Modern Control Methods for First Order Hyperbolic Partial Differential Equations

by

Navid Alavi Shoushtari

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

Master of Science

in

Process Control

Department of Department of Chemical and Materials Engineering

University of Alberta

Abstract

This work is focused on two control methods for first order hyperbolic partial differential equations (PDE). The first method investigated is output regulation by employing the internal model control (IMC) principle where the controller generates the internal dynamic of the exosystem in order to asymptotically track the trajectory and reject the disturbances. This method is implemented on both systems with in-domain and boundary actuation. The actuator is distributed throughout the system in the first approach, while it is limited at the boundary in the second one.The second method revolves around designing full state feed back controller using so-called backstepping method. Backstepping converts the unstable system to a stable target plant by an integral transformation. This method is assessed for a general first order hyperbolic PDE. Finally, the backstepping method is expanded to a special case in which the boundaries are coupled. This case represents tubular reactors with recycle and exhibits interesting response under controller implementation. Results show that both methods are able to successfully stabilize first order hyperbolic PDEs at the origin. Results are illustrated with simulations.

Acknowledgements

I would first like to thank my thesis advisor Dr. Dubljevic of the Chemical and Materials Engineering Department at University of Alberta. The door to Dr. Dubljevic office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it. I must also express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Contents

List of Figures

Chapter 1

Introduction

Partial differential equations (PDE) have a pronounced standing in chemical engineering. Tubular reactors, fixed and fluidized beds, extraction columns, and chromatographers are just a handful of processes amongst all which can be modelled by the PDEs [3]. First order hyperbolic PDEs are encountered mostly when the effect of advection (or convective term) is more dominant and the diffusion (or conduction) is neglected. This particular scenario can be considered in modelling different phenomena or chemical processes such as a tubular reactor [22, 27, 28]. Bearing that in mind, we concentrate on two methods, namely output regulation and backstepping to investigate the possibility of designing controllers for the aforementioned systems. These two methods are briefed in the next two sections. The elucidations and further elaborations are also found in the subsequent chapters.

1.1 Thesis Outline

In chapter I the basics of output regulation with internal model principle and backstepping methods are described. General derivation for each method is provided and results are illustrated with examples. Chapter II explores the output regulation of first order hyperbolic partial differential equations in details. Internal model control will be applied for infinite dimensional systems and in-domain or boundary controller will be designed. Chapter III is about applying backstepping method on a general form of first order hyperbolic system to develop a boundary controller. Two sample systems will be simulated in this chapter and the performance of the system will be analyzed. Chapter IV will expand the backstepping method to a special case of first order hyperbolic PDEs when the boundaries are coupled. This system which is a generalized representation of tubular reactors with recycle will be illustrated through a simulation.

1.2 Output Regulation by Internal Model Principle

One of the central problems in the control theory is output regulation. It concerns about how to control a system to achieve asymptotic tracking of a prescribed signal and/or asymptotic

rejection of disturbances. This goal can be achieved by four potential methods; dynamic inversion, adaptive tracking, linear quadratic regulation, and internal model control. Dynamic inversion generates the exact desired trajectory; nevertheless, it requires the perfect knowledge of initial state and the reference signal which may not be plausible in presence of uncertainties [2, 18, 25, 30]. Adaptive tracking is based on online adaptation of the controller with the update of system parameters. This approach successfully copes with the uncertainties in the plant model; however, it relies on exact knowledge of the trajectory to be tracked [4, 5]. Thus, this method cannot be used when dealing with tracking of unknown trajectories [2]. The idea of linear quadratic regulation is to find a full state feedback to minimize a cost function or optimize the system performance [19, 23]. This method works based upon full state feedback which might not be available sometimes. Although, observer may be designed to resolve this issue, it makes the problem too complicated. Furthermore, if there are uncertainties associated with the model, the controller likely will not work properly [14]. Internal model control on the other hand can approach the problems having both uncertainties in model parameters and reference trajectory [2]. Internal model control utilizes a set of differential equations called exosystem which generates the reference signals and disturbance ones. Exosystem is able to produce vast variety of common signals such as step and/or combinations of sinusoidal functions as reference output and/or disturbance [26]. Internal model control guarantees stability and robustness of the system with respect to uncertainties the system and exosignals as long as the trajectories and disturbances produced by the exosystem belong to same fixed dynamical system, and the controller is able to suitably incorporate the dynamic of internal model [2, 13]. This method which has been introduced first by Garcia and Morari, 1982 in a series articles, has remained for years for regulation of SISO and MIMO systems and tuning of PID controllers [11, 12, 15–17, 29]. Obymes, 2000 extended the method to linear distributed parameter systems [6]. Furthermore, coupling of output regulation with Backstepping method is presented by Deutscher, 2015 [9, 10]. The output regulation method is illustrated below for a finite dimensional system.

1.2.1 Output Regulation of Finite Systems with IMC

Posing the Problem

Consider the following mass spring damper system:

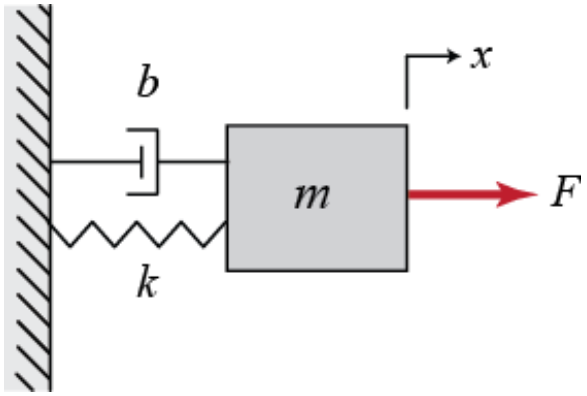

Figure 1.1: Mass spring damper system

which can be stated as;

$$\text{Plant:} m\ddot{x} + b\dot{x} + kx = F \tag{1.1}$$

If the system is to follow a trigonometric signal and reject a step signal, the exosystem could be stated as follows;

$$\begin{array}{ll} \text{Exosystem:} & \dot{v} = Sv \\ \text{Disturbance:} & Stepfunction \\ \text{Trajectory:} & sin(2t) \end{array} \tag{1.2}$$

Here, exosystem is responsible for gererating *cos(2t)-sin(2t)* as the disturbance and step trajectory signals. What we are willing to do is finding a full state feedback to track the trajectory and reject the disturbance.

Solution

First, we try to find the equivalent state space for our plant. Using the change of variables

$$\begin{array}{l} x = x_1 \\ \dot{x} = x_2 \end{array} \tag{1.3}$$

the state space is found:

$$\begin{bmatrix} \dot{x_1} \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k/m & -b/m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} F. \tag{1.4}$$

let define F as a combination of the input and the disturbance imposed to the plant:

$$F = u + mW \tag{1.5}$$

where u is the input and and W is the disturbance. The coefficient m is just added to simplify the derivations ahead. This leads to:

$$\begin{bmatrix} \dot{x_1} \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k/m & -b/m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u + \begin{bmatrix} 0 \\ 1 \end{bmatrix} W \tag{1.6}$$

or

$$\begin{cases} \dot{x} &= \bar{A}x + \bar{B}u + \bar{D}W \\ y &= \bar{C}x \end{cases} \tag{1.7}$$

where by defining $m = 1$, $b = 1$, and $k = 1$ the parameters associated become:

$$\bar{A} = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix}, \qquad \bar{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad \bar{D} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad \bar{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{1.8}$$

We seek to find a proper exosystem here. The disturbance which is a step function is equivalent to

$$\dot{v_1} = [0][v_1] \tag{1.9}$$

and the trajectory can be expressed as:

$$\begin{bmatrix} \dot{v_2} \\ v_3 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ -2 & 0 \end{bmatrix} \begin{bmatrix} v_2 \\ v_3 \end{bmatrix}. \tag{1.10}$$

Combining trajectory and disturbance together we come up with:

$$\dot{v} = Sv = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & -2 & 0 \end{bmatrix} v. \tag{1.11}$$

Thus, one can represent exosystem as follows:

$$\begin{cases} \dot{v} = Sv \\ W = Pv \\ y_{tr} = Qv \end{cases} \tag{1.12}$$

where y_{tr} is the trajectory to be tracked and

$$P = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \qquad Q = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}. \tag{1.13}$$

Introducing error and the full state feed back as a combination of plant and exosystem signals,

$$u = Kx + Ev \tag{1.14}$$

$$e = y - y_{tr} = \bar{C}x - Qv \tag{1.15}$$

we end up with the following system:

$$\begin{cases} \dot{x} = (\bar{A} + \bar{B}K)x + (\bar{D}P + \bar{B}E)v \\ \dot{v} = Sv \\ e = Cx - Qv \end{cases} \tag{1.16}$$

where the open loop system i.e. when $v = 0$ must be asymptotically stable around the origin and the error should converge to zero as $t \to \infty$. In other words, we have two objectives:

$$\bar{A} + \bar{B}K \tag{1.17}$$

must have negative eigenvalues, and

$$\lim_{t\to\infty} e = 0. \tag{1.18}$$

It has been proved that these two conditions only can be met if there exists a mapping between the plant state x and the exosystem state v such that:

$$x = \Pi v. \tag{1.19}$$

Imposing the mapping (1.19) into the system (1.16) results:

$$\begin{cases} \Pi S = \bar{A}\Pi + \bar{B}\Gamma + \bar{D}P \\ C\Pi = Q \end{cases} \tag{1.20}$$

which is the constrained Sylvester equation where e assumed to be zero, and $\Gamma = K\Pi + E$. Defining Π and Γ as:

$$\Pi = \begin{bmatrix} \pi_{11} & \pi_{12} & \pi_{13} \\ \pi_{21} & \pi_{22} & \pi_{23} \end{bmatrix}, \qquad \Gamma = \begin{bmatrix} \gamma_1 & \gamma_2 & \gamma_3 \end{bmatrix} \tag{1.21}$$

and substituting all the variables into the eq. (19) Π and Γ can be calculated as shown below:

$$\begin{bmatrix} \pi_{11} & \pi_{12} & \pi_{13} \\ \pi_{21} & \pi_{22} & \pi_{23} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & -2 & 0 \end{bmatrix} = \\ \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} \pi_{11} & \pi_{12} & \pi_{13} \\ \pi_{21} & \pi_{22} & \pi_{23} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} \gamma_1 & \gamma_2 & \gamma_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{1.22}$$

and

$$\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \pi_{11} & \pi_{12} & \pi_{13} \\ \pi_{21} & \pi_{22} & \pi_{23} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{1.23}$$

or,

$$\begin{cases} \begin{bmatrix} 0 & -2 & 0 \\ 0 & -2\pi_{23} & 2\pi_{22} \end{bmatrix} = \begin{bmatrix} \pi_{21} & \pi_{22} & \pi_{23} \\ 1-\pi_{21}+\gamma_1 & -\pi_{22}+\gamma_2 & -1-\pi_{23}+\gamma_3 \end{bmatrix} \\ \begin{bmatrix} \pi_{11} & \pi_{12} & \pi_{13} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \end{cases} \tag{1.24}$$

Hence,

$$\begin{array}{lll} \pi_{11} = 0 & \pi_{12} = 0 & \pi_{13} = 1 \\ \pi_{21} = 0 & \pi_{22} = -2 & \pi_{23} = 0 \\ \gamma_1 = -1 & \gamma_2 = -2 & \gamma_3 = -3 \end{array} \tag{1.25}$$

or,

$$\begin{array}{l} \Pi = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -2 & 0 \end{bmatrix} \\ \Gamma = \begin{bmatrix} -1 & -2 & -3 \end{bmatrix} \end{array} . \tag{1.26}$$

As it is said before, eq. (16) must have negative eigenvalues. Below, this is satisfied:

$$\bar{A} + \bar{B}K = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} k_1 & k_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ k_1 - 1 & k_2 - 1 \end{bmatrix}. \tag{1.27}$$

Eigenvalues of the above matrix is found through following determinant:

$$\begin{vmatrix} -\lambda & 1 \\ k_1 - 1 & k_2 - 1 - \lambda \end{vmatrix} = \lambda^2 - (k_2 - 1)\lambda - (k_1 - 1) = 0 \tag{1.28}$$

where in order to have negative eigenvalues:

$$\begin{aligned} &\lambda_1.\lambda_2 = 1 - k_1 > 0 \\ &\lambda_1 + \lambda_2 = k_2 - 1 < 0 \end{aligned} \tag{1.29}$$

must hold. Therefore, one needs to choose gains such that:

$$k_1 < 1 \qquad k_2 < 1 \tag{1.30}$$

At last, E can be calculated as:

$$E = \Gamma - K\Pi \tag{1.31}$$

which make us able to find the controller and plant response. This is demonstrated in the next section.

Numerical Solution, and the Response of the System

We wish to solve the system (1.16) for any desired initial condition to find the response of system. Let's start with finding the state of exosystem,

$$\dot{v} = Sv \qquad v(0) = v_0 \tag{1.32}$$

which is a first order ordinary differential equation with the solution of

$$v(t) = e^{St}v(0). \tag{1.33}$$

In order to have $v(t)$ found, one should calculate e^{St}. To do so, either of *Caley-Hamilton* theorem or the MATLAB built in function *expm* can be used. Having v calculated, we approach the differential equation for x. The equation

$$\dot{x} = (\bar{A} + \bar{B}K)x + (\bar{D}P + \bar{B}E)v \qquad x(0) = x_0 \tag{1.34}$$

has the solution of:

$$x(t) = e^{(\bar{A}+\bar{B}K)t}x(0) + \int_0^t e^{(\bar{A}+\bar{B}K)(t-\tau)}(\bar{B}E + \bar{D}P)v(\tau)d\tau. \tag{1.35}$$

Again, the exponentials must be calculated numerically. Modified Euler method has been used here as follows; Let

$$\dot{x}_i = F(x_i) \tag{1.36}$$

then

$$x_{i+1} = x_i + \Delta t F(x_i) \tag{1.37}$$

and then modifying x_{i+1}

$$x_{i+1} = x_i + \frac{\Delta t}{2}(F(x_i) + F(x_{i+1})). \tag{1.38}$$

This way $x(t)$ can be found accurately. The simulation results are depicted in the figures below. Figure 1.2 shows the responses of both states of the system. The reference trajectory is a trigonometric function, therefore the states have oscillatory behavior.

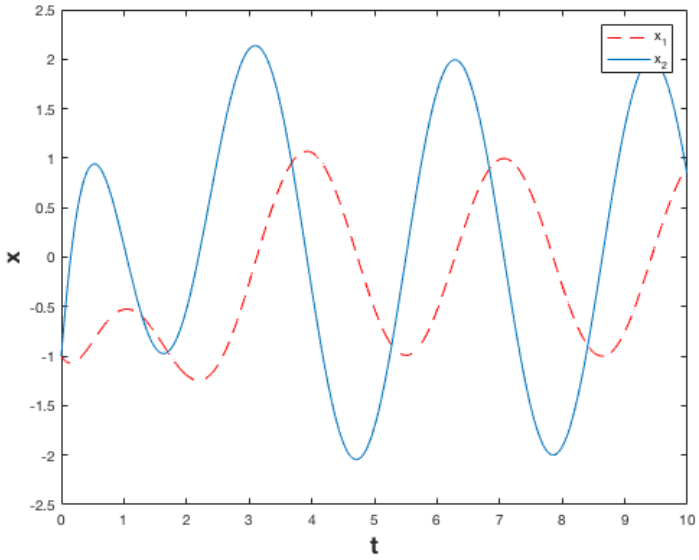

Figure 1.2: Plant's states vs. time

Figure 1.3 shows the reference signal and the response of the output of the system. It is clear that as time passes the output approaches the reference asymptotically which confirms the correctness of designed controller. Figure 1.4 also verifies this since it demonstrates that the error is converging to zero with time.

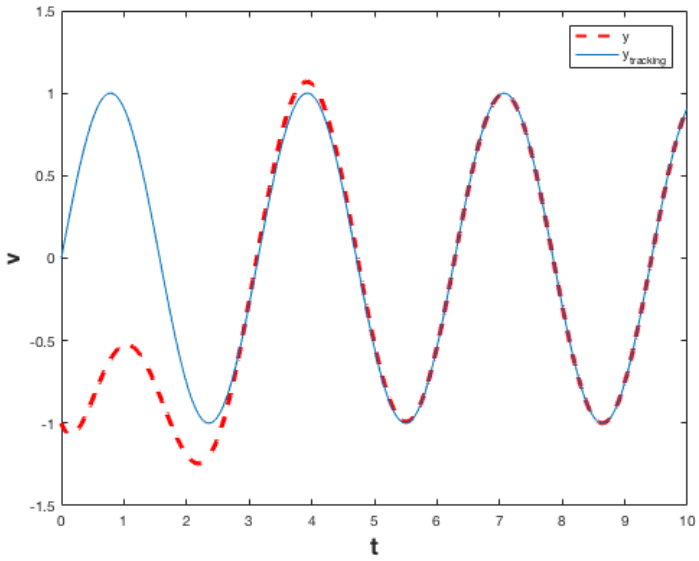

Figure 1.3: Output of the system and the reference trajectory

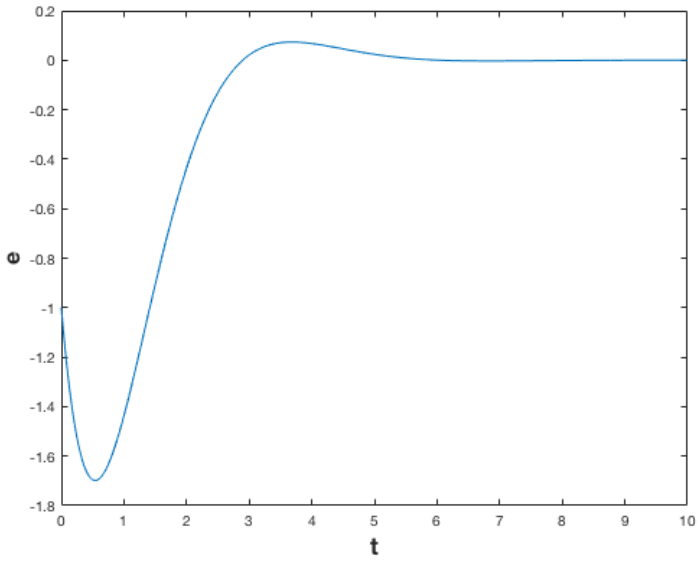

Figure 1.4: Error of the system

Extension of Exosystem

In the previous section, a regulator designed for a specific exosystem which was the combination of sine, cosine, and step functions. However, the exosystem can be further extended. For example, if one is interested in generating second order polynomial signals, the following

exosystem might be used:

$$\begin{aligned} \dot{v} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & -2 & 0 \end{bmatrix} v \\ D &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix} v \\ y_{tr} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix} v \end{aligned} \quad . \tag{1.39}$$

Then, if the initial condition of the exosystem is assumed to be:

$$v(0) = \begin{bmatrix} 2 & 0 & 0 & -1 & 0 \end{bmatrix} \tag{1.40}$$

the generated signals by exosystem will be as demonstrated below:

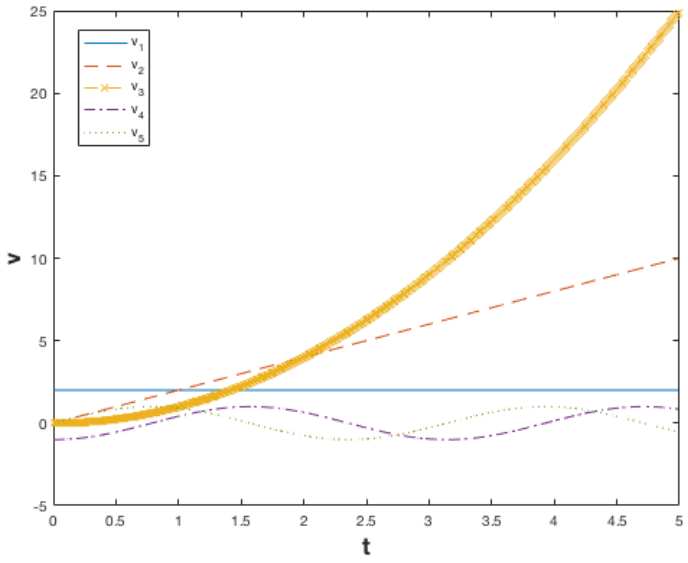

Figure 1.5: Extended exosystem signals

1.3 Backstepping Method

Backstepping method for control of distributed parameter systems has gained popularity in the past few years ([1, 20, 21, 31–33], etc.). The method uses an integral transformation, mostly Volterra transformation to convert the plant into a target system with desired stability properties, that is exponential stability. In other words, it tries to eliminate the unstability source in the plant by using the transformation. Then, the feedback controller law is derived and utilized in order to stabilize the system. The main feature of this method is its simplicity, which in some cases leads to a highly appealable explicit controller gain formula. This method also designs the controller at the boundary of the system in contrast to the other in-domain (or distributed) controller design methods. This is a more realistic

setting of actuation for majority of industrial processes which makes the backstepping so pleasing. The following section exhibits the idea of backstepping and how it is implemented on a generic parabolic system.

1.3.1 Developing Backstepping Controller for Parabolic Partial Differential Equations

Motivation

Parabolic PDEs are mostly encountered in transient phenomena when diffusion term has a dominant presence. For an instance consider a one dimensional transient heat transfer. The phenomenon under some assumption can be present as:

$$\rho c_p \frac{\partial T}{\partial z} = k \frac{\partial^2 T}{\partial z^2} - \rho c_p v \frac{\partial T}{\partial z} + Q \tag{1.41}$$

where ρ, c_p, k, Q, v, z, t, and T are density, specific heat, heat conductivity,heat source/sink, velocity, spacial coordinate, temporal coordinate, and temperature of the fluid, respectively. if heat source/sink term is assumed a linear function of temperature, one can represent it as αT. This equation might be presented in different way. In general parabolic PDE can be stated as

$$\bar{u}_t(x,t) = \epsilon(x)\bar{u}_{xx}(x,t) + b(x)\bar{u}_x(x,t) + \bar{\lambda}(x)\bar{u}(x,t) \tag{1.42}$$

$$\bar{u}_x(0,t) = \bar{q}\bar{u}(0,t) \tag{1.43}$$

$$\bar{u}(1,t) = U(t) \tag{1.44}$$

which is the main plant of our interest including its boundary conditions. In order to implement backstepping method, we have followed [32] in the next section.

