besides MLIB, MTRACE is another direct link between dSPACE DSP hardware and numerical toolboxes provided by MATLAB.

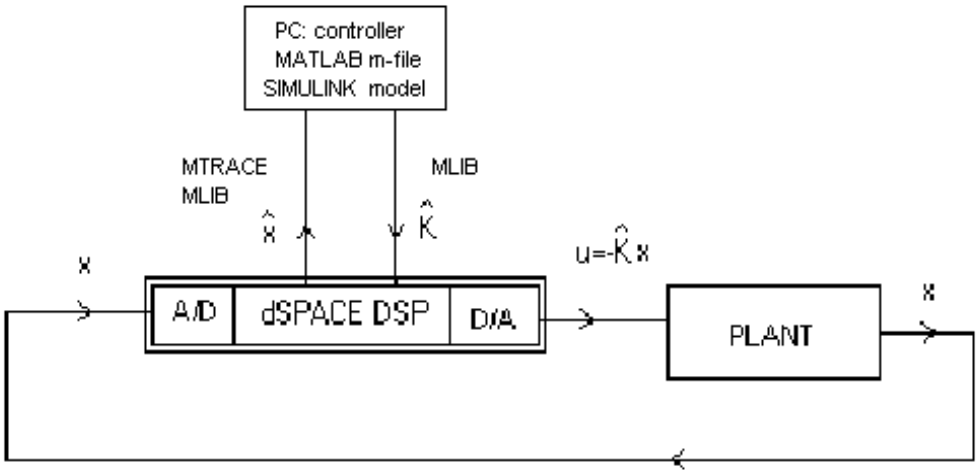


Figure B.1. Real-time implementation setup

The setup used for the Pendubot experiment using this approach is shown in Figure B.2. The measurable states of the state vector *x* are sent to the dSPACE DSP board, where they are digitized by the built in analog-to-digital converter at a certain rate. The nonmeasurable states, which are link velocities in this experiment, are computed in the SIMULINK graphical model.

First, by MLIB function calls, the addresses of the elements of the gain $\hat{K}$ and of the *Q* and *R* matrices are assigned in the DSP memory. Then the states, $\hat{x}$, are captured and transferred to the MATLAB environment by MTRACE function calls. In the controller PC, the gain $\hat{K}$ is determined by solving the Algebraic Riccati Equation in MATLAB environment using the MATLAB commands "LQR" or "ARE". The gains thus calculated are written to their already assigned addresses in the DSP memory by MLIB function calls. Once the gains $\hat{K}$ are received by the DSP, they are multiplexed by the states ($u = -\sum \hat{k}_i x_i$) to form the input signal *u*. Naturally, the input signal *u* is updated at a slower rate than the sampling rate. The input signal *u* is converted to an analog voltage signal by the digital-to-analog converter in the dSPACE board before it is finally sent to the plant. All MTRACE and MLIB function calls, as well as the computation of the gains take place in a single MATLAB M-file which runs together with the SIMULINK code. The MATLAB file *pendsdre.m*, is presented in section B.3.

B.2 Results

The reason why this approach did not work was because MLIB delivered the gains from MATLAB environment to the DSP not in real-time, but at arbitrary times. Unlike MTRACE which transfers data to MATLAB workspace in real-time, MLIB was not build to work in real-time. Thus, the gains do not vary synchronously with the states, as should have been the case. This is clearly seen in Figure B.2. The first two plots are of the link positions. The third plot is of the first gain, k_1, as a function of time. The variations of other three gains are similar to that of k_1. It is seen that although the positions vary continuously during the 4s interval, the gain k_1 is updated only at two brief intervals and is constant at other times. Moreover, the updating times of k_1 are totally arbitrary and not in sync with the state profiles. Clearly, this hardware setup (due to MLIB) is not useful in real-time control.

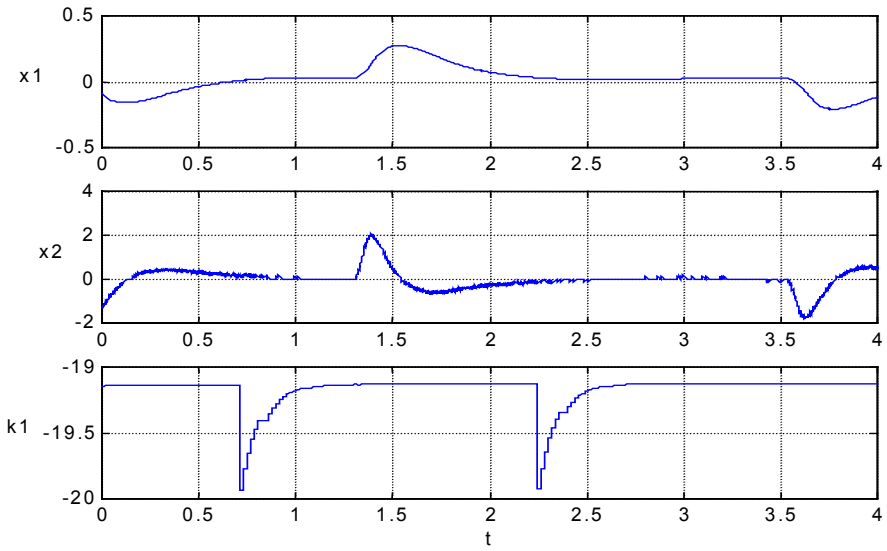


Figure B.2. Results of the first Pendubot experiment.