Simplifying the plant

To eliminate the convective term -$\bar{u}_x(x,t)$- the following state transformation is applied

$$u(x,t) = e^{\int_0^x \frac{b(s)}{2\epsilon(s)} ds} \bar{u}(x,t) \tag{1.45}$$

For the sake of simplicity the *(x,t)* will be dropped from now on. To reach the new plant derivatives of old state in relation to new one will be calculated.
Derivative with respect to t

$$\bar{u}_t = e^{-\int_0^x \frac{b(s)}{2\epsilon(s)} ds} u_t \tag{1.46}$$

Derivative with respect to x

$$\bar{u} = e^{-\int_0^x \frac{b(s)}{2\epsilon(s)} ds} (u_x - \frac{b}{2\epsilon} u) \tag{1.47}$$

Second derivative with respect to x

$$\bar{u}_{xx} = -\frac{b}{2\epsilon} e^{-\int_0^x \frac{b(s)}{2\epsilon(s)} ds} (u_x - \frac{b}{2\epsilon} u) + e^{-\int_0^x \frac{b(s)}{2\epsilon(s)} ds} (u_{xx} - \frac{b}{2\epsilon} u_x - (\frac{b_x}{2\epsilon} - \frac{b\epsilon_x}{2\epsilon^2})) \tag{1.48}$$

Substituting the derivatives into equation 1.42 and 1.43 gives new simplified plant below

$$u_t = \epsilon u_{xx} + \lambda u \tag{1.49}$$

$$u_x(0,t) = qu(0,t) \tag{1.50}$$

in which

$$\lambda = \bar{\lambda} - \frac{b^2}{4\epsilon} - \frac{b_x}{2} + \frac{b\epsilon_x}{2\epsilon} \tag{1.51}$$

$$q = \bar{q} + \frac{b(0)}{2\epsilon(0)} \tag{1.52}$$

Transformation to target system

The main idea is to use state transformation

$$w(x,t) = u(x,t) - \int_0^x k(x,y)u(y,t)dy \tag{1.53}$$

to map 1.49 and 1.50 into stable target system

$$w_t(x,t) = \epsilon(x)w_{xx}(x,t) - cw(x,t) \tag{1.54}$$

$$w_x(0,t) = qw(0,t) \tag{1.55}$$

$$w(1,t) = 0. \tag{1.56}$$

The derivatives of *w(x,t)* ought to be calculated and replaced in the target system as follows: Derivative of w with respect to x

$$w_x = u_x - k(x,x)u - \int_0^x k_x(x,y)u(y,t)dy, \tag{1.57}$$

second derivative of w with respect to x

$$w_{xx} = u_{xx} - u\frac{dk(x,x)}{dx} - u_x k(x,x) - k_x(x,x)u - \int_0^x k_{xx}(x,y)u(y,t)dy, \tag{1.58}$$

and derivative with respect to t

$$\begin{aligned}
w_t &= u_t - \int_0^x k(x,y)u_t(y,t) \\
&= \epsilon u_{xx} + \lambda u - \int_0^x (k(x,y)(\epsilon(y)u_{yy}(y,t) + \lambda(y)u(y,t)))dy \\
&= \epsilon u_{xx} + \lambda u - k(x,y)\epsilon u_x(y,t)|_0^x + \int_0^x (k(x,y)\epsilon(y))_y u_y(y,t)dy \\
&\qquad - \int_0^x k(x,y)\lambda u(y,t)dy \\
&= \epsilon u_{xx} + \lambda u - k(x,x)\epsilon u_x + k(x,0)\epsilon(0)u_x(0) + (k(x,y)\epsilon(y))_y u(y,t)|_0^x \\
&\qquad - \int_0^x (k(x,y)\epsilon(y))_{yy} u(y,t)dy - \int_0^x k(x,y)\lambda u(y,t)dy \\
&= \epsilon u_{xx} + \lambda u - k(x,x)\epsilon u_x + k(x,0)\epsilon(0)u_x(0) \\
&\qquad + (\epsilon(x)k_y(x,x) + \epsilon_x(x)k(x,x))u - (\epsilon(0)k_y(x,0) + \epsilon_x(0)k(x,0))u(0) \\
&\qquad - \int_0^x (k(x,y)\epsilon(y))_{yy} u(y,t)dy - \int_0^x k(x,y)\lambda u(y,t)dy
\end{aligned} \tag{1.59}$$

Finding the Kernel

Now substituting derivatives into 1.54, the following will be obtained

$$\begin{aligned}(\epsilon-\epsilon)u_{xx}+(-k(x,x)\epsilon(x)+\epsilon(x)k(x,x))u_x\\ +(\lambda+\epsilon k_y(x,x)+\epsilon k_x(x,x)+\epsilon_x k(x,x)+c+\epsilon\frac{dk(x,x)}{dx})u\\ +\int_0^x(-(k(x,y)\epsilon(y))_{yy}-k\lambda(y)+k_{xx}(x,y)-ck(x,y))u(y,t)dy\\ +(k(x,0)\epsilon(0)u_x(0)-\epsilon(0)k_y(x,0)u(0)-\epsilon_x(0)k(x,0)u(0))=0 \quad (1.60)\end{aligned}$$

which implies that in order to the target equation 1.54 be valid, all the coefficients must be equal to zero, therefore, gain kernel is found as follows;

$$-(k(x,y)\epsilon(y))_{yy}+k_{xx}(x,y)\epsilon(x)=(\lambda(y)+c)k(x,y) \quad (1.61)$$

$$k_y(x,0)=(q\frac{\epsilon_x(0)}{\epsilon(0)})k(x,0) \quad (1.62)$$

$$\frac{d}{dx}k(x,x)+\frac{\epsilon_x(x)}{2\epsilon(x)}k(x,x)=-\frac{(\lambda(x)+c)}{2\epsilon(x)} \quad (1.63)$$

$$k(0,0)=0 \quad (1.64)$$

Solving the ODE (1.63) and (1.64)

Here we seek to calculate the ODE (1.63) with its boundary condition (1.64) as demonstrated below.
Lets define $p=\frac{\epsilon_x(x)}{2\epsilon(x)}$ and $q=-\frac{(\lambda(x)+c)}{2\epsilon(x)}$. Then the solution of the ODE is

$$\begin{aligned}k(x,x)&=e^{-\int pdx}(\int e^{\int pdx}qdx+c_1)\\ &=\frac{1}{\sqrt{\epsilon}}(\int_0^x\sqrt{\epsilon}\frac{-(\lambda(\tau)+c)}{2\epsilon}dx+c_1)\end{aligned}$$

as $k(0,0)=0$, then $c_1=0$ So the (1.63) becomes

$$k(x,x)=\frac{-1}{2\sqrt{\epsilon}}(\int_0^x\frac{(\lambda(\tau)+c)}{\sqrt{\epsilon}}dx \quad (1.65)$$

Simplifying the Kernel

Defining

$$\begin{aligned}\breve{k}(\bar{x},\bar{y})&=\epsilon(y)k(x,y)\\ \bar{x}=\varphi(x)&\qquad \bar{y}=\varphi(y)\\ \varphi(x)&=\sqrt{\epsilon(0)}\int_0^x\frac{d\tau}{\epsilon(\tau)}\end{aligned}$$

and taking derivatives of *k(x,y)* in terms of $\breve{k}(x,y)$ we get to:

$$k_x(x,y) = \frac{1}{\epsilon(y)}\breve{k}_x(\bar{x},\bar{y}) \tag{1.66}$$

$$k_{xx}(x,y) = \frac{1}{\epsilon(y)}\breve{k}_{xx}(\bar{x},\bar{y}) \tag{1.67}$$

$$k_y(x,y) = \frac{1}{\epsilon(y)}\breve{k}_y(\bar{x},\bar{y}) - \frac{\epsilon_y(y)}{\epsilon^2(y)}\breve{k}(\bar{x},\bar{y}) \tag{1.68}$$

where using chain rule we find the derivatives with respect to $\bar{x}$ and $\bar{y}$ as follows:

$$\frac{d\breve{k}}{dx} = \frac{\partial\breve{k}}{\partial\bar{x}}\frac{d\bar{x}}{dx} \tag{1.69}$$

$$\frac{d\breve{k}}{dy} = \frac{\partial\breve{k}}{\partial\bar{y}}\frac{d\bar{y}}{dy} \tag{1.70}$$

$$\frac{d}{dx}\frac{d\breve{k}}{dx} = \frac{\partial\breve{k}_x}{\partial\bar{x}}\frac{d\bar{x}}{dx} + \frac{\partial\breve{k}_x}{\partial x} \tag{1.71}$$

$$\frac{d}{dy}\frac{d\breve{k}}{dy} = \frac{\partial\breve{k}_y}{\partial\bar{y}}\frac{d\bar{y}}{dy} + \frac{\partial\breve{k}_y}{\partial y} \tag{1.72}$$

Therefore,

$$\breve{k}_x = \breve{k}_{\bar{x}}(\frac{\epsilon(0)}{\epsilon(x)})^{1/2} \tag{1.73}$$

$$\breve{k}_y = \breve{k}_{\bar{y}}(\frac{\epsilon(0)}{\epsilon(y)})^{1/2} \tag{1.74}$$

$$\breve{k}_{xx} = \breve{k}_{\bar{x}\bar{x}}(\frac{\epsilon(0)}{\epsilon(x)}) + \breve{k}_{\bar{x}}(-1/2)(\frac{\epsilon(0)}{\epsilon(x)})^{1/2}(\frac{\epsilon_x(x)}{\epsilon(x)}) \tag{1.75}$$

$$\breve{k}_{yy} = \breve{k}_{\bar{y}\bar{y}}(\frac{\epsilon(0)}{\epsilon(y)}) + \breve{k}_{\bar{y}}(-1/2)(\frac{\epsilon(0)}{\epsilon(y)})^{1/2}(\frac{\epsilon_y(y)}{\epsilon(y)}) \tag{1.76}$$

Here the two aforementioned transformations and their derivatives will be substituted in the Eq. 1.61

$$\epsilon(x)(\frac{1}{\epsilon(y)}\breve{k}_{xx}(\bar{x},\bar{y})) - \breve{k}_{yy}(\bar{x},\bar{y}) = (\lambda(y)+c)\frac{\breve{k}(\bar{x},\bar{y})}{\epsilon(y)}$$

$$(\frac{\epsilon(0)}{\epsilon(x)}\breve{k}_{\bar{x}\bar{x}} - (1/2)(\frac{\epsilon(0)}{\epsilon(x)})^{1/2}(\frac{\epsilon_x(x)}{\epsilon(x)})\breve{k}_{\bar{x}\bar{x}})$$
$$- (\frac{\epsilon(0)}{\epsilon(y)}\breve{k}_{\bar{y}\bar{y}} - (1/2)(\frac{\epsilon(0)}{\epsilon(y)})^{1/2}(\frac{\epsilon_y(y)}{\epsilon(y)})\breve{k}_{\bar{y}\bar{y}}) = \frac{\lambda(y)+c}{\epsilon(y)}\breve{k}$$

Finally, we end up with

$$\breve{k}_{\bar{x}\bar{x}} - \breve{k}_{\bar{y}\bar{y}} = \frac{\epsilon_x(x)}{2\sqrt{\epsilon(0)\epsilon(x)}}\breve{k}_{\bar{x}} - \frac{\epsilon_y(y)}{2\sqrt{\epsilon(0)\epsilon(y)}}\breve{k}_{\bar{y}} + \frac{\lambda(y)+c}{\epsilon(0)}\breve{k} \tag{1.77}$$

Applying the transformation to boundary conditions of the kernel (equations 1.62 and 1.65)

$$\begin{gathered}\frac{1}{\epsilon(0)}\breve{k}_{\bar{y}}(\bar{x},0)-\frac{\epsilon_y(0)}{\epsilon^2(0)}\breve{k}(\bar{x},0)=(q-\frac{\epsilon_y(0)}{\epsilon(0)})\frac{\breve{k}(x,0)}{\epsilon(0)}\\ \breve{k}_y(\bar{x},0)=q\breve{k}(\bar{x},0)\end{gathered}\tag{1.78}$$

$$\frac{\breve{k}(\bar{x},\bar{x})}{\epsilon(x)}=\frac{-1}{2\sqrt{\epsilon(x)}}\int_0^x\frac{\lambda(\tau)+c}{\sqrt{\epsilon(\tau)}}d\tau$$

Let's define $\xi=\varphi(\tau)=\bar{x}$, then

$$\begin{aligned}d\xi&=\frac{d\varphi(\tau)}{d\tau}=\frac{\sqrt{\epsilon(0)}}{\sqrt{\epsilon(\tau)}}d\tau\\ d\tau&=\frac{\sqrt{\epsilon(\tau)}}{\sqrt{\epsilon(0)}}d\xi\end{aligned}$$

So,

$$\breve{k}(\bar{x},\bar{x})=-\frac{\sqrt{\epsilon(x)}}{2\sqrt{\epsilon(0)}}\int_0^{\bar{x}}(\lambda(\varphi^{-1}(\xi))+c)d\xi\tag{1.79}$$

Further Simplifying the Kernel

The transformation is defined as follow

$$\bar{k}(\bar{x},\bar{y})=(\epsilon(x)\epsilon(y))^{-1/4}\breve{k}(\bar{x},\bar{y})\tag{1.80}$$

taking the derivatives,

$$\begin{gathered}\frac{\sqrt{\epsilon(0)}}{\sqrt{\epsilon(x)}}\breve{k}_{\bar{x}}=(\epsilon(x)\epsilon(y))^{1/4}(\frac{\epsilon_x(x)}{4\epsilon(x)})\bar{k}+(\epsilon(x)\epsilon(y))^{1/4}\bar{k}_{\bar{x}}\frac{\sqrt{\epsilon(0)}}{\sqrt{\epsilon(x)}}\\ \breve{k}_{\bar{x}}=(\epsilon(x)\epsilon(y))^{1/4}(\bar{k}_{\bar{x}}+\frac{\epsilon_x(x)}{4\sqrt{\epsilon(x)\epsilon(0)}}\bar{k})\end{gathered}\tag{1.81}$$

$$\begin{aligned}\frac{\sqrt{\epsilon(0)}}{\sqrt{\epsilon(x)}}\breve{k}_{\bar{x}\bar{x}}=(\epsilon(x)\epsilon(y))^{1/4}&\frac{\epsilon_x(x)}{4\epsilon(x)}(\bar{k}_{\bar{x}}+\frac{\epsilon_x(x)}{4\sqrt{\epsilon(x)\epsilon(0)}}\bar{k})\\ &+(\epsilon(x)\epsilon(y))^{1/4}(\frac{\sqrt{\epsilon(0)}}{\sqrt{\epsilon(x)}}\bar{k}_{\bar{x}\bar{x}}+\frac{\sqrt{\epsilon(0)}}{\sqrt{\epsilon(x)}}\frac{\epsilon_x(x)}{4\sqrt{\epsilon(x)\epsilon(0)}}\bar{k}_{\bar{x}}\\ &\qquad+(\frac{\epsilon_{xx}(x)}{4\sqrt{\epsilon(x)\epsilon(0)}}-\frac{\epsilon_x^2(x)/\epsilon(x)}{8\sqrt{\epsilon(x)\epsilon(0)}})\bar{k})\end{aligned}$$

or in summary

$$\breve{k}_{\bar{x}\bar{x}}=(\epsilon(x)\epsilon(y))^{1/4}(\bar{k}_{\bar{x}\bar{x}}+\frac{\epsilon_x(x)}{2\sqrt{\epsilon(x)\epsilon(0)}}\bar{k}_{\bar{x}}+(\frac{\epsilon_{xx}(x)}{4\epsilon(0)}-\frac{\epsilon_x^2(x)}{16\epsilon(x)\epsilon(0)})\bar{k})\tag{1.82}$$

$$\breve{k}_{\bar{y}} = (\epsilon(x)\epsilon(y))^{1/4}(\bar{k}_{\bar{y}} + \frac{\epsilon_y(y)}{4\sqrt{\epsilon(y)\epsilon(0)}}\bar{k}) \tag{1.83}$$

$$\breve{k}_{\bar{y}\bar{y}} = (\epsilon(x)\epsilon(y))^{1/4}(\bar{k}_{\bar{y}\bar{y}} + \frac{\epsilon_y(y)}{2\sqrt{\epsilon(y)\epsilon(0)}}\bar{k}_{\bar{y}} + (\frac{\epsilon_{yy}(y)}{4\epsilon(0)} - \frac{\epsilon_y^2(y)}{16\epsilon(y)\epsilon(0)})\bar{k}) \tag{1.84}$$

Substituting (1.82) through (1.84) into (1.77) to (1.79), converts the (1.77) to

$$\begin{aligned}\bar{k}_{\bar{x}\bar{x}} + \frac{\epsilon_x(x)}{2\sqrt{\epsilon(x)\epsilon(0)}}\bar{k}_{\bar{x}} + (\frac{\epsilon_{xx}(x)}{4\epsilon(0)} - \frac{\epsilon_x^2(x)}{16\epsilon(x)\epsilon(0)})\bar{k} \\ - (\bar{k}_{\bar{y}\bar{y}} + \frac{\epsilon_y(y)}{2\sqrt{\epsilon(y)\epsilon(0)}}\bar{k}_{\bar{y}} + (\frac{\epsilon_{yy}(y)}{4\epsilon(0)} - \frac{\epsilon_y^2(y)}{16\epsilon(y)\epsilon(0)})\bar{k}) = \\ \frac{\epsilon_x(x)}{2\sqrt{\epsilon(x)\epsilon(0)}}(\bar{k}_{\bar{x}} + \frac{\epsilon_x(x)}{4\sqrt{\epsilon(x)\epsilon(0)}}\bar{k}) \\ - \frac{\epsilon_y(y)}{2\sqrt{\epsilon(y)\epsilon(0)}}(\bar{k}_{\bar{y}} + \frac{\epsilon_y(y)}{4\sqrt{\epsilon(y)\epsilon(0)}}\bar{k}) + \frac{\lambda(y)+c}{\epsilon(0)}\bar{k}\end{aligned}$$

or

$$\epsilon(0)(\bar{k}_{\bar{x}\bar{x}}(\bar{x},\bar{y}) - \bar{k}_{\bar{y}\bar{y}}(\bar{x},\bar{y})) = \bar{\lambda}(\bar{x},\bar{y})\bar{k}(\bar{x},\bar{y}) \tag{1.85}$$

Where

$$\bar{\lambda}(\bar{x},\bar{y}) = \frac{3}{16}(\frac{\epsilon_x^2(x)}{\epsilon(x)} - \frac{\epsilon_y^2(y)}{\epsilon(y)}) + \frac{1}{4}(\epsilon_{yy}(y) - \epsilon_{xx}(x)) + \lambda(y) + c \tag{1.86}$$

and the equation (1.78) to

$$(\epsilon(x)\epsilon(0))^{1/4}(\bar{k}_{\bar{y}}(\bar{x},0) + \frac{\epsilon_y(0)}{4\epsilon(0)}\bar{k}(\bar{x},0)) = q(\epsilon(x)\epsilon(0))^{1/4}\bar{k}(\bar{x},0)$$

$$\bar{k}_{\bar{y}}(\bar{x},0) = (q - \frac{\epsilon_y(0)}{4\epsilon(0)})\bar{k}(\bar{x},0). \tag{1.87}$$

At last equation (1.79) turns to

$$\bar{k}(\bar{x},\bar{x}) = -\frac{1}{2\sqrt{\epsilon(0)}}\int_0^{\bar{x}}(\lambda(\varphi^{-1}(\xi)) + c)d\xi. \tag{1.88}$$

In summary,

$$\begin{aligned}\epsilon(0)(\bar{k}_{\bar{x}\bar{x}}(\bar{x},\bar{y}) - \bar{k}_{\bar{y}\bar{y}}(\bar{x},\bar{y})) = \\ (\frac{3}{16}(\frac{\epsilon_x^2(x)}{\epsilon(x)} - \frac{\epsilon_y^2(y)}{\epsilon(y)}) + \frac{1}{4}(\epsilon_{yy}(y) - \epsilon_{xx}(x)) + \lambda(y) + c)\bar{k}(\bar{x},\bar{y}) \\ \bar{k}_{\bar{y}}(\bar{x},0) = (q - \frac{\epsilon_y(0)}{4\epsilon(0)})\bar{k}(\bar{x},0) \\ \bar{k}(\bar{x},\bar{x}) = -\frac{1}{2\sqrt{\epsilon(0)}}\int_0^{\bar{x}}(\lambda(\varphi^{-1}(\xi)) + c)d\xi\end{aligned}$$

The kernel system in the box have no closed form solution and has to be solved using numerical methods.

Examples

In this part controller for two different sample plants will be derived by the kernel obtained beforehand and the system will be stabilized. Furthermore, the plots of the open loop and closed loop system as well as the plot of the kernel will be depicted.

Plant with Constant λ

The plant to be controlled here is

$$u_t(x,t) = \epsilon(x)u_{xx}(x,t) + \lambda u(x,t) \tag{1.89}$$

$$u(0,t) = 0 \tag{1.90}$$

Gain Kernel: Following the expressions derived in the last section, the kernel for the PDE model given by Eq. 1.89 is

$$\epsilon(0)(\bar{k}_{\bar{x}\bar{x}}(\bar{x},\bar{y}) - \bar{k}_{\bar{y}\bar{y}}(\bar{x},\bar{y})) = \bar{\lambda}(\bar{x},\bar{y})\bar{k}(\bar{x},\bar{y}) \tag{1.91}$$

$$\bar{k}(\bar{x},\bar{x}) = -\frac{\lambda + c}{2\sqrt{\epsilon(0)}}\bar{x} \tag{1.92}$$

$$\bar{k}(\bar{x},0) = 0 \tag{1.93}$$

where λ is

$$\bar{\lambda} = \frac{3}{16}(\frac{\epsilon_x^2(x)}{\epsilon(x)} - \frac{\epsilon_y^2(y)}{\epsilon(x)}) + \frac{1}{4}(\epsilon_{yy}(y) - \epsilon_{xx}(x)) + \lambda + c \tag{1.94}$$

To find the closed form solution it is assumed that

$$\frac{3}{16}\frac{\epsilon_x^2(x)}{\epsilon(x)} - \frac{1}{4}\epsilon_{xx}(x) = C \tag{1.95}$$

where C is an arbitrary constant. This way $\bar{\lambda}$ will become

$$\bar{\lambda} = \lambda + c \tag{1.96}$$

Transforming the Kernel into Integral Equation is highly desired as it make the solution much easier to be found. First, the below variable transformations are introduced

$$\begin{aligned} \bar{k}(\bar{x},\bar{y}) &= G(\xi,\eta) \\ \xi &= \bar{x} + \bar{y} \\ \eta &= \bar{x} - \bar{y} \end{aligned} \tag{1.97}$$

Then, the derivatives of the kernel with respect to the new variable has been taken

$$\bar{k}_{\bar{x}} = \frac{\partial G}{\partial \xi}\frac{d\xi}{d\bar{x}} + \frac{\partial G}{\partial \eta}\frac{d\eta}{d\bar{y}} = G_\xi + G_\eta \tag{1.98}$$

$$\bar{k}_{\bar{x}\bar{x}} = G_{\xi\xi} + 2G_{\xi\eta} + G_{\eta\eta} \tag{1.99}$$

$$\bar{k}_{\bar{x}} = \frac{\partial G}{\partial \xi}\frac{d\xi}{d\bar{y}} + \frac{\partial G}{\partial \eta}\frac{d\eta}{d\bar{y}} = G_\xi - G_\eta \tag{1.100}$$

$$\bar{k}_{\bar{y}\bar{y}} = G_{\xi\xi} - 2G_{\xi\eta} + G_{\eta\eta} \tag{1.101}$$

Now by substituting these derivatives into equation (1.91), we come up with

$$\epsilon(0)(4G_{\xi\eta}) = (\lambda + c)G$$

or

$$G_{\xi\eta} = \frac{\lambda + c}{4}G \tag{1.102}$$

At $\bar{y} = \bar{x}$, $\xi = 2\bar{x}$ and $\eta = 0$; Therefor equation (1.92) becomes

$$G(\xi, 0) = -\frac{\lambda + c}{4\sqrt{\epsilon(0)}}\xi \tag{1.103}$$

Also at $\bar{y} = 0$, $\xi = \eta = \bar{x}$, Hence equation (1.93) is transformed to

$$G(\xi, \xi) = 0 \tag{1.104}$$

Conversion of the partial differential equation system (1.102) to (1.104) to a single integral equation is achieved via integrating equation (1.102) and replacing equations (1.103) and (1.104) into it as demonstrated below.

By inegrating equation 1.102 with respect to η from 0 to η

$$G_\xi = \int_0^\eta \frac{\lambda + c}{4\epsilon(0)}G(\tau)d\tau + G_\xi(\xi, 0) \tag{1.105}$$

and integrating with respect to ξ from η to ξ

$$G = \int_\eta^\xi \int_0^\eta \frac{\lambda + c}{4\epsilon(0)}G(s, \tau)d\tau ds + \int_0^\eta G_\xi(s, 0)ds + G(\eta, \eta) \tag{1.106}$$

and using Eq. (**??** and Eq. (1.104

$$)G = -\frac{\lambda + c}{4\sqrt{\epsilon(0)}}(\xi - \eta) + \frac{\lambda + c}{4\epsilon(0)}\int_\eta^\xi \int_0^\eta G(s, \tau)d\tau ds \tag{1.107}$$

Equation (1.107) can be given as

$$G = \tilde{\lambda}_1(\xi - \eta) + \tilde{\lambda}_2 \int_\eta^\xi \int_0^\eta G(s, \tau)d\tau ds \tag{1.108}$$

where

$$\tilde{\lambda}_1 = -\frac{\lambda + c}{4\sqrt{\epsilon(0)}} \tag{1.109}$$

$$\tilde{\lambda}_2 = \frac{\lambda + c}{4\epsilon(0)} \tag{1.110}$$

Solution of Integral Equation by Method of Successive Approximation: In this part the integral Eq. (1.108) will be solved recursively as follows:

$$G^{n+1} = \tilde{\lambda}_1(\xi - \eta) + \tilde{\lambda}_2 \int_\eta^\xi \int_0^\eta G^n(s,\tau) d\tau ds \tag{1.111}$$

Guessing, $G_0 = 0$

$$G^1 = \tilde{\lambda}_1(\xi - \eta) \tag{1.112}$$

Now defining

$$\Delta G^n = G^{n+1} - G^n \tag{1.113}$$

$$\Delta G^n = \tilde{\lambda}_2 \int_\eta^\xi \int_0^\eta \Delta G^{n-1}(s,\tau) d\tau ds \tag{1.114}$$

Now progressing this way

$$\begin{aligned} \Delta G^1 &= \tilde{\lambda}_2 \int_\eta^\xi \int_0^\eta \Delta G^0(s,\tau) d\tau ds \\ &= \tilde{\lambda}_2 \int_\eta^\xi \int_0^\eta \tilde{\lambda}_1(\tau - s) ds d\tau \\ &= \frac{\tilde{\lambda}_1 \tilde{\lambda}_2}{2} \xi\eta(\xi - \eta) \end{aligned} \tag{1.115}$$

$$\begin{aligned} \Delta G^2 &= \tilde{\lambda}_2 \int_\eta^\xi \int_0^\eta \Delta G^1(s,\tau) d\tau ds \\ &= \tilde{\lambda}_2 \int_\eta^\xi \int_0^\eta \tilde{\lambda}_1 \tilde{\lambda}_2 (\tau^2 s/2 - \tau s^2/2) ds d\tau \\ &= \frac{\tilde{\lambda}_1 \tilde{\lambda}_2^2}{2 \times 6} \xi^2 \eta^2 (\xi - \eta) \end{aligned} \tag{1.116}$$

$$\begin{aligned} \Delta G^3 &= \tilde{\lambda}_2 \int_\eta^\xi \int_0^\eta \Delta G^2(s,\tau) d\tau ds \\ &= \tilde{\lambda}_2 \int_\eta^\xi \int_0^\eta \frac{\tilde{\lambda}_1 \tilde{\lambda}_2^2}{2 \times 6} (\tau^3 s^2/2 - \tau^2 s^3/2) ds d\tau \\ &= \frac{\tilde{\lambda}_1 \tilde{\lambda}_2^3}{24 \times 6} \xi^3 \eta^3 (\xi - \eta) \end{aligned} \tag{1.117}$$

Continuing this way, we end up with the general form of ΔG

$$\Delta G^n = \xi^n \eta^n (\xi - \eta) \frac{\tilde{\lambda}_1 \tilde{\lambda}_2^n}{n!(n+1)!} \tag{1.118}$$

We know that

$$G(\xi,\eta) = \lim_{n\to\infty} G^n(\xi,\eta) \tag{1.119}$$

or

$$G(\xi,\eta)=\sum_{n=0}^{\infty}\Delta G^n(\xi,\eta) \tag{1.120}$$

Recalling that

$$I_1(x)=\sum_{n=0}^{\infty}\frac{(x/2)^{2n+1}}{n!(n+1)!} \tag{1.121}$$

where I is modified Bessel function of the first kind, G can be stated as

$$\begin{aligned} G(\xi,\eta)&=\frac{\tilde{\lambda}_1(\xi-\eta)}{\sqrt{\tilde{\lambda}_2\xi\eta}}\sum_{n=0}^{\infty}\frac{(\sqrt{\tilde{\lambda}_2\xi\eta})^{2n+1}}{n!(n+1)!}\\ &=\frac{\tilde{\lambda}_1(\xi-\eta)}{\sqrt{\tilde{\lambda}_2\xi\eta}}I_1(2\sqrt{\tilde{\lambda}_2\xi\eta}) \end{aligned} \tag{1.122}$$

Transforming G back to K we end up with

$$\bar{k}(\bar{x},\bar{y})=\frac{\tilde{\lambda}_1\bar{y}}{2\sqrt{\tilde{\lambda}_2(\bar{x}^2-\bar{y}^2)}}I_1(2\sqrt{\tilde{\lambda}_2(\bar{x}^2-\bar{y}^2)}) \tag{1.123}$$

or

$$\bar{k}(\bar{x},\bar{y})=-\frac{\lambda+c}{\sqrt{\epsilon(0)}}\bar{y}\frac{I_1(\sqrt{\frac{\lambda+c}{\epsilon(0)}(\bar{x}^2-\bar{y}^2)})}{\sqrt{\frac{\lambda+c}{\epsilon(0)}(\bar{x}^2-\bar{y}^2)}} \tag{1.124}$$

Finally transforming $\bar{k}(\bar{x},\bar{y})$ back to $k(x,y)$ leads to

$$\bar{k}(\bar{x},\bar{y})=(\epsilon(x)\epsilon(y))^{-1/4}\breve{k}(\bar{x},\bar{y})=(\epsilon(x)\epsilon(y))^{-1/4}\epsilon(y)k(x,y) \tag{1.125}$$

$$k(x,y)=-\frac{\epsilon(x)^{1/4}}{\epsilon(y)^{3/4}}\frac{\lambda+c}{\sqrt{\epsilon(0)}}\bar{y}\frac{I_1(\sqrt{\frac{\lambda+c}{\epsilon(0)}(\bar{x}^2-\bar{y}^2)})}{\sqrt{\frac{\lambda+c}{\epsilon(0)}(\bar{x}^2-\bar{y}^2)}} \tag{1.126}$$

$$\bar{x}=\varphi(x)\quad \bar{y}=\varphi(y) \tag{1.127}$$

$$\varphi(\xi)=\frac{1+\theta_0x_0^2}{\sqrt{\theta_0}}(arctan(\sqrt{\theta_0}(\xi-x_0))+arctan(\sqrt{\theta_0}x_0)) \tag{1.128}$$

Results The open loop system without imposing control on its boundary is shown in the figure 1.6. As shown, the state propagates with the time and is unstable obviously.