B.3 The code "pendsdre.m"

```
%pendsdre.m
%to be run on DSP in real-time
%calculates SDRE gains for the Pendubot
%Evrin Erdem 6/20/2000
%data transfer btw Matlab and DSP provided by MLIB and MTRACE modules
```

```
mlib('SelectBoard','ds1102');
mtrc31('SelectBoard');

mlibini;
mlibvxd;
mtrcvxd;

tic;
while (toc)<120,
k1addr=mlib('GetSimAddr','P[k1]');
k2addr=mlib('GetSimAddr','P[k2]');
k3addr=mlib('GetSimAddr','P[k3]');
k4addr=mlib('GetSimAddr','P[k4]');

N=500;
mtrc31('TraceVars','rti B[x1]','rti B[x2]','rti B[x3]','rti B[x4]');
mtrc31('SetFrame',[],1,0,N);
mtrc31('SetTrigger','off');

mtrc31('StartCapture');
    while mtrc31('CaptureState')~=0,
    drawnow;
    end
rawdata=mtrc31('FetchData');
x1=mtrc31('AssignData',rawdata,'rti B[x1]');
x2=mtrc31('AssignData',rawdata,'rti B[x2]');
x3=mtrc31('AssignData',rawdata,'rti B[x3]');
x4=mtrc31('AssignData',rawdata,'rti B[x4]');
m=mtrc31('AssignData',rawdata,'rti B[m]');

%-----------calculate gains by solving sdre-----------------------

for i=1:N,
    k1(i)=mlib('ReadF',k1addr);
    k2(i)=mlib('ReadF',k2addr);
    k3(i)=mlib('ReadF',k3addr);
    k4(i)=mlib('ReadF',k4addr);
    det(i)=o1*o2-o3^2*cos(x3(i))*cos(x3(i));

    if (abs(x1(i)))<0.05
    a21(i)=(o2*o4*g-o3*o5*g*cos(x3(i))*cos(x3(i)))/(det(i));
    a41(i)=(-o2*o4*g+o3*g*cos(x3(i))*(-o4+o5*cos(x3(i)))+o1*o5*g*cos(x3(i)))/(det(i));
    else
    a21(i)=sin(x1(i))*(o2*o4*g-o3*o5*g*cos(x3(i))*cos(x3(i)))/(x1(i)*det(i));
    a41(i)=sin(x1(i))*(-o2*o4*g+o3*g*cos(x3(i))*(-
o4+o5*cos(x3(i)))+o1*o5*g*cos(x3(i)))/(x1(i)*det(i));
    end
```

```
    if (abs(x3(i)))<0.05
    a23(i)=-(-2*o2*o3*x2(i)*x4(i)-o2*o3*(x4(i))^2-o2*o3*(x2(i))^2-
o3^2*(x2(i))^2*cos(x3(i))+o3*o5*g*cos(x1(i))*cos(x3(i)))/(det(i));
    a43(i)=-
(2*o2*o3*x2(i)*x4(i)+o2*o3*(x4(i))^2+2*o3^2*x2(i)*x4(i)*cos(x3(i))+o3^2*(x4(i))^2*cos(
x3(i))+(o1+o2+2*cos(x3(i)))*o3*(x2(i))^2-o1*o5*g*cos(x1(i))-
o3*o5*g*cos(x1(i))*cos(x3(i)))/(det(i));
    else
    a23(i)=-sin(x3(i))*(-2*o2*o3*x2(i)*x4(i)-o2*o3*(x4(i))^2-o2*o3*(x2(i))^2-
o3^2*(x2(i))^2*cos(x3(i))+o3*o5*g*cos(x1(i))*cos(x3(i)))/(x3(i)*det(i));
    a43(i)=-
sin(x3(i))*(2*o2*o3*x2(i)*x4(i)+o2*o3*(x4(i))^2+2*o3^2*x2(i)*x4(i)*cos(x3(i))+o3^2*(x4(
i))^2*cos(x3(i))+(o1+o2+2*cos(x3(i)))*o3*(x2(i))^2-o1*o5*g*cos(x1(i))-
o3*o5*g*cos(x1(i))*cos(x3(i)))/(x3(i)*det(i));
    end

    A=[0 1 0 0; a21(i) 0 a23(i) 0; 0 0 0 1; a41(i) 0 a43(i) 0];
    B=[0 ;o2; 0 ;-o2-o3*cos(x3(i))]/det(i);
    Q=diag([1 0 1 0]);
    R=1;
    K=lqr(A,B,Q,R);

    k1(i)=K(1);
    k2(i)=K(2);
    k3(i)=K(3);
    k4(i)=K(4);
    mlib('WriteF',k1addr,k1(i));
    mlib('WriteF',k2addr,k2(i));
    mlib('WriteF',k3addr,k3(i));
    mlib('WriteF',k4addr,k4(i));

end
end

end
```

B.4 References

[1] “MLIB: MATLAB-DSP Interface Library”, *dSPACE User’s Guide Reference Guide,* dSPACE GmbH.

[2] “MTRACE: Real-Time TRACE Module for MATLAB”, *dSPACE User’s Guide Reference Guide,* dSPACE GmbH.

Vita

Evrin Bilge Erdem was born on July 14, 1973 in Ankara, Turkey. She received her Bachelor of Science degree in Mechanical Engineering in 1994 from Middle East Technical University in Ankara, Turkey. During her master's studies, she worked on robotics while working as a teaching/research assistant in the same department. She received her Master of Science degree in 1996.

In the fall of 1996, she came to the University of Illinois to pursue a doctoral degree in Mechanical Engineering under the supervision of Prof. Andrew Alleyne. For her presentation in 1999 American Control Conference she received a Best Presentation Award. Her research interests are mainly in the area of dynamic systems and control, with emphasis on nonlinear control, robotics and real-time control.

Upon graduation, Evrin will join Emmeskay Incorporated in Plymouth, Michigan as a controls engineer.