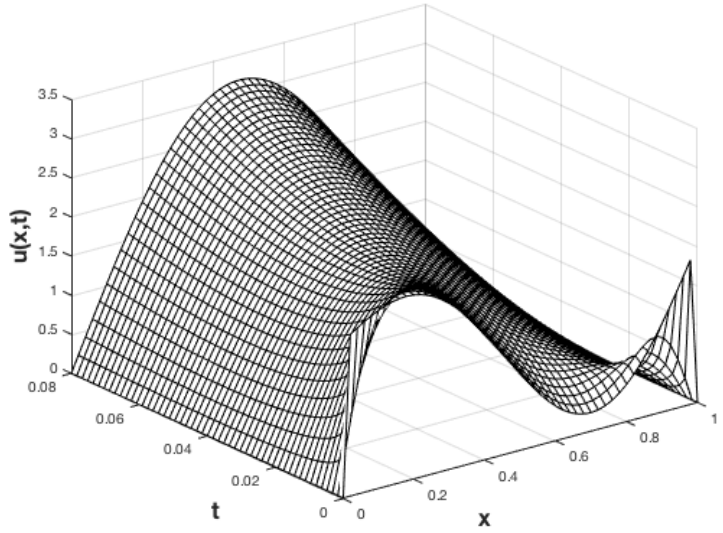

Figure 1.6: Open loop system behavior of the Parabolic plant with constant λ

On the other hand, Applying the control action on the boundary of equation 1.89 with the found kernel along with the equation 3.4 lead to the following figures which depicts the behavior of closed loop system, and demonstrate the ability of backstepping method to successfully stabilize the system. Figure 1.7 shows the state evolution of the system and how it converges to zero as the time goes by which means the system is successfully stabilized.

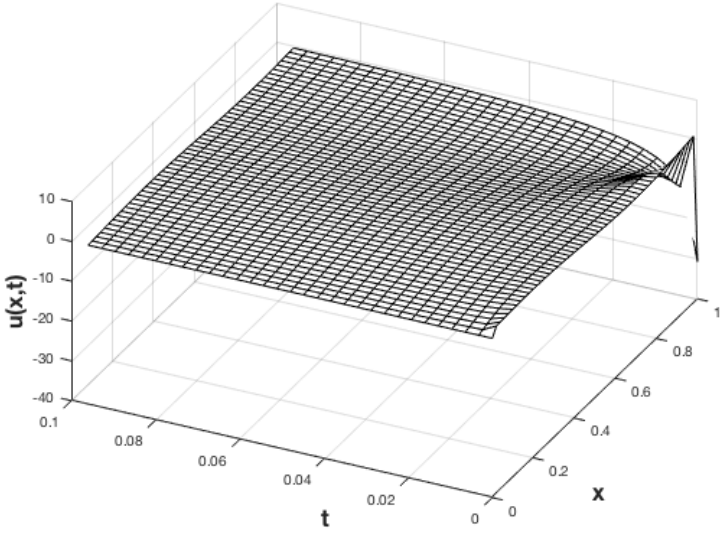

Figure 1.7: Closed loop system behavior for parabolic plant with constant λ

Furthermore, figure 1.8 shows the kernel of the system. As one can see, the absolute value of the kernel decreases near the left boundary (x=0) which is reasonable as the state of the system is fixed there. But the gain is higher near x=1 where the actuation is exerted and the controller has more authority.

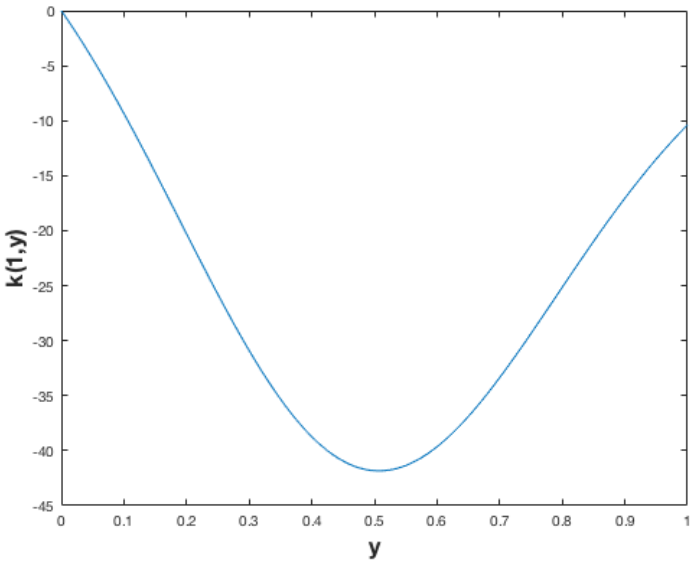

Figure 1.8: Kernel of the controller developed for parabolic system with constant λ

Plant with Non-constant Thermal Conductivity

Here we will examine the case when the plant has a non constant thermal conductivity term. the plant is state below:

$$u_t(x,t) = \frac{d}{dx}(\epsilon(x)\frac{d}{dx}(u(x,t)) + \lambda u(x,t) \tag{1.129}$$

$$u(0,t) = 0 \tag{1.130}$$

The advection term in the PDE can be eliminated by the transformation below

$$v = ue^{\int_0^x \frac{b(s)}{2\epsilon(s)}ds} = \sqrt{\epsilon(x)}u \tag{1.131}$$

which transform the Eq. (1.129) and (1.130) to

$$v_t = \epsilon v_{xx} + \bar{\lambda} v \tag{1.132}$$

$$v(0,t) = 0 \tag{1.133}$$

where

$$\bar{\lambda} = \lambda - \frac{b^2}{4\epsilon} - \frac{b_x}{2} + \frac{b\epsilon_x}{2\epsilon} = \lambda - \frac{\epsilon_{xx}}{2} + \frac{\epsilon_x^2}{4\epsilon} \tag{1.134}$$

Then, everything is almost similar to last section. To have a closed form solution, it is assumed that $\bar{\lambda}$ is equal to λ. Hence an $\epsilon(x)$ which satisfy this assumption and Eq. 1.94 as well is found to be

$$\epsilon(x) = \epsilon_0(x - x_0)^2 \tag{1.135}$$

and the kernel is

$$\bar{\bar{k}}(x,y) = -\frac{\epsilon(x)^{1/4}}{\epsilon(y)^{3/4}} \frac{\lambda + c}{\sqrt{\epsilon(0)}} \bar{y} \frac{I_1(\sqrt{\frac{\lambda+c}{\epsilon(0)}(\bar{x}^2 - \bar{y}^2)})}{\sqrt{\frac{\lambda+c}{\epsilon(0)}(\bar{x}^2 - \bar{y}^2)}} \tag{1.136}$$

This is the kernel calculated for (1.132). To find the kernel for the original system (1.129) the transformation below is applied;

$$k = \bar{\bar{k}} e^{\int_y^x \frac{b(s)}{2\epsilon(s)} ds} \tag{1.137}$$

which yields:

$$k(x,y) = -\frac{\epsilon(x)^{3/4}}{\epsilon(y)^{5/4}} \frac{\lambda + c}{\sqrt{\epsilon(0)}} \bar{y} \frac{I_1(\sqrt{\frac{\lambda+c}{\epsilon(0)}(\bar{x}^2 - \bar{y}^2)})}{\sqrt{\frac{\lambda+c}{\epsilon(0)}(\bar{x}^2 - \bar{y}^2)}} \tag{1.138}$$

$$\bar{x} = \varphi(x) \quad \bar{y} = \varphi(y) \tag{1.139}$$

$$\varphi(\xi) = x_0 ln(1 - \frac{\xi}{x_0}) \tag{1.140}$$

This kernel along with the Eq. (3.4) yields the closed loop system solution. Figure 1.9 shows the open loop system. it is obviously unstable as it grows with time.

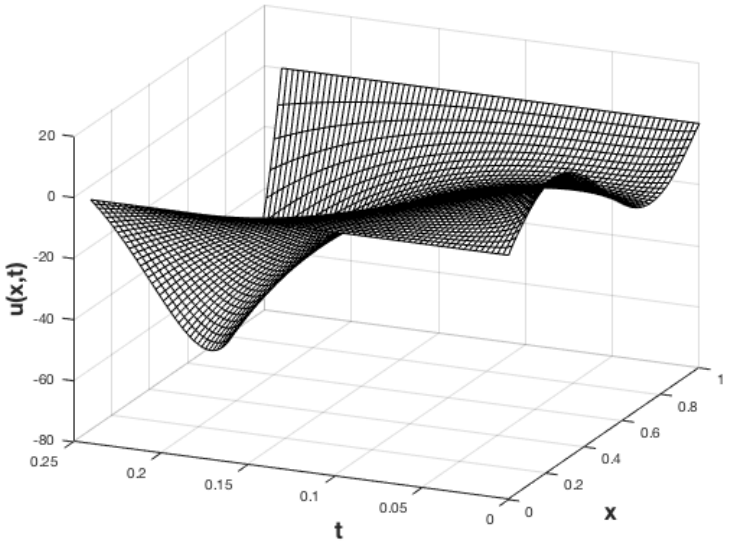

Figure 1.9: Open loop system behavior of parabolic plant with non constant thermal Conductivity

Nevertheless, when the controller imposed, the system will become stable as demonstrated in figure 1.10. This shows again, the successful performance of the controller.

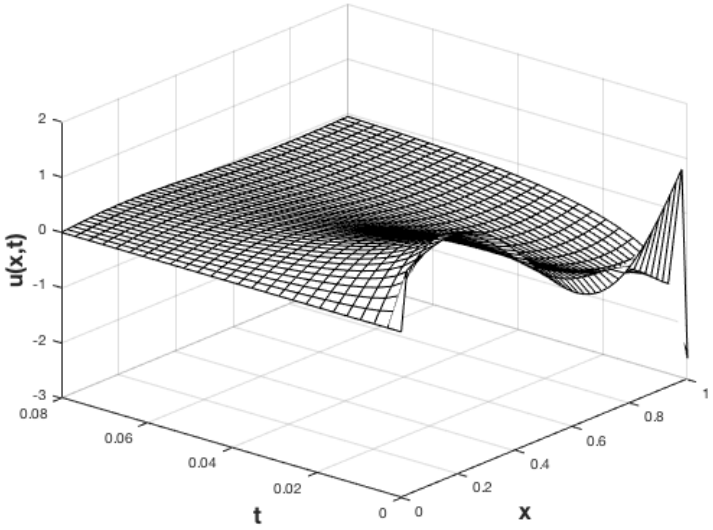

Figure 1.10: Closed loop system behavior of parabolic plant with non constant thermal Conductivity

The Kernel also has the similar trend as previous example which is depicted in figure 1.11.

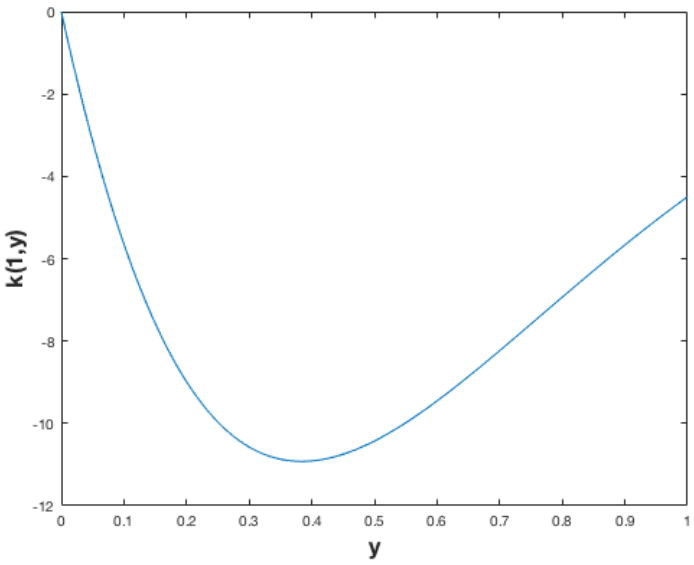

Figure 1.11: Kernel of closed loop system of parabolic plant with non constant thermal conductivity

1.4 Contribution of This Work

The contribution of this work could be summarized as:

- Output regulation method with internal model control principle is investigated by implementation of the method on first order partial differential equations with in-domain or boundary actuation settings. System is stabilized then, trajectories are tracked, and disturbances are rejected.
- Backstepping method is assessed through implementation on a generic first order hyperbolic partial differential equation.
- Backstepping method is implemented on a first order hyperbolic partial differential equation which represents a tubular reactor with recycle where the boundaries are coupled at the inlet and outlet of the system. The system is stabilized at the origin and afterwards a trajectory is tracked. The effect of different recycle ratios are also investigated on the response of the system.

Chapter 2

Output Regulation of Infinite Dimensional Systems by Internal Model Principle

2.1 Introduction

This chapter deals with the extension of output regulation with internal model control to hyperbolic partial differential equations. Two approaches are considered here. The first one is in-domain control where the controller is distributed through the domain. Mathematically speaking the control law is a function of space. Temperature control using a shell or coating around a one dimensional system might be considered as such a system. The second approach is design of boundary controller where tha actuator applies its action on one boundary of the system. This type of actuation usually deemed to be more practical and includes most of controller designs in practice. These two analysis are demonstrated in the following sections.

2.2 In-Domain Controller

2.2.1 Problem Formulation

Consider the following tubular reactor:

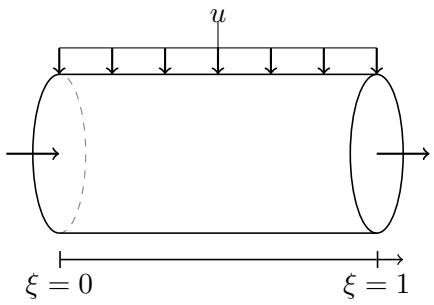

Figure 2.1: Tubular reactor with distributed acuation

Here, the temperature inside the reactor is desired to be regulated by distributed actuation. For a tubular reactor, coating around the shell can be considered as distributed actuation where it is represented as u in the figure 2.2.1. If the advection is much more dominant in the process, one is able to neglect the diffusion term. If this holds, the system can be modelled by a first order hyperbolic partial differential equation:

$$\rho c_p \frac{\partial T}{\partial z} = -\rho c_p v \frac{\partial T}{\partial z} + u + d \tag{2.1}$$

where ρ, c_p, v, z, t, and T are density, specific heat, velocity, spacial coordinate, temporal coordinate, and temperature of the fluid, respectively. u and d are also disturbance and control action throughout the system. The PDE presented here can be generalized as

$$\begin{cases} \frac{\partial x(\xi,t)}{\partial t} = -\frac{\partial x(\xi,t)}{\partial \xi} + Bu(\xi,t) + D(t) \\ x(0,t) = 0 \\ y(t) = x(1,t) \end{cases} \tag{2.2}$$

which is equivalent to;

$$\begin{cases} \dot{x}(t) = \mathcal{A}x(t) + Bu(\xi,t) + D(t) \\ y(t) = \mathcal{C}x(t) \end{cases} \tag{2.3}$$

where $\mathcal{A}$ and $\mathcal{C}$ are corresponding operators. First step to design the regulator is defining the exosystem:

$$\begin{cases} \dot{v}(t) = Sv(t) \\ D(t) = Hv(t) \\ y_{tr}(t) = Qv(t) \end{cases} \tag{2.4}$$

and the associated error:

$$e = y(t) - y_{tr}(t) \tag{2.5}$$

which we would like to make it zero by a full state feedback controller which is desined as:

$$u(\xi,t) = Kx(\xi,t) + L(\xi)v(t). \tag{2.6}$$

In the expressions presented above, x is the state of the system, v is the state of exosystem, D is the disturbance, u is the control law, y is the output of the system, and y_{tr} is the trajectory reference. This system can be represented by a block diagram as follows:

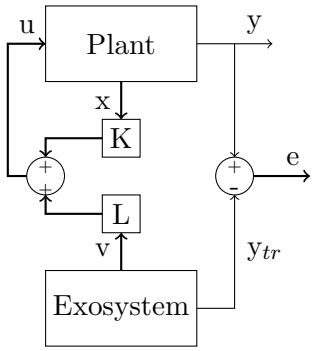

Figure 2.2: Block diagram representation of The system with IMC

The goal is the make the error zero as time goes by, and make the unforced system ($v = 0$) asymptotically stable. To start, the above system is restated as follows;

$$\begin{cases} \dot{x} = (\mathcal{A} + BK)x + (BL + H)v \\ \dot{v} = Sv \\ e = Cx - Qv \end{cases} \tag{2.7}$$

and then, the mapping below is introduced, which is necessary in order to meet the two goals stated above [13];

$$x(\xi, t) = \Pi(\xi)v(t) \tag{2.8}$$

where Π is defined as

$$\Pi(\xi) = \begin{bmatrix} \pi_1(\xi) & \pi_2(\xi) & \pi_3(\xi) \end{bmatrix}. \tag{2.9}$$

Applying the mapping into Eq. (2.7), we end up with constrained Sylvester equation;

$$\begin{cases} \Pi S = \mathcal{A}\Pi + B\Gamma + H \\ \mathcal{C}\Pi = Q \end{cases} \tag{2.10}$$

where Γ is defined as:

$$\Gamma = K\Pi + L \tag{2.11}$$

and Γ is

$$\Gamma = \begin{bmatrix} \gamma_1 & \gamma_2 & \gamma_3 \end{bmatrix} \tag{2.12}$$

2.2.2 Example - Case 1: Tracking trigonometric functions and Rejecting a Step

Consider the case when the disturbance is a step function and the reference trajectory is a combination of trigonometric functions; then we have;

$$S = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \qquad Q = [0 \quad 0 \quad 1] \qquad H = [1 \quad 0 \quad 0] \qquad B = 1 \tag{2.13}$$

then, Sylvester equation becomes;

$$\begin{cases} \begin{bmatrix} 0 & -\pi_3 & \pi_2 \end{bmatrix} = \begin{bmatrix} \frac{-\partial \pi_1}{\partial \xi} + \gamma_1 + 1 & \frac{-\partial \pi_2}{\partial \xi} + \gamma_2 & \frac{-\partial \pi_3}{\partial \xi} + \gamma_3 \end{bmatrix} \\ \begin{bmatrix} \pi_1(1) & \pi_2(1) & \pi_3(1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \\ \pi_i(0) = 0 \qquad for \quad i = 1, 2, 3 \end{cases} \tag{2.14}$$

where the last conditions come from the boundary condition of the plant. Now, the system of equations above is tried to be solved step by step; First,

$$\frac{-\partial \pi_1}{\partial \xi} + \gamma_1 + 1 = 0 \qquad \text{and} \qquad \pi_1(0) = \pi_1(1) = 0 \tag{2.15}$$

therefore;

$$\pi_1 = 0 \qquad \text{and} \qquad \gamma_1 = -1. \tag{2.16}$$

Now, if we consider the two last equalities of the first equation in (2.14), we have;

$$\begin{bmatrix}\pi_2\\ \pi_3\end{bmatrix}_\xi = \begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}\begin{bmatrix}\pi_2\\ \pi_3\end{bmatrix} + \begin{bmatrix}\gamma_2\\ \gamma_3\end{bmatrix} \tag{2.17}$$

or

$$\bar{\Pi}_\xi = \bar{A}\bar{\Pi} + \bar{B} \tag{2.18}$$

which has the solution;

$$\bar{\Pi} = e^{\bar{A}\xi}\bar{\Pi}(0) + \int_0^\xi e^{\bar{A}(\xi - z)}\bar{B}dz. \tag{2.19}$$

As, $\bar{\Pi}(0) = 0$, the solution reduces to;

$$\bar{\Pi} = \int_0^\xi e^{\bar{A}(\xi - z)}dz\bar{B}. \tag{2.20}$$

where $\bar{B}$ has to be found using the second condition in equation (2.14), i.e.

$$\bar{\Pi}(1) = \begin{bmatrix}0\\ 1.\end{bmatrix} \tag{2.21}$$

Imposing the condition, we come up with

$$\begin{bmatrix}0\\ 1\end{bmatrix} = e^{\bar{A}} \int_0^1 e^{-\bar{A}z}dz \begin{bmatrix}\gamma_2\\ \gamma_3\end{bmatrix} \tag{2.22}$$

in which $e^{\bar{A}z}$ is a matrix and is evaluated by Cayley Hamilton theorem here;

$$e^{\lambda_i z} = A_0 + A_1\lambda_i \qquad \text{for} \qquad i = 1, 2 \tag{2.23}$$

where λ_is are the eigenvalues of $\bar{A}$, that is, $\pm j$, and $j = \sqrt{-1}$. Here the A_0 and A_1 are found;

$$A_0 = \frac{e^{jz} + e^{-jz}}{2} \qquad A_1 = \frac{e^{jz} - e^{-jz}}{2j} \tag{2.24}$$

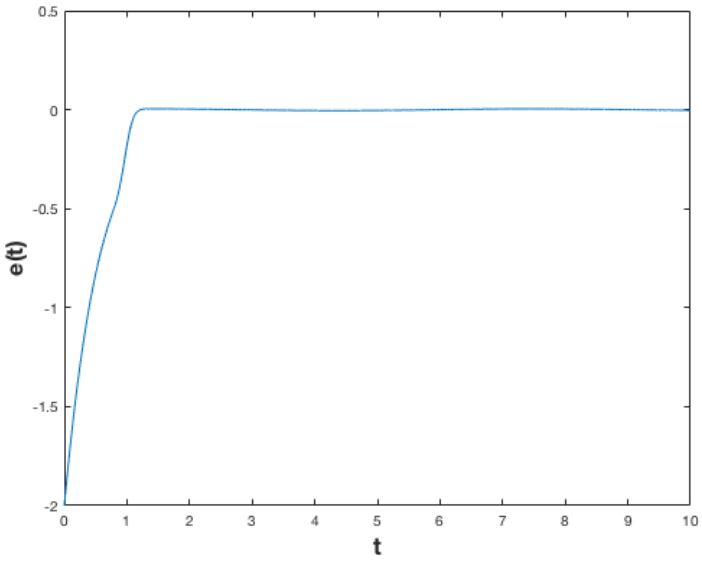

Figure 2.3: Error of the system (2.2) for Case 1

then,

$$e^{\bar{A}z} = cos(z)I + sin(z)A = \begin{bmatrix} cos(z) & sin(z) \\ -sin(z) & cos(z) \end{bmatrix} \tag{2.25}$$

where I is the identity Matrix. Having found the exponential of $\bar{A}$, γ can be calculated as;

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} cos(1) & -sin(1) \\ sin(1) & cos(1) \end{bmatrix} \int_0^1 \begin{bmatrix} cos(z) & -sin(z) \\ sin(z) & cos(z) \end{bmatrix} dz \begin{bmatrix} \gamma_2 \\ \gamma_3 \end{bmatrix} \tag{2.26}$$

which yields;

$$\gamma_2 = -0.5 \qquad \gamma_3 = 0.9152. \tag{2.27}$$

Finally, $\bar{\Pi}$ can be calculated as;

$$\bar{\Pi}(\xi) = \begin{bmatrix} cos(\xi) & sin(\xi) \\ -sin(\xi) & cos(\xi) \end{bmatrix} \int_0^\xi \begin{bmatrix} cos(z) & -sin(z) \\ sin(z) & cos(z) \end{bmatrix} dz \begin{bmatrix} -0.5 \\ 0.9152 \end{bmatrix} \tag{2.28}$$

from which the π_2 and π_3 are determined;

$$\pi_2 = -0.5sin(\xi) + 0.9152(1 - cos(\xi)) \qquad \pi_3 = 0.5(1 - cos(\xi)) + 0.9152sin(\xi). \tag{2.29}$$

Summarizing, we have found that;

$$\begin{aligned} &\pi_1 = 0 && \gamma_1 = -1 \\ &\pi_2 = -0.5sin(\xi) + 0.9152(1 - cos(\xi)) && \gamma_2 = -0.5 \\ &\pi_3 = 0.5(1 - cos(\xi)) + 0.9152sin(\xi) && \gamma_3 = 0.9152 \end{aligned}$$

To simulate the system and find the closed loop response, system (2.7) along with found Π and Γ is solved using numerical schemes and coded in MATLAB. Figure 2.3 shows how the error converges to zero with time. This verifies that the plant is stabilized. Figure 2.4 depicts the reference signal and the output. The output asymptotically approaches to the trajectory under controller action. Hence, our goals are both satisfied.

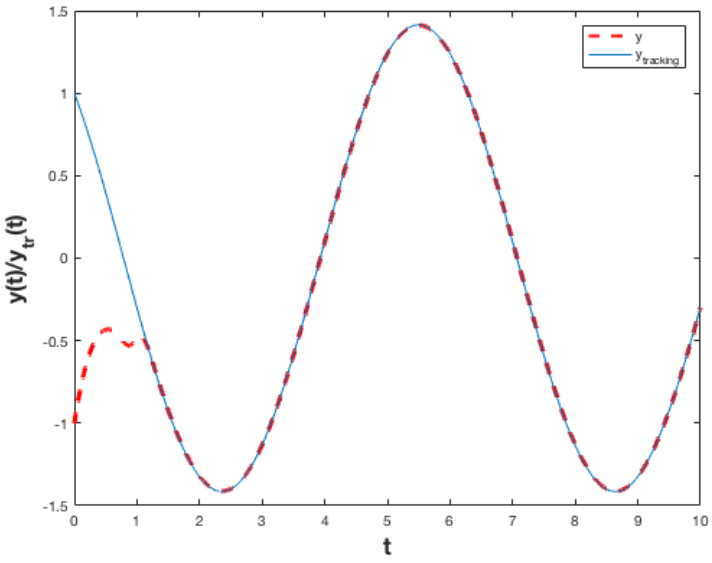

Figure 2.4: Output and the reference trajectory of the system (2.2) for Case 1

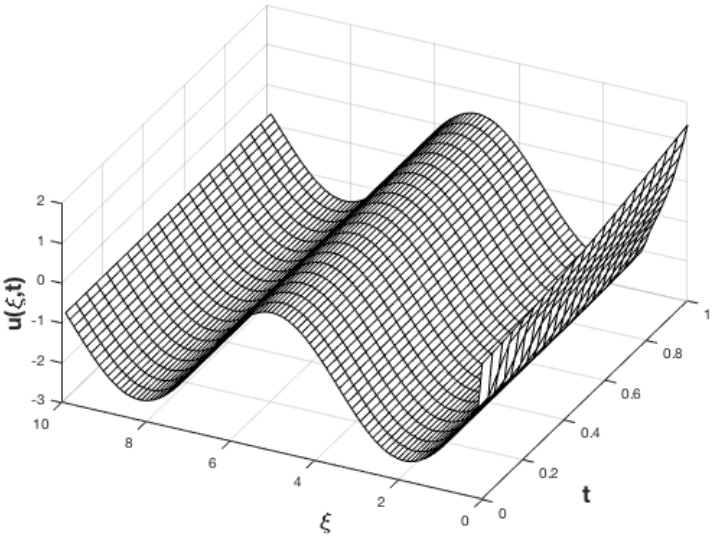

Figure 2.5: The controller response of Case 1

Figure 2.5 demonstrates the controller signal of the system. The controller action is

expectedly oscillatory as it has to track a trigonometric function.

2.2.3 Example - Case 2: Tracking a Step and Rejecting trigonometric functions

In this case we have the reverse case; say, step signal is to be tracked and a combination of trigonometric functions are to be rejected. We have:

$$S = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \qquad H = [0 \quad 0 \quad 1] \qquad Q = [1 \quad 0 \quad 0] \qquad B = 1 \tag{2.30}$$

then, Sylvester equation becomes:

$$\begin{cases} \begin{bmatrix} 0 & -\pi_3 & \pi_2 \end{bmatrix} = \begin{bmatrix} \frac{-\partial \pi_1}{\partial \xi} + \gamma_1 & \frac{-\partial \pi_2}{\partial \xi} + \gamma_2 & \frac{-\partial \pi_3}{\partial \xi} + \gamma_3 + 1 \end{bmatrix} \\ \begin{bmatrix} \pi_1(1) & \pi_2(1) & \pi_3(1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \\ \pi_i(0) = 0 \qquad for \quad i = 1, 2, 3 \end{cases} \tag{2.31}$$

Equations above is divided apart into two parts:

$$\frac{-\partial \pi_1}{\partial \xi} + \gamma_1 + 1 = 0 \qquad \text{and} \qquad \pi_1(0) = 0, \quad \pi_1(1) = 1 \tag{2.32}$$

and,

$$\begin{bmatrix} \pi_2 \\ \pi_3 \end{bmatrix}_{\xi} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \pi_2 \\ \pi_3 \end{bmatrix} + \begin{bmatrix} \gamma_2 \\ \gamma_3 + 1 \end{bmatrix} \tag{2.33}$$

where $\bar{B}$ can be found using

$$\bar{\Pi}(1) = \begin{bmatrix} 0 \\ 0. \end{bmatrix} \tag{2.34}$$

Which following the approach used in the last section leads to;

$\pi_1 = \xi$	$\pi_2 = 0$	$\pi_3 = 0$
$\gamma_1 = 1$	$\gamma_2 = 0$	$\gamma_3 = -1$

The results are shown below where the satisfactory function of the controller is made clear. Figure 2.6 shows that error converges to zero which implies that the system is asymptotically stable. Also, asymptotic tracking of the step signal is demonstrated in the figure 2.7.

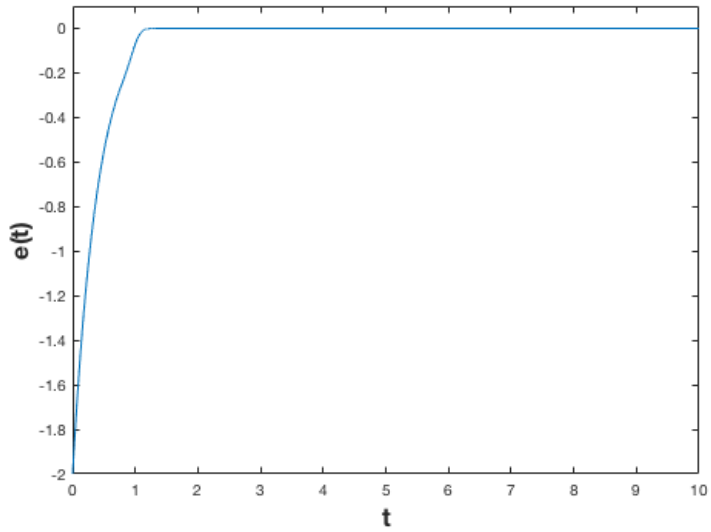

Figure 2.6: Error of the System

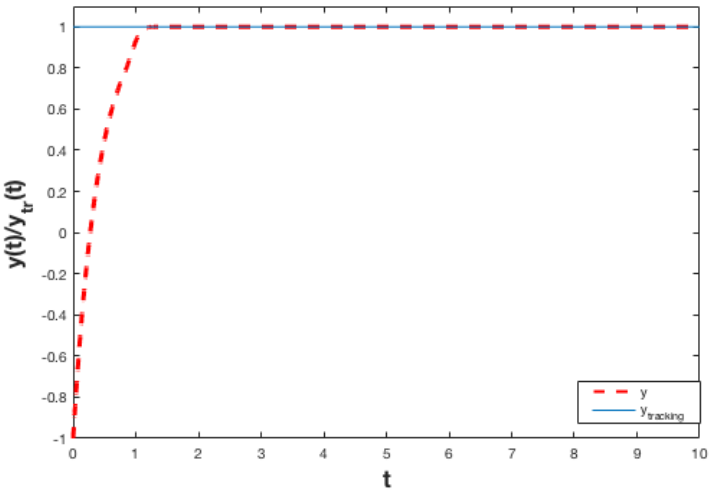

Figure 2.7: Output and reference trajectory of the system

Figure 2.8 depicts the controller action. Here again, the control act in an oscillatory manner as it must reject trigonometric functions. Furthermore, it should be mentions that as it is clear the controller is distributed throughout the system and is function of both time and space.

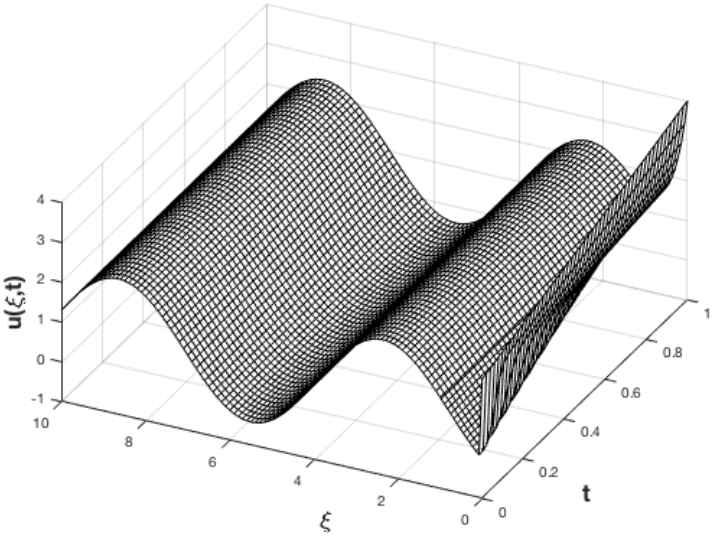

Figure 2.8: The design controller response

2.3 Boundary Controller

2.3.1 Posing the Problem

Consider the following tubular reactor:

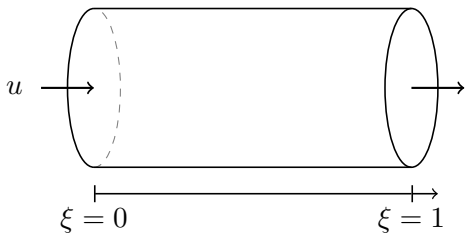

Figure 2.9: Tubular reactor with boundary acuation

which is similar to reactor in figure (2.2.1). The only difference here is the actuation method, which is on the boundary for the current reactor. Hence, the actuation term will appear in the boundary condition instead of system PDE. The plant can be stated as follows:

$$\begin{cases} \frac{\partial x(\xi,t)}{\partial t} = -\frac{\partial x(\xi,t)}{\partial \xi} + D(t) \\ x(0,t) = u(t) \\ y(t) = x(1,t) \end{cases} \tag{2.35}$$

where *D(t)* is the disturbance, *y(t)* is the output and *u(t)* is the controller to be found at the boundary. The above system can also be expressed as follows:

$$\begin{cases} \dot{x}(\xi,t) = \mathcal{A}x(\xi,t) + D(t) \\ \mathcal{B}x(\xi,t) = u(t) \\ y(t) = \mathcal{C}x(\xi,t) \end{cases} \tag{2.36}$$

Where $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ are corresponding operators.

The Exosystem which generates the reference trajectory and the disturbance is expressed below;

$$\begin{cases} \dot{v}(t) = Sv(t) \\ D(t) = Hv(t) \\ y_{tr} = Qv(t) \end{cases} \tag{2.37}$$

where y_{tr} is the reference trajectory. The error of the system is the difference between output and the reference signal;

$$e(t) = y(t) - y_{tr}(t) \tag{2.38}$$

2.3.2 Manipulation of the Plant

In order to get rid of the controller at the boundary and incorporate it in the differential equation, the following state transformation is used;

$$x(\xi, t) = P(\xi, t) + B(\xi)u(t) \tag{2.39}$$

which transform the plant to

$$\begin{cases} \dot{P}(\xi, t) = \mathcal{A}P(\xi, t) + \mathcal{A}Bu(t) - B\tilde{u}(t) + D(t) \\ \dot{u}(t) = \tilde{u}(t) \end{cases} \tag{2.40}$$

and the output to

$$y(t) = \mathcal{C}P(\xi, t) + \mathcal{C}Bu(t). \tag{2.41}$$

The transformed system is stated now in extended space as;

$$\begin{cases} \begin{bmatrix} \dot{u} \\ P \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ \mathcal{A}B & \mathcal{A} \end{bmatrix} \begin{bmatrix} u \\ P \end{bmatrix} + \begin{bmatrix} I \\ -B \end{bmatrix} \tilde{u} + \begin{bmatrix} 0 \\ I \end{bmatrix} D \\ y = \begin{bmatrix} \mathcal{C}B & \mathcal{C} \end{bmatrix} \begin{bmatrix} u \\ P \end{bmatrix} \end{cases} \tag{2.42}$$

which is equivalent to

$$\begin{cases} \dot{x}^e = A^e x^e + B^e \tilde{u} + D^e D \\ y = C^e x^e \end{cases} \tag{2.43}$$

When the transformed state is substituted into the boundary condition, following criteria should be met;

$$\mathcal{B}P(\xi, t) = 0 \qquad \mathcal{B}B(\xi) = 1 \tag{2.44}$$

Now, it is time to define the feedback controller as

$$\tilde{u} = Kx^e + Lv \tag{2.45}$$

then the plant becomes

$$\dot{x}^e = (A^e + B^e K)x^e + (B^e L + D^e H)v \tag{2.46}$$

where we want to design K in order to make this system stable when $v = 0$, i.e.

$$A^e + B^e K \tag{2.47}$$

must have negative eigenvalues. Moreover, we want that error approaches zero with time. A mapping is done in order to reach these goals, it is stated as

$$x^e(\xi, t) = \Pi(\xi)v(t) \tag{2.48}$$

where this makes the error zero, in other words

$$0 = C^e \Pi v - Qv \tag{2.49}$$

or

$$C^e \Pi = Q. \tag{2.50}$$

Furthermore, if we substitute the mapping (2.48) in equation (2.46) we come up with

$$\Pi S = A^e \Pi + B^e \Gamma + D^e H \tag{2.51}$$

where Γ is defined as

$$\Gamma = K\Pi + L. \tag{2.52}$$

In summery we have two goals:

1. Stabilizing
$$\dot{x}^e = (A^e + B^e K)x^e \tag{2.53}$$

2. Finding Π using
$$\begin{cases} \Pi S = A^e \Pi + B^e \Gamma + D^e H \\ C^e \Pi = Q \end{cases} \tag{2.54}$$

Next section explores the derivation procedure needed to find K and Π through an example.

2.3.3 Example

Suppose we want to track step change and reject a combination of trigonometric function, Hence;

$$S = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \qquad Q = [1 \quad 0 \quad 0] \qquad H = [0 \quad 0 \quad 1] \tag{2.55}$$

and let's define

$$B(\xi) = 1 \tag{2.56}$$

which is in accordance with

$$\mathcal{B}B(\xi) = 1. \tag{2.57}$$

Then we have

$$\mathcal{A}B(\xi) = 0 \tag{2.58}$$

and,

$$A^e = \begin{bmatrix} 0 & 0 \\ 0 & \mathcal{A} \end{bmatrix} \qquad B^e = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad C^e = \begin{bmatrix} 1 & \mathcal{C} \end{bmatrix} \quad D^e = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \tag{2.59}$$

with which we have all the necessary parameters to solve the problem. Let's start with equation (2.51);

$$\begin{bmatrix} 1 & \mathcal{C} \end{bmatrix} \begin{bmatrix} \pi_{11} & \pi_{12} & \pi_{13} \\ \pi_{21} & \pi_{22} & \pi_{23} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{2.60}$$

which leads to

$$\begin{cases} \pi_{11} + \pi_{21}(1) = 1 \\ \pi_{12} + \pi_{22}(1) = 0 \\ \pi_{13} + \pi_{23}(1) = 0 \end{cases} . \tag{2.61}$$

On the other hand equation (2.50) becomes

$$\begin{aligned} \begin{bmatrix} \pi_{11} & \pi_{12} & \pi_{13} \\ \pi_{21} & \pi_{22} & \pi_{23} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} &= \begin{bmatrix} \pi_{11} & \pi_{12} & \pi_{13} \\ \pi_{21} & \pi_{22} & \pi_{23} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & \mathcal{A} \end{bmatrix} \\ &+ \begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{bmatrix} \gamma_1 & \gamma_2 & \gamma_3 \end{bmatrix} \\ &+ \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2.62}$$

or

$$\begin{bmatrix} 0 & -\pi_{13} & \pi_{12} \\ 0 & -\pi_{23} & \pi_{22} \end{bmatrix} = \begin{bmatrix} \gamma_1 & \gamma_2 & \gamma_3 \\ -\frac{\partial \pi_{21}}{\partial \xi} - \gamma_1 & -\frac{\partial \pi_{22}}{\partial \xi} - \gamma_2 & -\frac{\partial \pi_{23}}{\partial \xi} - \gamma_3 + 1 \end{bmatrix} \tag{2.63}$$

which is equivalent to

$$\begin{cases} \gamma_1 = 0 \\ \gamma_2 = -\pi_{13} \\ \gamma_3 = \pi_{12} \\ -\frac{\partial \pi_{21}}{\partial \xi} - \gamma_1 = 0 \\ -\frac{\partial \pi_{22}}{\partial \xi} - \gamma_2 = -\pi_{23} \\ -\frac{\partial \pi_{23}}{\partial \xi} - \gamma_3 + 1 = \pi_{22} \end{cases} . \tag{2.64}$$

and can be solved considering boundary condition (2.61) along with operator conditions;

$$\pi_{2j}(0) = 0 \qquad j = 1, 2, 3. \tag{2.65}$$

At first, we try to solve the first differential equation in (2.64);

$$\begin{cases} \frac{\partial \pi_{21}}{\partial \xi} = 0 \qquad \pi_{21}(0) = 0 \\ \pi_{11} + \pi_{21}(1) = 1 \end{cases} \tag{2.66}$$

which leads to

$$\pi_{11} = 1 \qquad \pi_{21} = 0. \tag{2.67}$$

Now, let's express the other two differential equations in matrix form, with corresponding boundary conditions;

$$\begin{cases} \begin{bmatrix} \pi_{22} \\ \pi_{23} \end{bmatrix}_\xi = \begin{bmatrix} 0 & 1 \\ -1 & q0 \end{bmatrix} \begin{bmatrix} \pi_{22} \\ \pi_{23} \end{bmatrix} + \begin{bmatrix} -\gamma_2 \\ 1 - \gamma_3 \end{bmatrix} \\ \begin{bmatrix} \pi_{22}(0) \\ \pi_{23}(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} \pi_{22}(1) \\ \pi_{23}(1) \end{bmatrix} = \begin{bmatrix} -\gamma_3 \\ \gamma_2 \end{bmatrix} \end{cases} \tag{2.68}$$

or,

$$\begin{cases} \bar{\Pi} = \bar{A}\bar{\Pi} + \bar{B} \\ \bar{\Pi}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \bar{\Pi}(1) = \begin{bmatrix} -\gamma_3 \\ \gamma_2 \end{bmatrix} \end{cases} . \tag{2.69}$$

We have already known that this equation has the general solution of;

$$\bar{\Pi} = e^{\bar{A}\xi}\bar{\Pi}(0) + \int_0^{\xi} e^{\bar{A}(\xi - z)}\bar{B}dz \tag{2.70}$$

which by substitution of its parameters and simplifying the exponential term becomes;

$$\bar{\Pi} = \int_0^{\xi} \begin{bmatrix} cos(\xi - z) & sin(\xi - z) \\ -sin(\xi - z) & cos(\xi - z) \end{bmatrix} dz \begin{bmatrix} -\gamma_2 \\ 1 - \gamma_3 \end{bmatrix}. \tag{2.71}$$

The unknowns -γ_2 and γ_3- are found using value of $\bar{P}i$ at the boundary ξ=1. Then π_{22} and π_{23} are calculated. The result is as follows;

$$\begin{cases} \pi_{22} = -0.841471sin(\xi) + 0.540302(1 - cos(\xi)) \\ \pi_{23} = 0.841471(1 - cos(\xi)) + 0.540302sin(\xi) \\ \gamma_2 = 0.841471 \\ \gamma_3 = 0.459698. \end{cases} \tag{2.72}$$

In summary, components of Π and Γ found are as follows;

$$\Gamma = \begin{bmatrix} 0 & 0.841471 & 0.459698 \end{bmatrix}$$

$$\Pi = \begin{bmatrix} 0 & 0.459698 & -0.841471 \\ 0 & \text{-0.841471sin}(\xi)\text{+0.540302(1-cos}(\xi)\text{)} & \text{0.841471(1-cos}(\xi)\text{)+0.540302sin}(\xi) \end{bmatrix}$$

So far we have found all the necessary requirement for the design. The last part is the selection of K. It is shown that for the considered plant, the system is unconditionally stable around the origin. However, the second element of the gain has to be a boundary operator as u is only function of time and not the space. Therefore it is selected as;

$$K = \begin{bmatrix} -1 & -\beta \end{bmatrix} \tag{2.73}$$

with which L is found:

$$L = \begin{bmatrix} \gamma_1 + \pi 11 + \pi_{21}(0) & \gamma_2 + \pi 12 + \pi_{22}(0) & \gamma_3 + \pi 13 + \pi_{23}(0) \end{bmatrix} \tag{2.74}$$

.

At last, inserting K and L into the equation (2.46) we end up with the final form of the extend system, from which we are able to find the state and controller responses;

$$\begin{bmatrix} \dot{u} \\ P \end{bmatrix} = \begin{bmatrix} -1 & -\beta \\ 1 & \mathcal{A} + \mathcal{B} \end{bmatrix} \begin{bmatrix} u \\ P \end{bmatrix} + \begin{bmatrix} L_1 & L_2 & L_3 \\ -L_1 & -L_2 & 1 - L_3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{2.75}$$

which can be expanded into two separate ordinary differential equations;

$$\begin{cases} \dot{u} = -u + [L_1 \quad L_2 \quad L_3]v \\ \dot{P} = u - \frac{\partial P}{\partial \xi} + [-L_1 \quad -L_2 \quad 1 - L_3]v \end{cases} \tag{2.76}$$

2.3.4 Simulation

The first step in simulating the system is to find the exosystem state. This is reached by applying modified Euler method to equation (2.37). The state of exosystem is represented here:

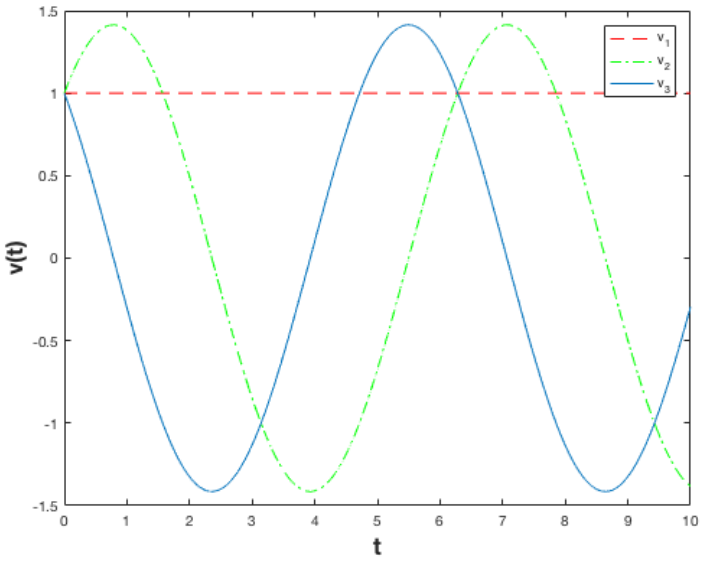

Figure 2.10: Exosystem (2.37) state evolution

The Euler method is applied on first equation of (2.76) to find the controller response. The response is shown below;

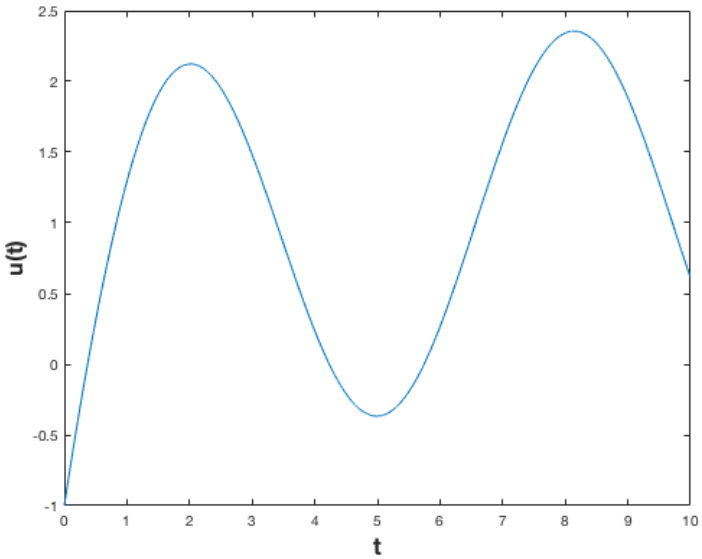

Figure 2.11: The controller response of system (2.35)

This oscillatory response is well expected as the controller is to reject a combination of Sine and Cosine function constantly. At last, the response of system from second equation of (2.76) is found numerically by forward discretization in time and backward in space to approximate the derivatives with respect to time and space, respectively. The results are depicted in the following figures and show how the controller stabilized the system and tracks the reference trajectory. Figure 2.12 represent the error vs time and shows how it converges to zero.

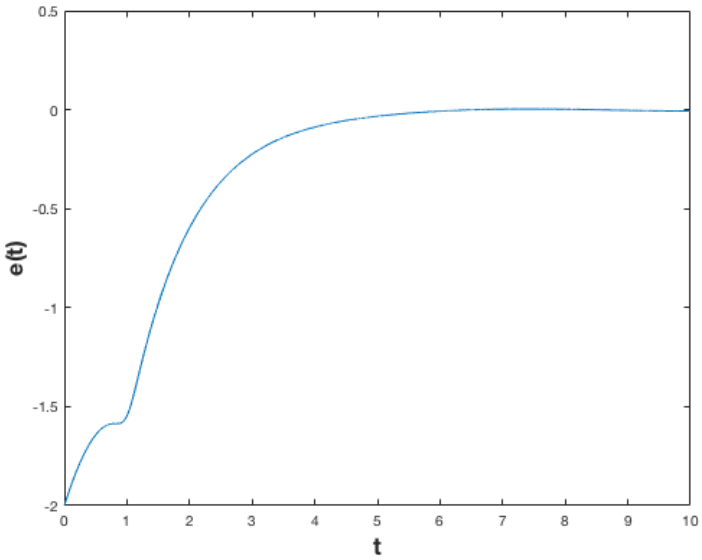

Figure 2.12: Error of the system (2.35)

Figure 2.13 on the other hand, demonstrate the ability of the controller to track the trajectory which is a step change in this example.

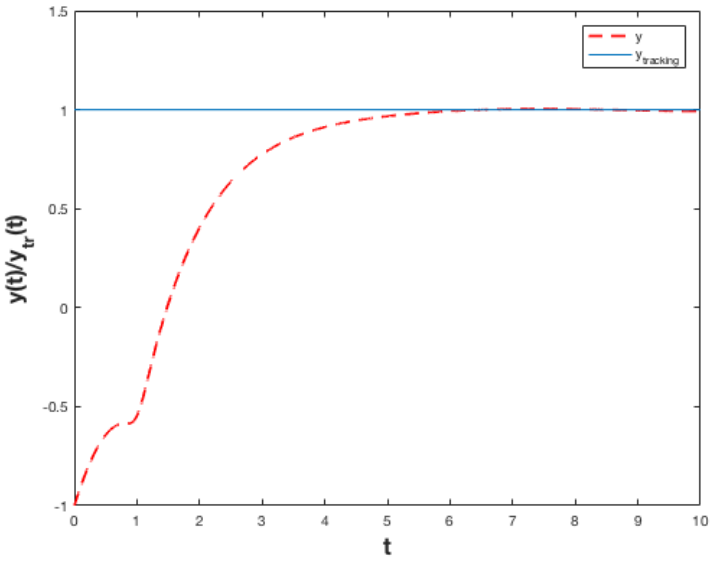

Figure 2.13: Tracking the reference trajectory for system (2.35)

Chapter 3

Backstepping Control of Hyperbolic Systems

3.1 Introduction

The main goal of backstepping is to impose control over the boundary of an unstable system such as the parabolic one below

$$u_t(x,t) = u_{xx}(x,t) + \lambda u(x,t) \tag{3.1}$$

$$u(0,t) = 0$$

$$u(1,t) = U(t)$$

$$u(x,0) = u_0(t)$$

to acquire or maintain a desirable state, where *u(x,t)* is the system state; a function of space and time. This goal is reached by transforming the system to another one with desirable stability properties (say, exponential stability):

$$w_t(x,t) = w_{xx}(x,t) \tag{3.2}$$

$$w(0,t) = 0$$

$$w(1,t) = 0$$

$$w(x,0) = w_0(t).$$

To reach the target system, the following transformation

$$w(x,t) = u(x,t) - \int_0^x k(x,y)u(y,t)dy \tag{3.3}$$

along with feedback control law

$$u(1,t) = \int_0^1 k(1,y)u(y,t)dy \tag{3.4}$$

is used. The transformation in fact is a Volterra integral transform which has a triangular structure, and *k(x,y)* is its kernel. One of the main features of this transformation is its invertibility, so that the stability of target system is equivalent to stability of closed loop system which includes the system itself and its boundary feedback. [20].

The same approach is followed in this chapter in order to find a boundary controller for hyperbolic partial differential equations. The chapter is based on the works of Krstic and Smyshlyaev [21,31].

3.2 Hyperbolic Partial Differential Equations

3.2.1 Theory

Hyperbolic system almost in its most general form can be stated as follows:

$$v_t(x,t) = v_x(x,t) + \lambda(x)v(x,t) + \bar{g}(x)v(0,t) + \int_0^x \bar{f}(x,y)v(y,t)dy \tag{3.5}$$

for 0 < x < 1 and initial condition

$$v(x,t) = v_0(x) \tag{3.6}$$

and boundary condition

$$v(1,t) = \bar{U}(t). \tag{3.7}$$

First, it has been attempted to simplify the plant, starting with the state transformation

$$v = e^{-\int_0^x \lambda(\xi)d\xi}u \tag{3.8}$$

we are going to replace v by u. Note that the *(x,t)* is dropt for simplicity. To achieve this, derivatives of v are to be calculated,

$$v_x = -\lambda u e^{-\int_0^x \lambda d\xi} + e^{-\int_0^x \lambda d\xi}u_x \tag{3.9}$$

$$v_t = e^{-\int_0^x \lambda d\xi}u_t \tag{3.10}$$

and then by substituting into equation (3.5) we get to the equation

$$e^{-\int_0^x \lambda d\xi}u_t = -\lambda u e^{-\int_0^x \lambda d\xi} + u_x e^{-\int_0^x \lambda d\xi} + \lambda u e^{-\int_0^x \lambda d\xi} + \bar{g}u(0,t) + \int_0^x \bar{f}(x,y)e^{-\int_0^y \lambda d\xi}u(y,t) \tag{3.11}$$

which by rearranging it, we end up to the transformed plant

$$u_t = u_x + g(x)u(0,t) + \int_0^x f(x,y)u(y,t)dy \tag{3.12}$$

$$u(1,t) = U(t) \tag{3.13}$$

$$f(x,y) = \bar{f}(x,y)e^{\int_y^x \lambda d\xi} \tag{3.14}$$

$$g(x) = \bar{g}e^{\int_0^x \lambda(\xi)d\xi} \tag{3.15}$$

$$U(t) = \bar{U}(t)e^{\int_0^1 \lambda(\xi)d\xi} \tag{3.16}$$

Getting to the New simplified plant, we start backstepping method by applying a Volterra transformation

$$w = u - \int_0^x k(x,y)u(y)dy \tag{3.17}$$

as a state transform, along with the feedback

$$u(1,t) = \int_0^1 k(1,y)u(y)dy \tag{3.18}$$

where *k(x,y)* is the kernel of integral equation. The plant is desired to be transformed into the target system below

$$w_t(x,t) = w_x(x,t) \tag{3.19}$$

$$w(1,t) = 0 \tag{3.20}$$

where it is in fact a delay line with a unit delay and has a stable solution

$$w(x,t) = \begin{cases} w_0(t+x) & 0 \le x+t \le 1 \\ 0 & x+t \ge 1 \end{cases} \tag{3.21}$$

where $w_0(x)$ is the initial condition.

Applying Volterra Transformation

First, the derivatives of transformation (3.17) with respect of x and t has to be taken. then, they will be substituted into target system in order to evaluate the validity of the transformation. Proceeding so, we reach at the end to a set of conditions which they must hold. These conditions are in fact a system of hyperbolic partial differential equations which its solution determines the kernel. The procedure described is demonstrated below;

$$w_x = u_x - k(x,x)u - \int_0^x k_x(x,y)u(y)dy \tag{3.22}$$

$$
\begin{aligned}
w_t &= u_t - \int_0^x k(x,y)u_t(y)dy \\
&= (u_x + g(x)u(0) + \int_0^x f(x,y)u(y)dy) \\
&\quad - \int_0^x k(x,y)(u_x + g(y)u(0) + \int_0^y f(y.\xi)u(\xi)d\xi)dy \\
&= u_x + (g(x) - \int_0^x k(x,y)g(y)dy)u(0) + \int_0^x f(x,y)u(y)dy \\
&\quad - \int_0^x k(x,y)\int_0^y f(y,\xi)u(\xi)d\xi dy - \int_0^x k(x,y)u_x(y)dy \\
&= u_x + u(0)(g(x) + \int_0^x k(x,y)g(y)dy) \\
&\quad + \int_0^x u(y,t)(f(x,y) - \int_y^x k(x,\xi)f(\xi,y)d\xi)dy \\
&\quad - k(x,x)u(x,t) + k(x,0)u(0) + \int_0^x k_y(x,y)u(y)dy
\end{aligned}
\tag{3.23}
$$

Now substituting the derivatives (3.22) and (3.23) into the target system (3.19) and(3.20)

$$
\begin{aligned}
&u_x - k(x,x)u - \int_0^x k_x u(y)dy = u_x - k(x,x)u \\
&+ u(0,t)(g - \int_0^x kgdy + k(x,0)) \\
&+ \int_0^x u(f - \int_y^x k(x,\xi)f(\xi,y)d\xi + k_y)dy
\end{aligned}
\tag{3.24}
$$

then, rearranging and simplifying the equation above we are led to the kernel system

$$
k_y(x,y) + k_x(x,y) = \int_0^x k(x,\xi)f(\xi,y)d\xi - f(x,y) \tag{3.25}
$$

$$
k(x,0) = \int_0^x k(x,y)g(y)dy - g(x) \tag{3.26}
$$

Now the only thing remained is to solve the kernel system which could be done numerically or explicitly. When solved, controller is able to be found using using (3.18). To demonstrate this, two different sample plants are stabilized using the derived equations and the kernel system in the next section.

3.2.2 Examples

Example One

First example is a simple one dimensional hyperbolic equation with an extra term for effect of boundary on the system. The coefficient of the boundary term is an exponential function of space which can make the system unstable. Therefore, backstepping method is chosen

to stabilize the system.The plant is as follow;

$$u_t = u_x + ge^{bx}u(0) \tag{3.27}$$

where g and b are constants. The kernel system of this plant based on equations 3.25 and 3.26 is

$$k_y + k_x = 0 \tag{3.28}$$

$$k(x,0) = \int_0^x k(x,y)ge^{by}dy - ge^{bx}. \tag{3.29}$$

Implementing the method of characteristics on equation (3.28) we have

$$\frac{dy}{1} = \frac{dx}{1} = \frac{dk}{0} \tag{3.30}$$

which yields

$$y - x = \phi \tag{3.31}$$

$$k = \psi \tag{3.32}$$

which are the characteristic lines, therefore

$$F(\phi,\psi) = 0 \quad \Rightarrow \quad k = \phi(x-y) \tag{3.33}$$

now substituting this into (3.26) we reach

$$\phi(x) = \int_0^x \phi(x-y)ge^{by}dy - ge^{bx} \tag{3.34}$$

which by taking its laplace transform, ϕ(x) can be found as follows;

$$\hat{\phi}(s) = \hat{\phi}(s)\frac{g}{s-b} - \frac{g}{s-b} \tag{3.35}$$

or

$$\hat{\phi}(s) = -\frac{\frac{g}{s-b}}{1-\frac{g}{s-b}}. \tag{3.36}$$

Taking inverse Laplace transform of last equation we end up with

$$\phi(x) = -ge^{(b+g)x} \tag{3.37}$$

along with equation (3.33) where we know that k(x,y)=ϕ(x-y), so

$$k(x,y) = -ge^{(b+g)(x-y)} \tag{3.38}$$

Finally, knowing the kernel, controller law is also found at the boundary by equation (3.18)

$$u(1,t) = \int_0^1 -ge^{(b+g)(1-y)}u(y)dy. \tag{3.39}$$

The closed loop system behavior using the expression developed is depicted here

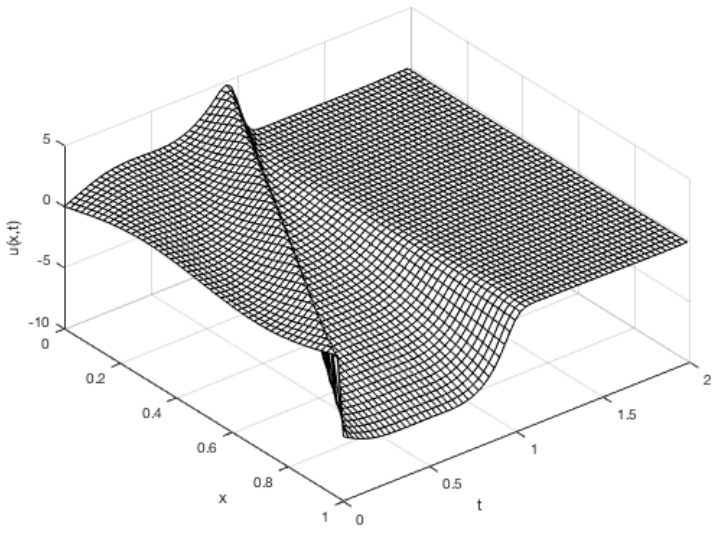

Figure 3.1: Closed loop behavior hyperbolic partial differential equation given by Eq. (3.27)

As it is obvious in the figure the controller is able to completely stabilize the system. When the controller is not involved the open loop system would be unstable as demonstrated below.

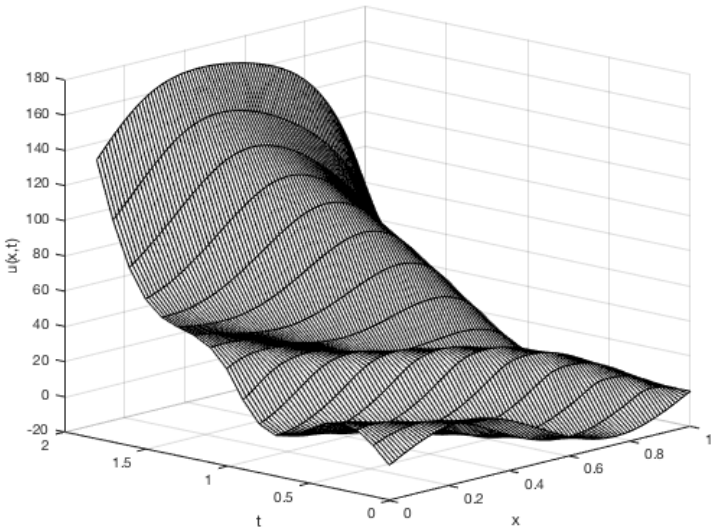

Figure 3.2: Open loop behavior of hyperbolic partial differential equation given by Eq. (3.27)

Example Two

The desired plant to implement control on is

$$u_t(x,t) = u_x(x,t) + \int_0^x f e^{\lambda(x-y)} u(y,t) dy \tag{3.40}$$

The kernel system based on equations (3.25) and (3.26) is

$$k_x + k_y = \int_y^x k(x,\xi) f e^{\lambda(\xi-y)} d\xi - f e^{\lambda(x-y)} \tag{3.41}$$

$$k(x,0) = 0. \tag{3.42}$$

To get rid of the integral term we differentiate equation (3.41) with respect to y

$$\begin{aligned} k_{xy} + k_{yy} &= -kf - \int_y^x k(x,\xi) f\lambda e^{\lambda(\xi-y)} d\xi + f\lambda e^{\lambda(x-y)} \\ &= -kf - \lambda(\int_y^x k(x,\xi) f e^{\lambda(\xi-y)} d\xi - f e^{\lambda(x-y)}) \\ &= -kf - \lambda k_x - \lambda k_y \end{aligned} \tag{3.43}$$

Differentiating the original equation increases the order of differential equation, hence to solve the new PDE system we need an extra condition, which is introduced at x=y this way;

$$k_x(x,x) + k_y(x,x) = -f \tag{3.44}$$

which is equivalent to

$$\frac{d}{dx} k(x,x) = -f \tag{3.45}$$

with which we get to

$$k(x,x) = -fx. \tag{3.46}$$

Now, introducing two change of variables

$$z = 2x - y \tag{3.47}$$

$$k(x,y) = p(z,y) e^{\lambda(z-y)/2} \tag{3.48}$$

and taking their derivatives

$$\frac{dk}{dx} = \frac{\partial k}{\partial z}\frac{\partial z}{\partial x} \tag{3.49}$$

$$k_x = e^{\lambda(z-y)/2}(2p_z + \lambda p) \tag{3.50}$$

$$k_{xy} = \frac{\partial k_x}{\partial y} + \frac{\partial k_x}{\partial z}\frac{dz}{dy} \tag{3.51}$$

$$
\begin{aligned}
k_{xy} &= 2(p_{zy}e^{\lambda(z-y)/2} + p_z(-\lambda/2)e^{\lambda(z-y)/2} + \frac{\lambda}{2}p_y e^{\lambda(z-y)/2} \\
&\quad + p(\lambda/2)(-\lambda/2)e^{\lambda(z-y)/2}) - 2(p_{zz}e^{\lambda(z-y)/2} + p_z(\lambda/2)e^{\lambda(z-y)/2} \\
&\quad + (\lambda/2)p_z e^{\lambda(z-y)/2} + (\lambda/2)(\lambda/2)pe^{\lambda(z-y)/2}) \\
&= e^{\lambda(z-y)/2}(2p_{zy} - 2p_{zz} + \lambda p_y - 3\lambda p_z - \lambda^2 p)
\end{aligned} \tag{3.52}
$$

$$
k_y = \frac{\partial k}{\partial y} + \frac{\partial k}{\partial z}\frac{\partial z}{\partial y} \tag{3.53}
$$

$$
k_y = e^{\lambda(z-y)/2}(p_y - p_z - \lambda p) \tag{3.54}
$$

$$
k_{yy} = \frac{\partial k_y}{\partial y} + \frac{\partial k_y}{\partial z}\frac{dz}{dy} \tag{3.55}
$$

$$
k_{yy} = -\frac{\lambda}{2}e^{\lambda(z-y)/2}(p_y - p_z - \lambda p) + e^{\lambda(z-y)/2}(p_{yy} - p_{yz} - \lambda p_y) \tag{3.56}
$$

$$
- (e^{\lambda(z-y)/2}(p_{yz} - p_{zz} - \lambda p_z) + \frac{\lambda}{2}e^{\lambda(z-y)/2}(p_y - p_z - \lambda p)) \tag{3.57}
$$

$$
= e^{\lambda(z-y)/2}(p_{yy} - 2p_{yz} + p_{zz} + 2\lambda p_z + \lambda^2 p - 2\lambda p_y) \tag{3.58}
$$

and then replacing the derivatives into equation (3.43)

$$
\begin{aligned}
&(2p_{zy} - 2p_{zz} + \lambda p_y - 3\lambda p_z - \lambda^2 p) \\
&\qquad + (p_{yy} - 2p_{yz} + p_{zz} + 2\lambda p_z + \lambda^2 p - 2\lambda p_y) \\
&\qquad\qquad + \lambda(2p_z + \lambda p + p_y - p_z - \lambda p) = -fp
\end{aligned} \tag{3.59}
$$

we end up with the transformed kernel system

$$
p_{yy} - p_{zz} = -fp \tag{3.60}
$$

$$
p(z,0) = 0 \tag{3.61}
$$

$$
p(z,z) = -fz \tag{3.62}
$$

which has the solution

$$
p(z,y) = -2fy\frac{I_1(\sqrt{f(z^2-y^2)})}{\sqrt{f(z^2-y^2)}}. \tag{3.63}
$$

Hence, the original kernel is

$$
k(x,y) = -fe^{\lambda(x-y)}y\frac{I_1(2\sqrt{fx(x-y)})}{\sqrt{fx(x-y)}}. \tag{3.64}
$$

Knowing the kernel, the control law will be

$$
u(1,t) = -\int_0^1 fe^{\lambda(1-y)}y\frac{I_1(2\sqrt{f(1-y)})}{\sqrt{f(1-y)}}u(y,t)dy \tag{3.65}
$$

which will be used to stabilize the system. Following is the figure showing the behavior of open loop system which demonstrate how the system grows with time.

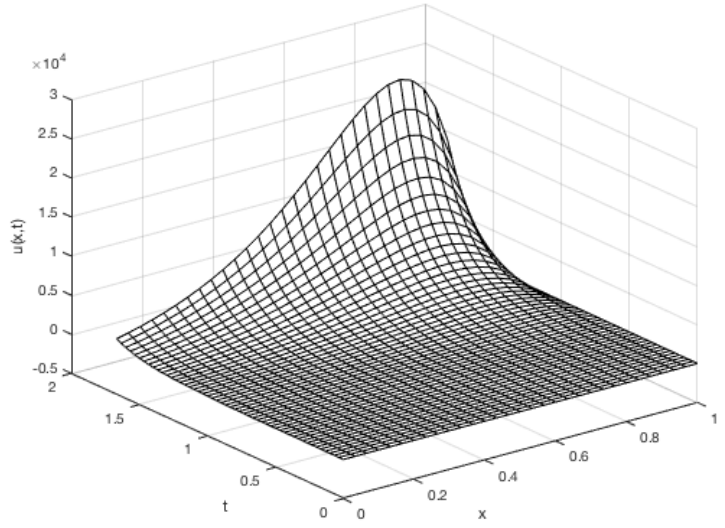

Figure 3.3: Open loop system behavior of second example of hyperbolic partial differential equation

On the other hand, when the controller is applied the system is stabilized as shown below;

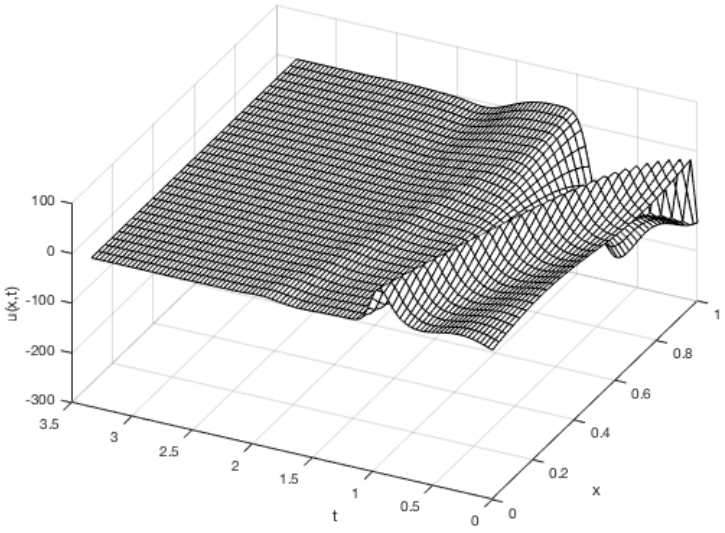

Figure 3.4: Closed loop system behavior of second example of hyperbolic partial differential equation

which verify the performance of the controller.

Chapter 4

Backstepping Controller for a Special Case of First Order Hyperbolic Partial Differential Equation

4.1 Introduction

In the current chapter, a particular unstable first order hyperbolic PDE system is considered where the boundary at the inlet is coupled with the other boundary. As usual in backstepping method, the plant is mapped into a new desirable asymptotically stable target system by use of Volterra integral transformation. Afterwards, a reference signal is selected at the boundary, state trajectory is generated and the ability of controller to track the trajectory is examined. The method will be completely elaborated in the next sections by illustrations.

4.2 Motivation

Consider the tubular reactor with recycle illustrated in figure.

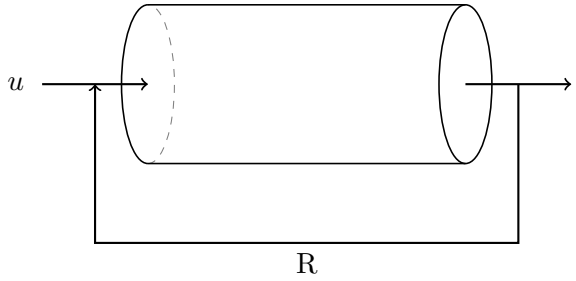

Figure 4.1: Tubular reactor with recycle

Under the following assumption:

- the flow is incompressible,

- the flow is one dimensional,
- the reaction is exothermic,
- velocity is high enough to make the diffusion term negligible,
- the reaction occurring in the system is first order.

one can model the species or temperature distribution in the reactor as

$$\rho\frac{\partial x(z,t)}{\partial t} + \rho v\frac{\partial x(z,t)}{\partial z} = kx(z,t) + D(t) \tag{4.1}$$

where x is the concentration or temperature of the system, ρ is the density of the fluid, v is the velocity of the fluid, k is the reaction rate coefficient or conductivity, z is the spacial variable, and t is the temporal variable of the system. Also, D added to include any source of unstability/Disturbance. First term of the equation (4.1) on the left hand side is the transient term which shows the rate of changes of the state, second term on the right hand side represents the transport by convection, and the third term is the representative of the first order reaction in the system or the energy degenerated. The boundary condition of the system of is stated as follows [24],

$$x(0,t) = Ru(t) + (1-R)x(1,t) \tag{4.2}$$

where $0 < R \leq 1$ is a ratio defined as relative flow of input stream to the recycle stream, and *u(t)* is input to the system.

Continuity equation for the reactor based on the aforementioned assumptions is

$$\frac{dv}{dz} = 0 \tag{4.3}$$

which means that the velocity of the fluid inside the reactor is constant.

The model obtained here may demonstrate unstable behavior base on the term *D(t)*. Our goal is to design a backstepping feedback controller which stabilizes the unstable reactor and track a desired trajectory. This is elaborated in the following sections.

4.3 Backstepping Controller Design

If the dimensionless velocity is assumed to be one, The equation (4.1) could be generalized as:

$$\frac{\partial x(z,t)}{\partial t} = -\frac{\partial x(z,t)}{\partial z} + a(z)x(z,t) + g(z)x(1,t) \tag{4.4}$$

for $0 \leq z \leq 1$, and $t \in \mathbb{R}^+$ with initial condition $x(z,t) = x_0(z)$, and equation (4.2) as the boundary condition. The last term in the equation (4.4) is added to include the destabilizing effect of disturbances originating from the boundary. Assuming $a(z) \leq 0$, and *g(z)* are continuous functions, we define our goal to stabilize the system at the zero

equilibrium point by the controller *u(t)*. To do so, we first start by manipulating the equation to get to a desired form. Defining a coordinate transformation

$$\xi = 1 - z \tag{4.5}$$

the plant is transformed to

$$\frac{\partial x(\xi,t)}{\partial t} = \frac{\partial x(\xi,t)}{\partial \xi} + A(\xi)x(\xi,t) + G(\xi)x(0,t) \tag{4.6}$$

$$x(1,t) = Ru(t) + (1-R)x(0,t) \tag{4.7}$$

where $A(\xi) = a(1-z)$, and $G(\xi) = g(1-z)$. It is also shown that the reaction term in a first order hyperbolic PDE can be eliminated ([7], [8], [21]). Applying a state transformation

$$x(\xi,t) = e^{-\int_0^\xi A(\alpha)d\alpha}\chi(\xi,t) \tag{4.8}$$

leads to the plant below,

$$\frac{\partial \chi(\xi,t)}{\partial t} = \frac{\partial \chi(\xi,t)}{\partial \xi} + \lambda(\xi)\chi(0,t) \tag{4.9}$$

$$\chi(1,t) = RU(t) + (1-R)e^{\int_0^1 A(\alpha)d\alpha}\chi(0,t) \tag{4.10}$$

where

$$\lambda(\xi) = e^{\int_0^\xi A(\alpha)d\alpha}G(\xi) \tag{4.11}$$

$$U(t) = e^{\int_0^1 A(\alpha)d\alpha}u(t). \tag{4.12}$$

We would like to map the plant (4.9)-(4.10) into the target system below

$$\frac{\partial w(\xi,t)}{\partial t} = \frac{\partial w(\xi,t)}{\partial \xi} \tag{4.13}$$

$$w(1,t) = (1-R)e^{\int_0^1 A(\alpha)d\alpha}w(0,t) \tag{4.14}$$

by utilizing the Volterra integral transformation [21]

$$w(\xi,t) = \chi(\xi,t) - \int_0^\xi k(\xi,y)\chi(y,t)dy \tag{4.15}$$

along with the feedback

$$U(t) = \frac{1}{R}\int_0^1 k(1,y)\chi(y,t)dy \tag{4.16}$$

where $k(\xi,y)$ is the kernel of the transformation. Our mission from now on is to find a kernel which satisfy this transformation.

Lemma 1. The target system (4.13)-(4.14) is asymptotically stable.

Proof. Consider the Lyapunov function

$$V(t) = \frac{1}{2}\int_0^1 w^2(\xi,t)d\xi = ||w(\xi,t)||^2, \tag{4.17}$$

its time derivative is;

$$\begin{aligned}\dot{V}(t) &= \textstyle\int_0^1 w(\xi,t)\dot{w}(\xi,t)d\xi \\ &= \textstyle\int_0^1 w(\xi,t)\frac{\partial w(\xi,t)}{\partial \xi}d\xi \\ &= w^2(1,t) - w^2(0,t) \\ &= (1-R)^2 e^{\int_0^1 2A(\alpha)d\alpha} w^2(0,t) - w^2(0,t) \\ &= -(1-(1-R)^2 e^{\int_0^1 2A(\alpha)d\alpha})w^2(0,t).\end{aligned} \tag{4.18}$$

For all for $0 < R \leq 1$.

$$\dot{V} < 0 \tag{4.19}$$

hence, the target is asymptotically stable.

To find the kernel, we start with finding the derivatives of $w(\xi,t)$ using the equation (4.8) with respect to ξ t successively; derivative with respect to ξ,

$$\begin{aligned}\tfrac{\partial w(\xi,t)}{\partial \xi} &= \tfrac{\partial \chi(\xi,t)}{\partial \xi} \quad -k(\xi,\xi)\chi(\xi,t) \\ &\qquad\qquad -\textstyle\int_0^\xi \frac{\partial k(\xi,y)}{\partial \xi}\chi(y,t)dy,\end{aligned} \tag{4.20}$$

and derivative with respect to t,

$$\begin{aligned}\tfrac{\partial w(\xi,t)}{\partial t} &= \chi(\xi,t) - \textstyle\int_0^\xi k(x,y)\frac{\partial \chi(y,t)}{\partial t}dy \\ &= \tfrac{\partial \chi(\xi,t)}{\partial \xi} + \lambda(\xi)\chi(0,t) \\ &- \textstyle\int_0^\xi k(\xi,y)[\frac{\partial \chi(\xi,y)}{\partial y} + \lambda(y)\chi(0,t)]dy \\ &= \tfrac{\partial \chi(\xi,t)}{\partial \xi} + \lambda(\xi)\chi(0,t) - \textstyle\int_0^\xi k(\xi,y)\frac{\chi(y,t)}{\partial y}dy \\ &- \textstyle\int_0^\xi k(\xi,y)\lambda(y)\chi(0,t)dy \\ &= \tfrac{\partial \chi(\xi,t)}{\partial \xi} + \lambda(\xi)\chi(0,t) - k(\xi,\xi)\chi(\xi,t) \\ &+k(\xi,0)\chi(0,t) - \textstyle\int_0^\xi k(\xi,y)\lambda(y)\chi(0,t)dy \\ &+ \textstyle\int_0^\xi \frac{\partial k(\xi,y)}{\partial y}\chi(y,t)dy.\end{aligned} \tag{4.21}$$

Substituting the computed derivatives into equation (4.13) results in

$$\begin{aligned}&[\lambda(\xi) - \textstyle\int_0^\xi k(\xi,y)\lambda(y)dy + k(\xi,0)]\chi(0,t) \\ &+ \textstyle\int_0^\xi [\frac{\partial k(\xi,y)}{\partial y} + \frac{\partial k(\xi,y)}{\partial \xi}]\chi(y,t)dy = 0\end{aligned} \tag{4.22}$$

In order to have the target system satisfied, the following conditions must hold, which in fact ia the kernel system we are seeking for;

$$\frac{\partial k(\xi,y)}{\partial y} + \frac{\partial k(\xi,y)}{\partial \xi} = 0 \tag{4.23}$$

$$k(\xi,0) = \int_0^{\xi} k(\xi,y)\lambda(y)dy - \lambda(\xi). \tag{4.24}$$

Having the kernel system solved, one is able to find the feedback controller law by the equation (4.16).

4.4 Trajectory Tracking

The trajectory we would like to track here is as follow

$$\chi^r(0,t) = Asin(\omega t) + Bcos(\omega t) \tag{4.25}$$

where χ^r is the reference trajectory. The above trajectory is chosen as it can be considered as the representation of most common responses e.g. step, sinusoidal, etc. by Fourier transform. Considering $\xi = 0$ as the output of the system, we seek to generate the trajectory at the input $\xi = 1$, corresponding to equation (4.25). To do so, we are first required to find the full state trajectory $\chi^r(\xi,t)$, which satisfies equations (4.9) and (4.25). We start our investigation to find the state trajectory by using a Taylor series expansion in ξ with time dependent coefficients as suggested by [20];

$$\chi(\xi,t) = \sum_{i=0}^{\infty} a_i(t)\frac{\xi^i}{i!}. \tag{4.26}$$

where we have to find $a_i(t)$ coefficients. Substituting above equation into (4.25), we will find the first coefficient;

$$a_0(t) = \chi^r(0,t) = A.Im\{e^{j\omega t}\} + B.Re\{e^{j\omega t}\}. \tag{4.27}$$

To find the rest of the coefficients, equation (4.25) is substituted in (4.9) as shown below;

$$\sum_{i=0}^{\infty} \dot{a}_i(t)\frac{\xi^i}{i!} = \sum_{i=1}^{\infty} a_i(t)\frac{\xi^{(i-1)}}{(i-1)!} + \lambda(\xi)a_0(t) \tag{4.28}$$

which can be divide into

$$\begin{array}{c} \sum_{i=1}^{\infty} \dot{a}_i(t)\frac{\xi^i}{i!} = \sum_{i=2}^{\infty} a_i(t)\frac{\xi^{(i-1)}}{(i-1)!} \\ \dot{a}_0(t) = a_1(t) + \lambda(t)a_0(t), \end{array} \tag{4.29}$$

that implies

$$\begin{array}{c} \dot{a}_i(t) = a_{i+1}(t) \\ \dot{a}_0(t) = a_1(t) + \lambda(t)a_0(t). \end{array} \tag{4.30}$$

The equation (4.30) leads to the following expression for the coefficients;

$$\begin{array}{rl} a_{k+1}(t) & = A.Im\{(j\omega - \lambda(t))(j\omega)^i e^{j\omega t}\} \\ & +B.Re\{(j\omega - \lambda(t))(j\omega)^i e^{j\omega t}\} \end{array} \tag{4.31}$$

Hence, the state trajectory becomes

$$\begin{aligned}\chi^r(\xi,t) &= A.Im\{e^{j\omega t}\} + B.Re\{e^{j\omega t}\} \\ &+\sum_{k=1}^{\infty}[(A.Im\{(1-\tfrac{g}{j\omega})(jw)^i e^{j\omega t}\} \\ &+B.Re\{(1-\tfrac{g}{j\omega})(j\omega)^i e^{j\omega t}\})\tfrac{\xi^i}{i!}] \\ &= A.Im\{\tfrac{g}{j\omega}e^{j\omega t} + (1-\tfrac{g}{j\omega})e^{j\omega(t+\xi)}\} \\ &+B.Re\{\tfrac{g}{j\omega}e^{j\omega t} + (1-\tfrac{g}{j\omega})e^{j\omega(t+\xi)}\} \\ &= A[sin(\omega(t+\xi)) - \tfrac{g}{\omega}cos(\omega t) + \tfrac{g}{\omega}cos(\omega(t+\xi))] \\ &+B[\tfrac{g}{\omega}sin(\omega t) + cos(\omega(t+\xi)) - \tfrac{g}{\omega}cos(\omega(t+\xi))]\end{aligned} \tag{4.32}$$

which provides us with the reference input as;

$$\begin{aligned}\chi^r(1,t) &= A[sin(\omega(t+1)) - \tfrac{g}{\omega}cos(\omega t) + \tfrac{g}{\omega}cos(\omega(t+1))] \\ &+B[\tfrac{g}{\omega}sin(\omega t) + cos(\omega(t+1)) - \tfrac{g}{\omega}cos(\omega(t+1))].\end{aligned} \tag{4.33}$$

Having the input reference obtained, one can utilize the controller developed earlier to track the trajectory and stabilizes the plant (4.9)-(4.10). This is done as follow; first, the error variable is introduced;

$$\tilde{\chi}(\xi,t) = \chi(\xi,t) - \chi^r(\xi,t) \tag{4.34}$$

from which we come up with the plant in term of error variable

$$\frac{\partial\tilde{\chi}(\xi,t)}{\partial t} = \frac{\partial\tilde{\chi}(\xi,t)}{\partial\xi} + \lambda(\xi)\tilde{\chi}(0,t) \tag{4.35}$$

$$\tilde{\chi}(1,t) = RU(t) + (1-R)e^{\int_0^1 A(\alpha)d\alpha}\tilde{\chi}(0,t). \tag{4.36}$$

The equation (4.36) can be rewritten as;

$$\begin{aligned}\chi(1,t) &= \chi^r(1,t) + RU(t) \\ &+(1-R)e^{\int_0^1 A(\alpha)d\alpha}[\chi(0,t) - \chi^r(0,t)].\end{aligned} \tag{4.37}$$

At last, the developed controller (4.16) in the last section is used to regulate the plant to zero.

$$\begin{aligned}U(t) &= \tfrac{1}{R}\int_0^1 k(1,y)\tilde{\chi}(y,t)dy \\ &= \tfrac{1}{R}\int_0^1 k(1,y)[\chi(y,t) - \chi^r(y,t)]dy\end{aligned} \tag{4.38}$$

The ability of the developed controller to stabilize the system and to track the reference trajectory will be shown in the next section.

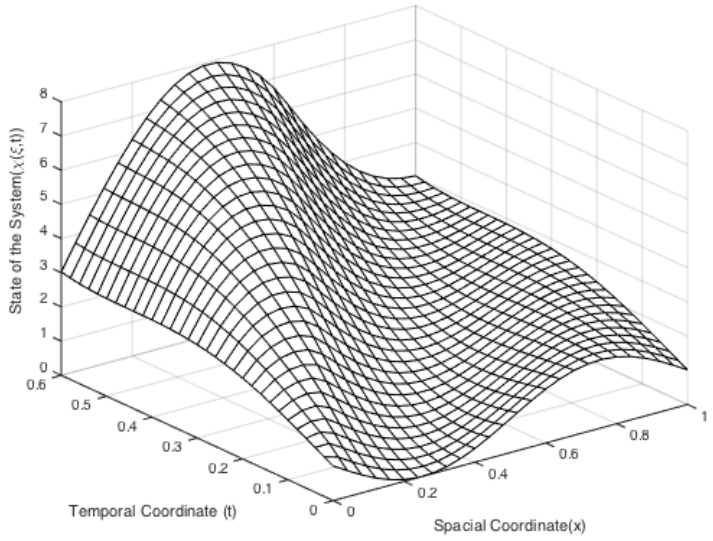

Figure 4.2: Open-loop behavior of system (4.39)-(4.40) when g=1 and b=2.

4.5 Simulation and Results

Consider the following plant

$$\frac{\partial \chi(z,t)}{\partial t} = -\frac{\partial \chi(z,t)}{\partial z} + ge^{b(1-z)}x(1,t) \tag{4.39}$$

$$\chi(0,t) = RU(t) + (1-R)\chi(1,t). \tag{4.40}$$

The open loop behavior of the system, when the $R = 0$ (Full Recycle) is shown in Fig. 4.2. As shown, the system is obviously unstable as it grows with time. Applying the transformation (4.5) leads to

$$\frac{\partial \chi(\xi,t)}{\partial t} = \frac{\partial \chi(\xi,t)}{\partial \xi} + ge^{b\xi}x(0,t) \tag{4.41}$$

$$\chi(1,t) = RU(t) + (1-R)\chi(0,t). \tag{4.42}$$

The kernel for this system, based on (4.23)-(4.24) is

$$\begin{array}{l} \frac{\partial k(\xi,y)}{\partial y} + \frac{\partial k(\xi,y)}{\partial \xi} = 0 \\ k(\xi,0) = \int_0^{\xi} k(\xi,y)ge^{by}dy - ge^{b\xi} \end{array} \tag{4.43}$$

which has the explicit solution [21];

$$k(\xi,y) = -ge^{b+g}(\xi - y) \tag{4.44}$$

Therefore, we end up with the controller below;

$$U(t) = -\frac{1}{R}\int_0^1 ge^{b+g}(1-y)\chi(y,t)dy. \tag{4.45}$$

Using the controller developed, the closed-loop system response for different values of R is acquired. For the case when $R = 1$ system is stabilized sharply, which is usual for backstepping controllers (Fig. 4.3, and Fig. 4.4), while with decreasing the R, the response of the controller, and consequently, the system behavior differ from what is expected.

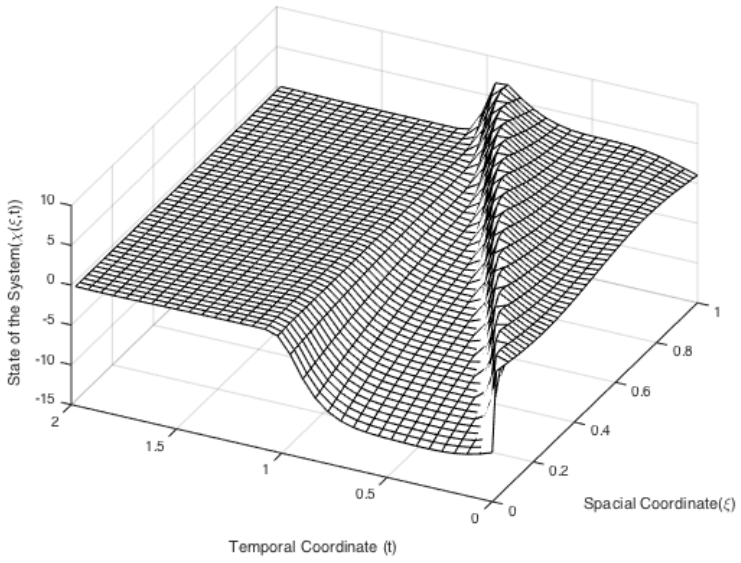

Figure 4.3: Closed-loop behavior of system (4.39)-(4.40) when *R=1*.

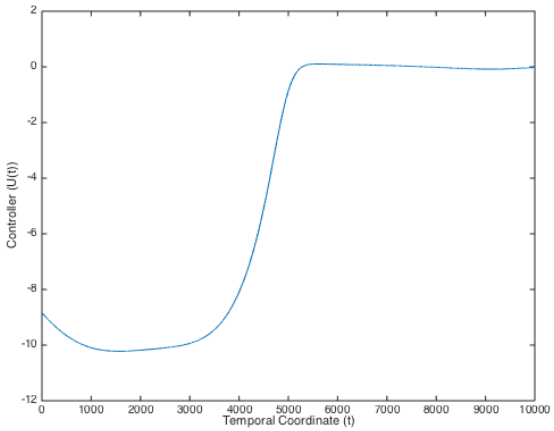

Figure 4.4: Controller progression when *R=1*

For an instance, when *R=0.5*, as shown in Fig. 4.6 the controller oscillates, which postpones the convergence of the system (Fig. 4.5). This behavior is even bolder for the system with *R=0.1*. Fig. 4.7 clearly illustrates such behavior. It can be noticed that how the system and the controller (Fig. 4.8) fluctuate more vigorously, and the extra time the plant needs to be stabilized with increase of the recycle.

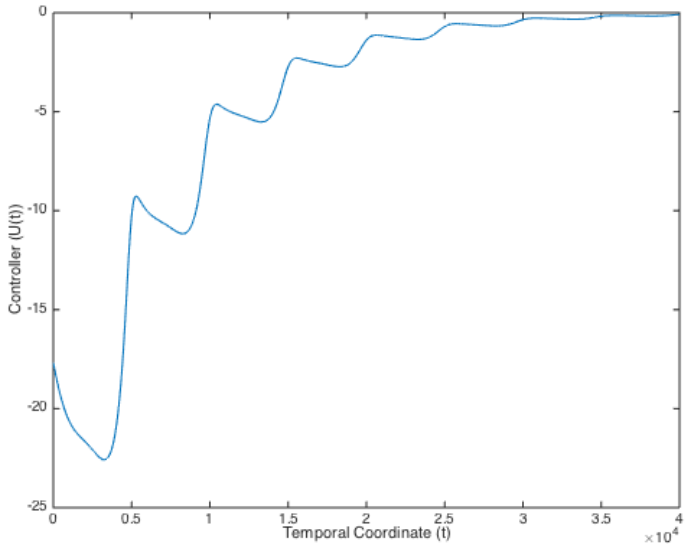

Figure 4.5: Controller progression when *R=0.5*

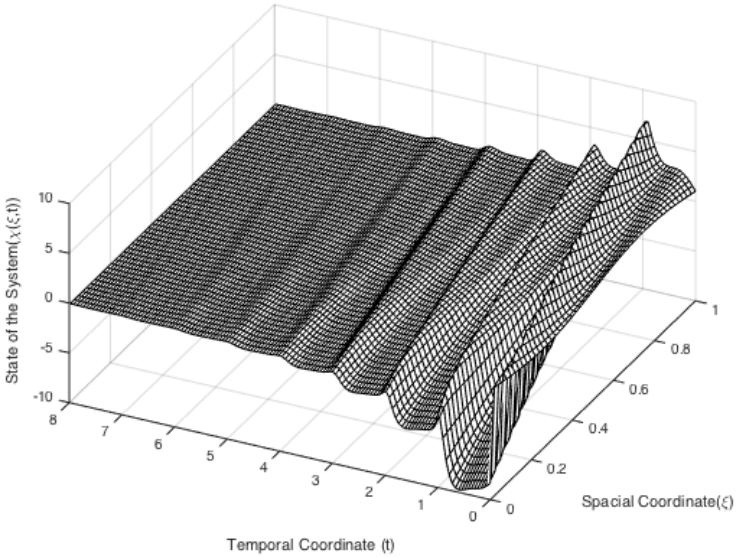

Figure 4.6: Closed-loop behavior of system (4.39)-(4.40) when *R=0.5*.

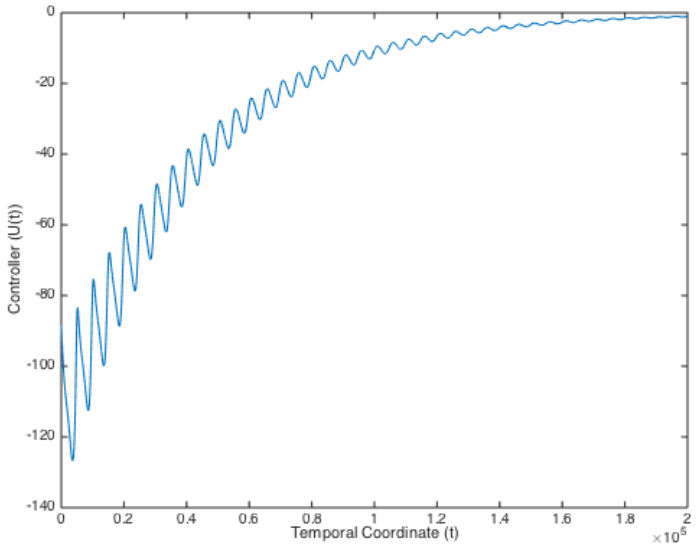

Figure 4.7: Controller progression when *R=0.1*

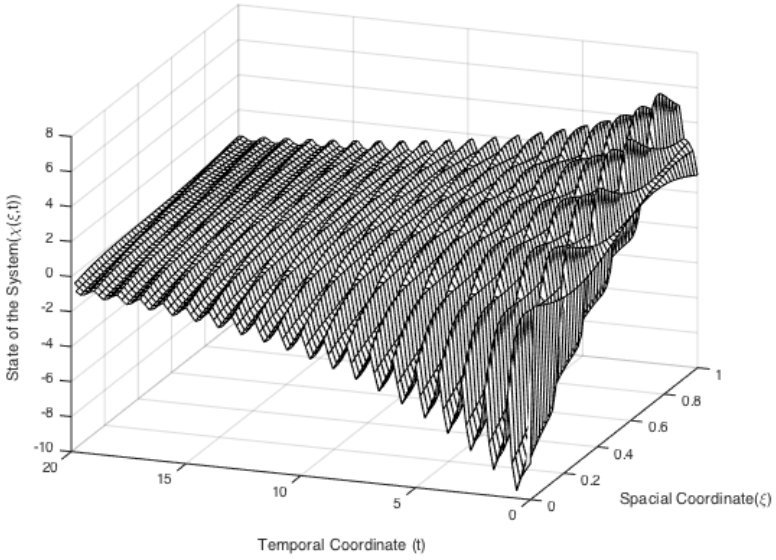

Figure 4.8: Closed-loop behavior of system (4.39)-(4.40) when *R=0.1*.

Seeking to track the reference trajectory for the system (4.39)-(4.40) and following the method used in previous section, Fig. (4.9) achieved as the result. This demonstrates succesful asymptotic tracking of the reference at the input of the system. However, the developed tracking method, not only regulates the system at the input point, it tracks the state trajectory generated throughout the system. This is depicted in Fig. (1.3).

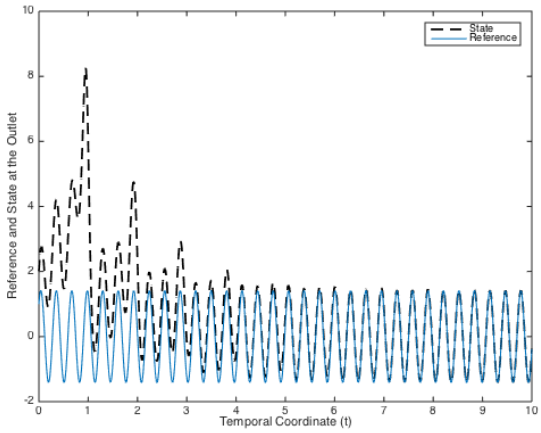

Figure 4.9: Reference tracking of system (4.39)-(4.40) when R=*0.5*.

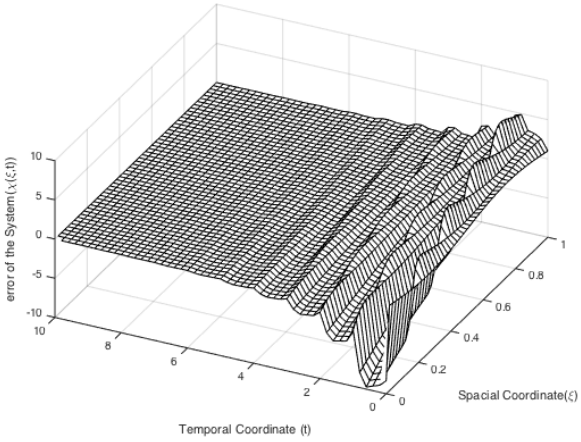

Figure 4.10: State trajectory tracking of the system

At last, this must be mentioned that are the simulations are conducted through coding with MATLAB which can be found in the Appendix.

4.6 Conclusion

A feedback controller designed to asymptotically stabilizes a first order hyperbolic partial differential equation where the boundaries are coupled. This is done by the backstepping method introduced by Smyshlyaev and Krstic (2004) [1]. Moreover, a reference trajectory selected to be tracked by the system at the boundary. A state trajectory is generated from the reference and is regulated by the developed controller. Simulations demonstrate successful control and tracking of the plant by the controller. Results obtained here show an oscillatory response of the system under the controller action due to the recycle associated with the system. This oscillatory behavior aggravates with increase of the recycle rate.

Chapter 5

Summary and Future Work

In this work two common methods for control of first order hyperbolic partial differential equations i.e. Output regulation with internal model control and backstepping were investigated in details. Controllers were developed based on the methods for different systems and their performance investigated through simulations. Finally, the backstepping method was extend to a special case of hyperbolic PDEs when the boundaries are coupled. This case demonstrated intriguing behavior which is illustrated for different recycle ratios. In the future, the coupling of two methods will be studied for two systems and possibility of simultaneous in-domain and boundary control will be inspected.

Bibliography

[1] Ole Morten Aamo and Miroslav Krstic. *Flow Control by Feedback: Stablization and Mixing*. Springer-Verlag London, 2003.

[2] Andrea Serrani Alberto Isidori, Lorenzo Marconi. *Robust Autonomous Guidance*. Advances in Industrial Control. Springer London, 2003.

[3] Neal R. Amundson. *The Mathematical Understanding of Chemical Engineering Systems*. Springer London, 1980.

[4] Karl J. Astrom and Dr. Bjorn Wittenmark. *Adaptive Control*. Addison-Wesley, second edition, 1995.

[5] Itzhak Barkana. Adaptive control? but is so simple! a tribute to the efficiency, simplicity and beauty of adaptive control. *Journal of Intelligent and Robotic Systems*, 2015.

[6] C. I. Bymes, I. G. Lauko, D. S. Gilliam, and V. I. Shubov. Output regulation for linear distributed parameter systems. *IEEE Transactions on Automatic Control*, 45(12):2236–2252, Dec 2000.

[7] David Colton. Constructive and computational methods for differential and integral equations. volume 430 of *Lecture Notes in Mathematics*, pages 95–111. Springer Berlin Heidelberg, 1974.

[8] David Colton. The solution of initial-boundary value problems for parabolic equations by the method of integral operators. *Journal of Differential Equations*, 26(2):181 – 190, 1977.

[9] J. Deutscher. Backstepping design of robust output feedback regulators for boundary controlled parabolic pdes. *IEEE Transactions on Automatic Control*, PP(99):1–1, 2015.

[10] Joachim Deutscher. A backstepping approach to the output regulation of boundary controlled parabolic {PDEs}. *Automatica*, 57:56 – 64, 2015.

[11] Constantin G. Economou and Manfred Morari. Internal model control: multiloop design. *Industrial & Engineering Chemistry Process Design and Development*, 25(2):411–419, 1986.

[12] Constantin G. Economou, Manfred Morari, and Bernhard O. Palsson. Internal model control: extension to nonlinear system. *Industrial & Engineering Chemistry Process Design and Development*, 25(2):403–411, 1986.

[13] B. A. Francis and W. M. Wonham. The internal model principle of control theory. *Automatica*, 12(5):457–465, 1976.

[14] Hagag M Lamin Gabasa. Lqr controller tuning by using particle swarm optimization, 2009.

[15] Carlos E. Garcia and Manfred Morari. Internal model control. a unifying review and some new results. *Industrial & Engineering Chemistry Process Design and Development*, 21(2):308–323, 1982.

[16] Carlos E. Garcia and Manfred Morari. Internal model control. 2. design procedure for multivariable systems. *Industrial & Engineering Chemistry Process Design and Development*, 24(2):472–484, 1985.

[17] Carlos E. Garcia and Manfred Morari. Internal model control. 3. multivariable control law computation and tuning guidelines. *Industrial & Engineering Chemistry Process Design and Development*, 24(2):484–494, 1985.

[18] Ronald M. Hirschorn. lnvertibility of multivariable nonlinear control systems. *IEEE Transactions on Automatic Control*, 24(6):855–865, 1979.

[19] Donald E. Kirk. *Optimal control theory an introduction, Donald*. Prentice Hall, 1970.

[20] Miroslav Krstic. *Boundary Control of PDEs: A Course on Backstepping Designs*. Society for Industrial and Applied Mathematics, 2008.

[21] Miroslav Krstic and Andrey Smyshlyaev. Backstepping boundary control for first-order hyperbolic PDEs and application to systems with actuator and sensor delays. *Systems and Control Letters*, 57(9):750 – 758, 2008.

[22] Dan Luss and Neal R. Amundson. Stability of loop reactors. *AIChE Journal*, 13(2):279–290, 1967.

[23] D. Subbaram Naidu. *Optimal Control Systems*. CRC Press, 2002.

[24] S. H. Oh and R. A. Schmitz. A study of the control of a tubular reactor wih recycle i. theoretical results for a zero-order exothermic reaction. *Chemical Engineering Communications*, 1(4):199–216, 1974.

[25] R. Padhi and S.N. Balakrishnan. Optimal dynamic inversion control design for a class of nonlinear distributed parameter systems with continuous and discrete actuators. *IET Control Theory Appl.*, 1(6):1662?1671, 2007.

[26] Alexey V. Pavlov. *The output regulation problem: a convergent dynamics approach*. PhD thesis, Eindhoven University of Technology, 2004.

[27] M. J. Reilly and R. A. Schmitz. Dynamics of a tubular reactor with recycle: Part i. stability of the steady state. *AIChE Journal*, 12(1):153–161, 1966.

[28] M. J. Reilly and R. A. Schmitz. Dynamics of a tubular reactor with recycle: Part ii. nature of the transient state. *AIChE Journal*, 13(3):519–527, 1967.

[29] Daniel E. Rivera, Manfred Morari, and Sigurd Skogestad. Internal model control: Pid controller design. *Industrial & Engineering Chemistry Process Design and Development*, 25(1):252–265, 1986.

[30] Leonard M. Silverman. Inversion of multivariable linear systems. *IEEE Transactions on Automatic Control*, 14(3):270–276, 1969.

[31] Andrey Smyshlyaev and Miroslav Krstic. Closed-form boundary state feedbacks for a class of 1-d partial integro-differential equations. *IEEE Transactions on Automatic Control*, 49(12):2185 – 2202, 2004.

[32] Andrey Smyshlyaev and Miroslav Krstic. On control design for PDEs with space-dependent diffusivity or time-dependent reactivity. *Automatica*, 41(9):1601 – 1608, 2005.

[33] Andrey Smyshlyaev and Miroslav Krstic. *Adaptive Control of Parabolic PDEs*. Princeton University Press, 2010.

Appendix A

MATLAB Code

The MATLAB codes which are developed for the simulations are presented in this section.

A.1 Codes Developed for Output Regulation

A.1.1 Finite System

This code is developed for control of a finite system with output regulation method in chapter I.

```
clc
clear all
close all
% Internal Model Control Example
% The aim of this code is to simulate a plant where the disturbances and
% reference trajectories are imposed to the system and a full state
% feedback is to be designed to track the trajectory and reject the
% disturbances

%% Parameters Engaged in the Model
% xdot=Abar*x+Bbar*u+Dbar*D
Abar=[0 1; -1 -1];
Bbar=[0 ; 1];
Dbar=[0 ; 1];
% y=C*x
C=[1 0];
% vdot=S*v
S=[0 0 0; 0 0 2; 0 -2 0];
% D=P*v
P=[1 0 0];
% ytracking=Q*v
Q=[0 0 1];
% x=Pi*v
Pi=[0 0 1; 0 -2 0];
% Gamma=K*Pi+E
Gamma=[-1 -2 -3];
% Time Step
Deltat=0.01;
% Termination Time Step
tfinal=10^3;

%% Design Prameter
```

```
% u=K*x+E*v
K=[-1 -1];
% E=Gamma-K*Pi
E=Gamma-K*Pi;

%% Initial Condtion
v0=[0 ; -1 ; 0];
x0=[-1 ; -1];
%x0=Pi*v0;

%% Solving for v (Exosystem)
v=zeros(3,tfinal);
for i=1:tfinal
t=(i-1)*Deltat;
v(:,i)=expm(S*t)*v0;
end

%% Solving for System Responce
x=zeros(2,tfinal);
x(:,1)=x0;
F=zeros(2,tfinal);
for i=2:tfinal
% Euler
F(:,i-1)=(Abar+Bbar*K)*x(:,i-1)+(Bbar*E+Dbar*P)*v(:,i-1);
x(:,i)=x(:,i-1)+F(:,i-1)*Deltat;
% Modified Euler
F(:,i)=(Abar+Bbar*K)*x(:,i)+(Bbar*E+Dbar*P)*v(:,i);
x(:,i)=x(:,i-1)+(F(:,i-1)+F(:,i))*Deltat/2;
end


%% Plotting Section
t=0:Deltat:(tfinal-1)*Deltat;
% Error
figure()
e=C*x-Q*v;
plot(t,e)
xlabel('t','FontSize',16,'FontWeight','bold')
ylabel('e','FontSize',16,'FontWeight','bold')

% Plant State
figure()
plot(t,x(1,:),'r--',t,x(2,:))
xlabel('t','FontSize',16,'FontWeight','bold')
ylabel('x','FontSize',16,'FontWeight','bold')
legend('x_1','x_2')

% Exosystem State
figure()
plot(t,v(1,:),t,v(2,:),t,v(3,:))
xlabel('t','FontSize',16,'FontWeight','bold')
ylabel('v','FontSize',16,'FontWeight','bold')
legend('v_1','v_2','v_3')
% Tracking
figure
plot(t,x(1,:),'r--','LineWidth',2)
hold on
plot(t,v(3,:))
hold off
```

```
xlabel('t','FontSize',16,'FontWeight','bold')
ylabel('v','FontSize',16,'FontWeight','bold')
legend('y','y_{tracking}')

% Control law
figure
u=K*x+E*v;
plot(t,u);
```

A.1.2 Infinite System with In-domain Actuation

In this section, the codes used for simulation of infinite systems in chapter II is presented while the controller is distributed throughout the plant.

Case 1

```
clc
clear all
close all
% Internal Model Control Example
% The aim of this code is to simulate a plant where the disturbances and
% reference trajectories are imposed to the system and a full state
% feedback is to be designed to track the trajectory and reject the
% disturbances

%% Simulation Parameters
% Time Step
Deltat=0.001;
% Interval in Space
Deltax=0.01;
% Termination Time Step
tfinal=10^4;
% spacial cordinate
xi=0:Deltax:1;

%% Parameters Engaged in the Model
% xdot=x_\xi+B*u+D
B=1;
% y=x(1,t)
% vdot=S*v
S=[0 0 0; 0 0 1; 0 -1 0];
% D=H*v
H=[1 0 0];
% ytracking=Q*v
Q=[0 0 1];
% x=Pi*v
pi1=zeros(1,1/Deltax+1);
pi2=-0.5*sin(xi)+0.9152*(1-cos(xi));
pi3=0.5*(1-cos(xi))+0.9152*sin(xi);
Pi=[pi1',pi2',pi3'];
% \Gamma
Gamma=[-1*ones(1/Deltax+1,1) -0.5*ones(1/Deltax+1,1) 0.9152*ones(1/Deltax+1,1)];
%% Design Prameter
% u=K*x+E*v
K=-1;
% L=Gamma-K*Pi
```

```
L=Gamma - K*Pi;

%% Initial Condtion
v0=[1 ; 1 ; 1];
x0=-1*ones(1/Deltax+1,1);
%x0=Pi*v0;

%% Solving for v (Exosystem)
v=zeros(3,tfinal);
for i=1:tfinal
t=(i-1)*Deltat;
v(:,i)=expm(S*t)*v0;
end

%% Solving for System Responce
x=zeros(1/Deltax+1,tfinal);
x(:,1)=x0;
for j=1:tfinal-1
for i=2:1/Deltax+1
x(i,j+1)=(-(x(i,j)-x(i-1,j))/Deltax+B*K*x(i,j)+(B*L(i,:)+H)*v(:,j))*Deltat+x(i,j);
end
end

%% Plotting Section
t=0:Deltat:(tfinal-1)*Deltat;
% Error
figure(1)
e=x(1/Deltax+1,:)-Q*v;
plot(t,e)
xlabel('t','FontSize',16,'FontWeight','bold')
ylabel('e(t)','FontSize',16,'FontWeight','bold')
% Plant State
figure(2)
[m,n]=size(x);
mesh(xi(1:5:m),t(1:100:n),x(1:5:m,1:100:n)')
xlabel('\xi','FontSize',16,'FontWeight','bold')
ylabel('t','FontSize',16,'FontWeight','bold')
zlabel('u(x,t)','FontSize',16,'FontWeight','bold')
colormap(figure(2),gray(1))
view(-64,48)
% Exosystem State
figure(3)
plot(t,v(1,:),t,v(2,:),t,v(3,:))
xlabel('t')
ylabel('v')
title('Exosystem States (x) vs. Time (t)')
legend('v_1','v_2','v_3')
% Tracking
figure(4)
plot(t,x(101,:),'r--','LineWidth',2)
hold on
plot(t,v(3,:))
hold off
xlabel('t','FontSize',16,'FontWeight','bold')
ylabel('y(t)/y_{tr}(t)','FontSize',16,'FontWeight','bold')
legend('y','y_{tracking}')

% Control law
figure(5)
```

```
u=K*x+L*v;
[q,p]=size(u);
mesh(xi(1:5:q),t(1:100:p),u(1:5:q,1:100:p)')
xlabel('t','FontSize',16,'FontWeight','bold')
ylabel('\xi','FontSize',16,'FontWeight','bold')
zlabel('u(\xi,t)','FontSize',16,'FontWeight','bold')
colormap(figure(5),gray(1))
view(-64,48)
```

Case 2

```
clc
clear all
close all
% Internal Model Control Example
% The aim of this code is to simulate a plant where the disturbances and
% reference trajectories are imposed to the system and a full state
% feedback is to be designed to track the trajectory and reject the
% disturbances

%% Simulation Parameters
% Time Step
Deltat=0.001;
% Interval in Space
Deltax=0.01;
% Termination Time Step
tfinal=10^4;
% spacial cordinate
xi=0:Deltax:1;

%% Parameters Engaged in the Model
% xdot=x_\xi+B*u+D
B=1;
% y=x(1,t)
% vdot=S*v
S=[0 0 0; 0 0 1; 0 -1 0];
% D=H*v
H=[0 0 1];
% ytracking=Q*v
Q=[1 0 0];
% x=Pi*v
pi1=xi;
pi2=zeros(1,1/Deltax+1);
pi3=zeros(1,1/Deltax+1);
Pi=[pi1',pi2',pi3'];
% \Gamma
Gamma=[1*ones(1/Deltax+1,1) 0*ones(1/Deltax+1,1) -1*ones(1/Deltax+1,1)];
%% Design Prameter
% u=K*x+E*v
K=-2;
% L=Gamma-K*Pi
L=Gamma - K*Pi;

%% Initial Condtion
v0=[1 ; 1 ; 1];
x0=-1*ones(1/Deltax+1,1);
%x0=Pi*v0;
```

```
%% Solving for v (Exosystem)
v=zeros(3,tfinal);
for i=1:tfinal
t=(i-1)*Deltat;
v(:,i)=expm(S*t)*v0;
end

%% Solving for System Responce
x=zeros(1/Deltax+1,tfinal);
x(:,1)=x0;
for j=1:tfinal-1
for i=2:1/Deltax+1
x(i,j+1)=(-(x(i,j)-x(i-1,j))/Deltax+B*K*x(i,j)+(B*L(i,:)+H)*v(:,j))*Deltat+x(i,j);
end
end

%% Plotting Section
t=0:Deltat:(tfinal-1)*Deltat;
% Error
figure(1)
e=x(1/Deltax+1,:)-Q*v;
plot(t,e)
xlabel('t','FontSize',16,'FontWeight','bold')
ylabel('e(t)','FontSize',16,'FontWeight','bold')
% Plant State
figure(2)
[m,n]=size(x);
mesh(xi(1:5:m),t(1:100:n),x(1:5:m,1:100:n)')
xlabel('\xi')
ylabel('t')
title('Plant States (x) vs. Time (t)')
colormap(figure(2),gray(1))
view(-64,48)
% Exosystem State
figure(3)
plot(t,v(1,:),t,v(2,:),t,v(3,:))
xlabel('t')
ylabel('v')
title('Exosystem States (x) vs. Time (t)')
legend('v_1','v_2','v_3')
% Tracking
figure(4)
plot(t,x(101,:),'r--','LineWidth',2)
hold on
plot(t,v(1,:))
hold off
xlabel('t','FontSize',16,'FontWeight','bold')
ylabel('y(t)/y_{tr}(t)','FontSize',16,'FontWeight','bold')
legend('y','y_{tracking}')
% Control law
figure(5)
u=K*x+L*v;
[q,p]=size(u);
mesh(xi(1:2:q),t([1:75:1000,1000:100:p]),u(1:2:q,[1:75:1000,1000:100:p])')
xlabel('t','FontSize',16,'FontWeight','bold')
ylabel('\xi','FontSize',16,'FontWeight','bold')
zlabel('u(\xi,t)','FontSize',16,'FontWeight','bold')
colormap(figure(5),gray(1))
```

```
view(-64,48)
```

A.1.3 Infinite System with Boundary Actuation

The code presented here corresponds to the system with boundary actuation in Chapter II.

```
clc
clear all
close all
% Internal Model Control Example
% The aim of this code is to simulate a plant where the disturbances and
% reference trajectories are imposed to the system and a full state
% feedback is to be designed to track the trajectory and reject the
% disturbances

%% Simulation Parameters
% Time Step
Deltat=0.001;
% Interval in Space
Deltax=0.01;
% Termination Time Step
tfinal=10^4;
% spacial cordinate
xi=0:Deltax:1;
%% Parameters Engaged in the Model
% xdot=x_\xi+B*u+D
B=1;
% y=x(1,t)
% vdot=S*v
S=[0 0 0; 0 0 1; 0 -1 0];
% D=H*v
H=[0 0 1];
% ytracking=Q*v
Q=[1 0 0];
% x=Pi*v
Pi=@(z) [1  0.459698 -0.841471;...
0 -0.841471*sin(z)+0.540302*(1-cos(z)) 0.841471*(1-cos(z))+0.540302*sin(z)];
% \Gamma
Gamma=[0 0.841471 0.459698];
%% Design Prameter
% u=K*xe+E*v
% K=[k1 -\Beta]; where \Beta is boundary operator at 0
k1=-1;
% L=Gamma-K*Pi
Pi0=Pi(0);
L= Gamma - [k1*Pi0(1,1)-Pi0(2,1) k1*Pi0(1,2)-Pi0(2,2) k1*Pi0(1,3)-Pi0(2,3)];

%% Initial Condtion
v0=[1 ; 1 ; 1];
x0=-1*ones(1/Deltax+1,1);
%u0=Pi0(1,1)*v0(1)+Pi0(1,2)*v0(2)+Pi0(1,3)*v0(3);
u0=-1;
% x(\xi,t)=P(\xi,t)+B(\xi)u(t), B(\xi) chosen to be one.
P0=x0-u0;
%% Solving for v (Exosystem)
v=zeros(3,tfinal);
for i=1:tfinal
t=(i-1)*Deltat;
```

```
v(:,i)=expm(S*t)*v0;
end

%% Solving for Contorl Law
u=zeros(1,tfinal);
u(1)=u0;
F=zeros(1,tfinal);
for i=2:tfinal
% Euler
F(i-1)=k1*u(i-1)+L*v(:,i-1);
u(i)=u(i-1)+F(i-1)*Deltat;
% Modified Euler
F(i)=-u(i)+L*v(:,i);
u(i)=u(i-1)+(F(i-1)+F(i))*Deltat/2;
end

%% Solving for System Responce
P=zeros(1/Deltax+1,tfinal);
for j=1:tfinal-1
for i=2:1/Deltax+1
P(i,j+1)=(-(P(i,j)-P(i-1,j))/Deltax+u(j)+[-L(1) -L(2) 1-L(3)]*v(:,j))*Deltat+P(i,j);
end
end

%% Post Processing
x=zeros(1/Deltax+1,tfinal);
x(:,1)=x0;
for j=2:tfinal
x(:,j)=P(:,j)+u(j);
end
%% Plotting Section
% Time
t=0:Deltat:(tfinal-1)*Deltat;
% Error
figure(1)
e=x(1/Deltax+1,:)-Q*v;
plot(t,e)
xlabel('t','FontSize',16,'FontWeight','bold')
ylabel('e(t)','FontSize',16,'FontWeight','bold')
% Plant State
figure(2)
[m,n]=size(x);
mesh(xi(1:5:m),t(1:100:n),x(1:5:m,1:100:n)')
xlabel('\xi')
ylabel('t')
title('Plant States (x) vs. Time (t)')
colormap(figure(2),gray(1))
view(-64,48)
% Exosystem State
figure(3)
plot(t,v(1,:),'r--',t,v(2,:),'g-.',t,v(3,:))
xlabel('t','FontSize',16,'FontWeight','bold')
ylabel('v(t)','FontSize',16,'FontWeight','bold')
legend('v_1','v_2','v_3')
% Tracking
figure(4)
plot(t,x(101,:),'r--','LineWidth',1.7)
hold on
plot(t,v(1,:))
```

```
hold off
xlabel('t','FontSize',16,'FontWeight','bold')
ylabel('y(t)/y_{tr}(t)','FontSize',16,'FontWeight','bold')
legend('y','y_{tracking}')

% Control law
figure(5)
plot(t,u);
xlabel('t','FontSize',16,'FontWeight','bold')
ylabel('u(t)','FontSize',16,'FontWeight','bold')
```

A.2 Codes Developed for Backstepping

In this section, codes which are developed for backstepping method are presented.

A.2.1 Parabolic System

Here, the codes utilized for parabolic system in Chapter I are presented.

Example 1

Open loop system

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A Project on Backstepping
% Code Developed by "Navid Alavi Sh."
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The goal of this code is to simulate the open loop system behavior
% The plant under consideration is
% u_t=\epsilon(x)*u_{xx}+\lambda*u with the BC u(0,t)=0
% \episilon(x) is found to be
% \epsilon(x) = \epsilon_0*(1+\theta_0*(x-x_0)^2)^2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
clear all

%% Setting up the neccessary parameters
% Parameters of the plant
e0=1;   % Epsilon_0
th0=1;  % Theta_0
x0=1/2; % x_0
lam=20; % Lambda_0
% Parameters used for numerical solution
r=0.1; %r=deltaT/deltaX^2
t=5e4; % Final step in time
int=250; % Number of Intervals between 0 and 1
u(1:1+int,1)=2+sin(2*pi/int.*(1:1+int))-sin(pi/int.*(1:1+int)); % Intial condition
X=0:1/int:1; % y from 0 to 1
deltaX=1/int;
deltaT=r*deltaX^2;

%% Calculation of \epsilon(x)
epss = e0.*(1+th0.*(X-x0).^2).^2;

% %% Caclulation of \bar(x) and \bar(y); the transformed coordinations
% xb=(1+th0*x0^2)/sqrt(th0).*(atan(sqrt(th0)*(1-x0))+atan(sqrt(th0)*x0)); % at x=1; the bounda
```

```
% yb=(1+th0*x0^2)/sqrt(th0).*(atan(sqrt(th0)*(X-x0))+atan(sqrt(th0)*x0));

%% Finding the state
% Explicit rule used to discritized the plant:
% forward in time, central in space

for j=1:t
u(1,j)=0; u(1+int,j)=0;  % BC
for i=2:int
u(i,j+1)=(1-2*epss(i)*r+lam*deltaT)*u(i,j) + (epss(i)*r)*u(i+1,j)...
+(epss(i)*r)*u(i-1,j);
end
end

%% Post processing the Data
% Ploting epsilon
figure('Name','Epsilon(x) vs x','NumberTitle','off')
plot(X,epss)
title('Epsilon(x) vs x')
xlabel('x')
ylabel('epsilon')
% Sclaing the Coordinates
XX=0:deltaX:1; % Spacial coordinate
YY=0:deltaT:t*deltaT; % Time coordiante
%Creating the mesh
figure('Name','u(x,t) vs x and t','NumberTitle','off')
[m,n]=size(u);
p=1; % Resolution of the figure in spatial direction
q=100; % Resolution of the graph in time direction
mesh(XX(1:p:m),YY(1:q:n),u(1:p:m,1:q:n)')
title('u(x,t) vs x and t')
xlabel('x')
ylabel('t')
```

Closed loop

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A Project on Backstepping
% Code Developed by "Navid Alavi Sh."
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The goal of this code is to simulate the closed loop system behavior
% The plant under consideration is
% u_t=\epsilon(x)*u_{xx}+\lambda*u with the BC u(0,t)=0
% \episilon(x) is found to be
% \epsilon(x) = \epsilon_0*(1+\theta_0*(x-x_0)^2)^2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
clear all

%% Setting up the neccessary parameters
% Parameters of the plant
e0=1;   % epsilon_0
th0=1;  % Theta_0 % It cannot be zero
x0=1/2; % x_0
lam=20; % lambda
% Parameters used for numerical solution
r=0.1; %r=deltaT/deltaX^2
```

```
t=1e5; % Final step in time
int=250; % Number of Intervals between 0 and 1
u=zeros(1+int,t+1);
u(1:1+int,1)=2+sin(2*pi/int.*(1:1+int))-sin(pi/int.*(1:1+int)); % Intial condition
X=0:1/int:1; % y from 0 to 1
deltaX=1/int;
deltaT=r*deltaX^2;

%% Calculation of \epsilon(x) and c
epss = e0.*(1+th0.*(X-x0).^2).^2;
% c >= \epsilon_{xx}/2
c=max(4e0*th0*(1+3*th0*(X-x0).^2))+1;

%% Caclulation of \bar(x) and \bar(y); the transformed coordinations
xb=(1+th0*x0^2)/sqrt(th0).*(atan(sqrt(th0)*(1-x0))+atan(sqrt(th0)*x0)); % at x=1; the boundary
yb=(1+th0*x0^2)/sqrt(th0).*(atan(sqrt(th0)*(X-x0))+atan(sqrt(th0)*x0));

%% Calculation of the Kernel at x=1
k=-yb.*(lam+c)./sqrt(epss(1)).*epss(1+int).^(1/4)./epss.^(3/4).*...
besseli(1,sqrt((lam+c)/epss(1).*(xb^2-yb.^2)+eps))./...
sqrt((lam+c)/epss(1).*(xb^2-yb.^2)+eps);

%% Finding the state
% Explicit rule used to discritized the plant:
% forward in time, central in space
for j=1:t
if j~=1
u(1+int,j)=u(1+int,j-1);
end
integrant=k.*u(:,j)'; % Integrant of controller Eq.
u(1+int,j)=trapz(X,integrant); % Boundary conditon at u(1,t);
u(1,j)=0;  % BC at x=0
for i=2:int
u(i,j+1)=(1-2*epss(i)*r+lam*deltaT)*u(i,j) + (epss(i)*r)*u(i+1,j)...
+(epss(i)*r)*u(i-1,j);
end
end

integrant=k.*u(:,j)'; % Integrant of controller Eq.
u(1+int,t+1)=trapz(X,integrant); % Boundary conditon at u(1,t);
u(1,t+1)=0;  % BC at x=0

%% Post processing the Data
% Ploting epsilon
figure('Name','Epsilon(x)','NumberTitle','off')
plot(X,epss)
title('Epsilon(x)')
xlabel('x')
ylabel('epsilon')
% Sclaing the Coordinates
XX=0:deltaX:1; % Spacial coordinate
YY=0:deltaT:t*deltaT; % Time coordiante
%Creating the mesh
figure('Name','State vs. time','NumberTitle','off')
[m,n]=size(u);
p=1; % Resolution of the figure in spatial direction
q=100; % Resolution of the graph in time direction
mesh(XX(1:p:m),YY(1:q:n),u(1:p:m,1:q:n)')
title('u(x,t)')
```

```
xlabel('x')
ylabel('t')
% Ploting the Boundary
figure('Name','Controller vs. time','NumberTitle','off')
plot(1:t+1,u(1+int,:))
title('u(x,t) at x=1')
xlabel('t')
ylabel('u(1,t)')
% Ploting the Kernel
figure('Name','The Kernel at x=1','NumberTitle','off')
plot(X,k)
title('k(1,y)')
xlabel('y')
ylabel('k')
```

Example 2

Open loop system

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A Project on Backstepping
% Code Developed by "Navid Alavi Sh."
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The goal of this code is to simulate the open loop system behavior
% The plant under consideration is
% u_t=d/dx(\epsilon(x)*d/dx(u))+\lambda*u with the BC u(0,t)=0
% \episilon(x) is found to be
% \epsilon(x) = \epsilon_0*(x-x_0)^2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
clear all

%% Setting up the neccessary parameters
% Parameters of the plant
e0=1;   % Epsilon_0
x0=3/2; % x_0
lam=20; % Lambda_0
% Parameters used for numerical solution
r=0.1; %r=deltaT/deltaX^2
t=5e4; % Final step in time
int=250; % Number of Intervals between 0 and 1
u(1:1+int,1)=15*sin(2*pi/int.*(1:1+int)); % Intial condition
X=0:1/int:1; % y from 0 to 1
deltaX=1/int;
deltaT=r*deltaX^2;

%% Calculation of \epsilon(x)
epss = e0.*(X-x0).^2;
epss_x=2*e0.*(X-x0);

% %% Caclulation of \bar(x) and \bar(y); the transformed coordinations
% xb=(1+th0*x0^2)/sqrt(th0).*(atan(sqrt(th0)*(1-x0))+atan(sqrt(th0)*x0)); % at x=1; the bounda
% yb=(1+th0*x0^2)/sqrt(th0).*(atan(sqrt(th0)*(X-x0))+atan(sqrt(th0)*x0));

%% Finding the state
% Explicit rule used to discritized the plant:
% forward in time, central in space
```

```
for j=1:t
u(1,j)=0; u(1+int,j)=0;  % BC
for i=2:int
u(i,j+1)=(1-2*epss(i)*r+lam*deltaT)*u(i,j)...
+(epss(i)*r+epss_x(i)/2*r*deltaX)*u(i+1,j)...
+(epss(i)*r-epss_x(i)/2*r*deltaX)*u(i-1,j);
end
end

%% Post processing the Data
% Ploting epsilon
figure('Name','Epsilon(x) vs x','NumberTitle','off')
plot(X,epss)
title('Epsilon(x) vs x')
xlabel('x')
ylabel('epsilon')
% Sclaing the Coordinates
XX=0:deltaX:1; % Spacial coordinate
YY=0:deltaT:t*deltaT; % Time coordiante
%Creating the mesh
figure('Name','u(x,t) vs x and t','NumberTitle','off')
[m,n]=size(u);
p=1; % Resolution of the figure in spatial direction
q=100; % Resolution of the graph in time direction
mesh(XX(1:p:m),YY(1:q:n),u(1:p:m,1:q:n)')
title('u(x,t) vs x and t')
xlabel('x')
ylabel('t')
```

Closed loop system

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A Project on Backstepping
% Code Developed by "Navid Alavi Sh."
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The goal of this code is to simulate the closed loop system behavior
% The plant under consideration is
% u_t=d/dx(\epsilon(x)*d/dx(u))+\lambda*u with the BC u(0,t)=0
% \episilon(x) is found to be
% \epsilon(x) = \epsilon_0*(x-x_0)^2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
clear all

%% Setting up the neccessary parameters
% Parameters of the plant
e0=1/2;   % epsilon_0
x0=-3/2; % x_0
lam=20; % lambda
% Parameters used for numerical solution
r=0.1; %r=deltaT/deltaX^2
t=5e4; % Final step in time
int=250; % Number of Intervals between 0 and 1
u=zeros(1+int,t+1);
u(1:1+int,1)=1+sin(2*pi/int.*(1:1+int))-sin(pi/int.*(1:1+int)); % Intial condition

X=0:1/int:1; % y from 0 to 1
```

```
deltaX=1/int;
deltaT=r*deltaX^2;

%% Calculation of \epsilon(x) and c
epss = e0.*(X-x0).^2;
epss_x=2*e0.*(X-x0);
% c >= \epsilon_{xx}/2
c=max(2*e0)+1;

%% Caclulation of \bar(x) and \bar(y); the transformed coordinations
xb=-x0.*log(1-1/x0); % at x=1; the boundary
yb=-x0.*log(1-X./x0);

%% Calculation of the Kernel at x=1
k=-yb.*(lam+c)./sqrt(epss(1)).*epss(1+int).^(3/4)./epss.^(5/4).*...
besseli(1,sqrt((lam+c)/epss(1).*(xb^2-yb.^2)+eps))./...
sqrt((lam+c)/epss(1).*(xb^2-yb.^2)+eps);

%% Finding the state
% Explicit rule used to discritized the plant:
% forward in time, central in space
for j=1:t
if j~=1
u(1+int,j)=u(1+int,j-1);
end
integrant=k.*u(:,j)'; % Integrant of controller Eq.
u(1+int,j)=trapz(X,integrant); % Boundary conditon at u(1,t);
u(1,j)=0;  % BC at x=0
for i=2:int
u(i,j+1)=(1-2*epss(i)*r+lam*deltaT)*u(i,j)...
+(epss(i)*r+epss_x(i)/2*r*deltaX)*u(i+1,j)...
+(epss(i)*r-epss_x(i)/2*r*deltaX)*u(i-1,j);
end
end

integrant=k.*u(:,j)'; % Integrant of controller Eq.
u(1+int,t+1)=trapz(X,integrant); % Boundary conditon at u(1,t);
u(1,t+1)=0;  % BC at x=0

%% Post processing the Data
% Ploting epsilon
figure('Name','Epsilon(x)','NumberTitle','off')
plot(X,epss)
title('Epsilon(x)')
xlabel('x')
ylabel('epsilon')
% Sclaing the Coordinates
XX=0:deltaX:1; % Spacial coordinate
YY=0:deltaT:t*deltaT; % Time coordiante
%Creating the mesh
figure('Name','State vs. time','NumberTitle','off')
[m,n]=size(u);
p=1; % Resolution of the figure in spatial direction
q=100; % Resolution of the graph in time direction
mesh(XX(1:p:m),YY(1:q:n),u(1:p:m,1:q:n)')
title('u(x,t)')
xlabel('x')
ylabel('t')
% Ploting the Boundary
```

```
figure('Name','Controller vs. time','NumberTitle','off')
plot(1:t+1,u(1+int,:))
title('u(x,t) at x=1')
xlabel('t')
ylabel('u(1,t)')
% Ploting the Kernel
figure('Name','The Kernel at x=1','NumberTitle','off')
plot(X,k)
title('k(1,y)')
xlabel('y')
ylabel('k')
```

A.2.2 Generic Hyperbolic System

Here the codes developed for the systems in Chapter III are presented.

First Example

Open loop System

```
clc
clear all
% PDE: u_t=u_x+gexp((b+g)(x-y))
%% Setting the necessary parameters
g=1;
b=2;
r=0.1; %r=deltaT/deltaX^2
t=9*10^3; % Final step in time
int=500; % Number of Intervals between 0 and 1
u(1:1+int,1)=15*sin(2*pi/int.*(1:1+int)); % Intial condition
Y=0:1/int:1; % y from 0 to 1

%% Solving the system by a cool finite difference scheme
% Forward in time (Explicit)
% Forward in Space
for j=1:t %step in time
u(1+int,j)=0; % B.C
for i=1:int %nodes in space
u(i,j+1)=r*u(i+1,j)+(1-r)*u(i,j)+r/int*g*exp(b*(i-1)/int)*u(1,j);
end
end
%% Post processing the Data
%reversing the x cordinate
%u2=flipud(u);
% Sclaing the Coordinates
XX=0:1/int:1; % Spacial coordinate
YY=0:r/int:t*r/int; % Time coordiante
%Creating the mesh
figure(1)
[m,n]=size(u);
mesh(XX(1:5:m),YY(1:500:n),u(1:5:m,1:500:n)')
xlabel('x')
ylabel('t')
zlabel('u(x,t)')
colormap(gcf,gray(1))
view(-64,48)
%figure(2)
```

```
%mesh(XX(1:5:m),YY(1:5:n),u2(1:5:m,1:5:n)')
% figure(3)
% plot(1:t+1,u(1+int/2,:))
% figure(4)
% plot(1:int+1,u(:,1+t))
```

Closed loop System

```
clc
clear all
% PDE: u_t=u_x-gexp((b+g)(x-y))
%% Setting up the required parameters
g=1;
b=2;
r=0.1; %r=deltaT/deltaX
t=1*10^4; % Final step in time
int=500; % Number of Intervals between 0 and 1
u(1:1+int,1)=abs(1+sin(2*pi/int.*(1:1+int))-cos(4/5*pi/int.*(1:1+int))); % Intial condition
Y=0:1/int:1; % y from 0 to 1

%% Solving the system
for j=1:t %step in time
if j~=1
u(1+int,j)=u(1+int,j-1);
end
integrant=-g*exp((b+g).*(1-Y)).*u(:,j)'; % Integrant of controller Eq.
u(1+int,j)=trapz(Y,integrant); % Boundary conditon at u(1,t);
for i=1:int %nodes in space
u(i,j+1)=r*u(i+1,j)+(1-r)*u(i,j)+r/int*g*exp(b*(i-1)/int)*u(1,j);
end
end

integrant=-g*exp((b+g)./(1-Y)).*u(:,j)'; % Integrant of controller Eq.
u(1+int,t+1)=trapz(Y,integrant);

%% Postprocessing
% Sclaing the Coordinates
XX=0:1/int:1; % Spacial coordinate
YY=0:r/int:t*r/int; % Time coordiante
%Creating the mesh
figure(1)
[m,n]=size(u);
mesh(XX(1:10:m),YY(1:150:n),u(1:10:m,1:150:n)')
xlabel('x')
ylabel('t')
zlabel('u(x,t)')
colormap(gcf,gray(1))
view(-64,48)
% Ploting the Boundary
figure(2)
plot(1:t+1,u(1+int,:))
% figure(3)
% plot(1:t+1,u(1+int/2,:))
% figure(4)
% plot(1:int+1,u(:,1+t))
```

Second Example

Open loop System

```
clc
clear all
% PDE: u_t = u_x + Int_0^x f*exp(lambda(x-y))u(y,t)dy
f=2;
lambda=5;
r=0.1; %r=deltaT/deltaX^2
t=5e3; % Final step in time
int=300; % Number of Intervals between 0 and 1
u(1:1+int,1)=15*sin(2*pi/int.*(1:1+int)); % Intial condition
Y=0:1/int:1; % y from 0 to 1
deltaX=1/int;
deltaT=r/int;


for j=1:t %step in time
u(1+int,j)=0; % B.C
for i=1:int %nodes in space
R=0;
if i==1
u(i,j+1)=r*u(i+1,j)+(1-r)*u(i,j);
else
for k=1:i-1
R=R + deltaX/2*(exp(-lambda*(k)*deltaX)*u(k+1,j)+exp(-lambda*(k-1)*deltaX)*u(k,j));
end
u(i,j+1)=r*u(i+1,j)+(1-r)*u(i,j)+f*deltaT*exp(lambda*(i-1)*deltaX)*R;
end
end
end
%% Post processing the Data
% Sclaing the Coordinates
XX=0:1/int:1; % Spacial coordinate
YY=0:r/int:t*r/int; % Time coordiante
%Creating the mesh
figure(1)
[m,n]=size(u);
mesh(XX(1:10:m),YY(1:100:n),u(1:10:m,1:100:n)')
xlabel('x')
ylabel('t')
zlabel('u(x,t)')
colormap(gcf,gray(1))
%view(-64,48)
```

Closed loop System

```
clc
clear all
% PDE: u_t = u_x + Int_0^x f*exp(lambda(x-y))u(y,t)dy
f=2;
lambda=5;
r=0.1; %r=deltaT/deltaX
t=10e3; % Final step in time
int=300; % Number of Intervals between 0 and 1
u(1:1+int,1)=15*sin(2*pi/int.*(1:1+int)); % Intial condition
Y=0:1/int:1; % y from 0 to 1
deltaX=1/int;
```

```
deltaT=r/int;


for j=1:t %step in time
if j~=1
u(1+int,j)=u(1+int,j-1);
end
integrant=-exp(lambda.*(1-Y)).*Y.*f.*(besseli(1,2.*sqrt(f.*(1-Y)))+eps)./(sqrt(f.*(1-Y))+eps).
u(1+int,j)=trapz(Y,integrant); % Boundary conditon at u(1,t);
for i=1:int %nodes in space
R=0;
if i==1
u(i,j+1)=r*u(i+1,j)+(1-r)*u(i,j);
else
for k=1:i-1
R=R + deltaX/2*(exp(-lambda*(k)*deltaX)*u(k+1,j)+exp(-lambda*(k-1)*deltaX)*u(k,j));
end
u(i,j+1)=r*u(i+1,j)+(1-r)*u(i,j)+f*deltaT*exp(lambda*(i-1)*deltaX)*R;
%               Y2=0:deltaX:(i-1)*deltaX;
%               int2= exp(-Y2.*lambda).*u(1:length(Y2),j)';
%               integral=trapz(Y2,int2);
%               u(i,j+1)=r*u(i+1,j)+(1-r)*u(i,j)+f*deltaT*exp(lambda*(i-1)*deltaX)*integral;
end
end
end

integrant=-exp(lambda.*(1-Y)).*Y.*f.*(besseli(1,2.*sqrt(f.*(1-Y)))+eps)./(sqrt(f.*(1-Y))+eps).
u(1+int,t+1)=trapz(Y,integrant);
%% Post processing the Data
% Sclaing the Coordinates
XX=0:1/int:1; % Spacial coordinate
YY=0:r/int:t*r/int; % Time coordiante
%Creating the mesh
figure(1)
[m,n]=size(u);
mesh(XX(1:10:m),YY(1:100:n),u(1:10:m,1:100:n)')
xlabel('x')
ylabel('t')
zlabel('u(x,t)')
colormap(gcf,gray(1))
view(-64,48)
figure(2)
plot(1:t+1,u(1+int,:))
```

A.2.3 Special Hyperbolic System

The codes developed for systems in Chapter IV are presented here.

Open loop system

```
clc
clear all
% PDE: u_t=u_x+gexp((b+g)(x-y))
%% Setting the necessary parameters
g=1;
b=2;
r=0.1; %r=deltaT/deltaX^2
t=3*10^3; % Final step in time
```

```
int=500; % Number of Intervals between 0 and 1
u(1:1+int,1)=abs(1+sin(2*pi/int.*(1:1+int))); % Intial condition
Y=0:1/int:1; % y from 0 to 1

%% Solving the system by a cool finite difference scheme
% Forward in time (Explicit)
% Forward in Space
for j=1:t %step in time
    u(1+int,j)=u(1,j); % B.C
    for i=1:int %nodes in space
        u(i,j+1)=r*u(i+1,j)+(1-r)*u(i,j)+r/int*g*exp(b*(i-1)/int)*u(1,j);
    end
end
u(1+int,t+1)=u(1,t+1);
%% Post processing the Data
%reversing the x cordinate
%u2=flipud(u);
% Sclaing the Coordinates
XX=0:1/int:1; % Spacial coordinate
YY=0:r/int:t*r/int; % Time coordiante
%Creating the mesh
figure(1)
[m,n]=size(u);
%mesh(XX(1:10:m),YY(1:10:n),u(1:10:m,1:10:n)')
mesh(XX(1:20:m),YY(1:100:n),u(m:-20:1,1:100:n)')
xlabel('Spacial Coordinate(x)')
ylabel('Temporal Coordinate (t)')
zlabel('State of the System(\chi(\xi,t))')
colormap(gray(1))
%figure(2)
%mesh(XX(1:5:m),YY(1:5:n),u2(1:5:m,1:5:n)')
% figure(3)
% plot(1:t+1,u(1+int/2,:))
% figure(4)
% plot(1:int+1,u(:,1+t))
```

closed loop system

```
clc
clear all
% PDE: u_t=u_x+gexp((b+g)(x-y))
%% Setting up the required parameters
R=1;
g=1;
b=2;
r=0.1; %r=deltaT/deltaX
t=3*10^4; % Final step in time
int=500; % Number of Intervals between 0 and 1
u(1:1+int,1)=abs(1+sin(2*pi/int.*(1:1+int))); % Intial condition
Y=0:1/int:1; % y from 0 to 1

%% Solving the system
for j=1:t %step in time
    if j~=1
    u(1+int,j)=u(1+int,j-1);
    end
    integrant=-g*exp((b+g).*(1-Y)).*u(:,j)'; % Integrant of controller Eq.
```

```
    U(j)=1/R*trapz(Y,integrant); % Boundary conditon at u(1,t);
    u(1+int,j)=R*U(j)+(1-R)*u(1,j);
    for i=1:int %nodes in space
        u(i,j+1)=r*u(i+1,j)+(1-r)*u(i,j)+r/int*g*exp(b*(i-1)/int)*u(1,j);
    end
end

integrant=-g*exp((b+g)./(1-Y)).*u(:,j)'; % Integrant of controller Eq.
U(t+1)=1/R*trapz(Y,integrant);
u(1+int,1+t)=R*U(t+1)+(1-R)*u(1,j);

%% Postprocessing
% Sclaing the Coordinates
XX=0:1/int:1; % Spacial coordinate
YY=0:r/int:t*r/int; % Time coordiante
%Creating the mesh
figure(1)
[m,n]=size(u);
mesh(XX(1:10:m),YY(1:1000:n),u(m:-10:1,1:1000:n)')
xlabel('Spacial Coordinate(\xi)')
ylabel('Temporal Coordinate (t)')
zlabel('State of the System(\chi(\xi,t))')
colormap(figure(1),gray(1))
view(-64,48)

% Ploting the Boundary
%figure(2)
%plot(1:t+1,u(1+int,:))

% Ploting the Controller
figure(3)
plot(1:t+1,U);
xlabel('Temporal Coordinate (t)')
ylabel('Controller (U(t))')



% figure(3)
% plot(1:t+1,u(1+int/2,:))
% figure(4)
% plot(1:int+1,u(:,1+t))
```

Trajectory tracking

```
clc
clear all
% Here the following pde is in fact the error pde; u is equivalant to
% u_tilda where u_tilda=u-u_r
% PDE: u_t=u_x+gexp((b+g)(x-y))
% Trajectory to be tracked: u_r(0,t)=Asin(wt)+Bcos(wt)

%% Setting up the required parameters
R=0.5;
g=1;
b=2;
r=0.1; %r=deltaT/deltaX
t=5*10^4; % Final step in time
```

```
int=500; % Number of Intervals between 0 and 1
u(1:1+int,1)=abs(1+sin(2*pi/int.*(1:1+int))); % Intial condition
Y=0:1/int:1; % y from 0 to 1

% Trajectory Prameters
w=20;
A=1;
B=1;



%% Solving the system
U=zeros(1,1+t);
for j=1:t %step in time
    if j~=1
    u(1+int,j)=u(1+int,j-1);
    end
    integrant=-g*exp((b+g).*(1-Y)).*u(:,j)'; % Integrant of controller Eq.
    U(j)=1/R*trapz(Y,integrant); % Boundary conditon at u(1,t);
    u(1+int,j)=R*U(j)+(1-R)*u(1,j);

    for i=1:int %nodes in space
        u(i,j+1)=r*u(i+1,j)+(1-r)*u(i,j)+r/int*g*exp(b*(i-1)/int)*u(1,j);
    end
end

integrant=-g*exp((b+g)./(1-Y)).*u(:,j)'; % Integrant of controller Eq.
U(t+1)=1/R*trapz(Y,integrant);
u(1+int,1+t)=R*U(t+1)+(1-R)*u(1,j);

%% Trajectory Tracking
% Trajectoy
for j=1:t+1
    T=(j-1)*r/int;
    u_r(:,j)=A*(-g/w*cos(w.*T)+sin(w.*(T+Y))+g/w*cos(w.*(Y+T)))...
    +B*(g/w*sin(w.*T)+cos(w.*(T+Y))-g/w*sin(w.*(T+Y)));
end

% Responce of system when tracking rules applied at x=0
for j=1:t+1
    integrant_T=-g*exp((b+g).*(1-Y)).*(u(:,j)-u_r(:,j))'; % Integrant of controller Eq.
    U_T(j)=1/R*trapz(Y,integrant); % Boundary conditon at u(1,t);
    u_T(j)=R*U(j)+(1-R)*(u(1,j)-u_r(1,j))+u_r(1);
end

%% Postprocessing
% Sclaing the Coordinates
XX=0:1/int:1; % Spacial coordinate
YY=0:r/int:t*r/int; % Time coordiante

% Error of system
figure(1)
[m,n]=size(u);
mesh(XX(1:10:m),YY(1:1000:n),u(m:-10:1,1:1000:n)')
xlabel('Spacial Coordinate(\xi)')
ylabel('Temporal Coordinate (t)')
zlabel('error of the System(\chi(\xi,t))')
colormap(figure(1),gray(1))
view(-64,48)
```

```
% The State
figure(2)
[m,n]=size(u);
mesh(XX(1:10:m),YY(1:1000:n),u(m:-10:1,1:1000:n)'+u_r(m:-10:1,1:1000:n)')
xlabel('Spacial Coordinate(\xi)')
ylabel('Temporal Coordinate (t)')
zlabel('State of the System(\chi(\xi,t))')
colormap(figure(2),gray(1))
view(-64,48)


% Ploting the Boundary at x=0
figure(3)
uu=u+u_r; % The state
plot(YY,uu(1,:),'--k','LineWidth',1.7)
hold on
plot(YY,u_r(1,:))
hold off
xlabel('Temporal Coordinate (t)')
ylabel('Reference and State at the Outlet')
legend('State','Reference')

% Ploting state and error at the boundary at x=1
% Or ploting input of the original system and error system
figure(4)
plot(YY,u(1+int,:));
hold on
plot(YY,u_T);
hold off
xlabel('Temporal Coordinate (t)')
ylabel('Input of System')
legend('Error','State','Location','northwest')
```