CONTROL OF LINEAR TIME-INVARIENT DISTRIBUTED PARAMETER SYSTEMS: FROM INTEGER ORDER TO FRACTIONAL ORDER

by

Jinsong Liang

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

MASTER OF SCIENCE

in

Electrical Engineering

Approved:

Dr. YangQuan Chen
Major Professor

Dr. Rees Fullmer
Committee Member

Dr. Wei Ren
Committee Member

UTAH STATE UNIVERSITY
Logan, Utah
2005
Acknowledgments

I would like to express my deepest gratitude to my major advisor, Dr. YangQuan Chen for his excellent guidance, time, and participation, without which the completion of this thesis would not be possible.

I would also like to thank my wife, Ye, for her encouragement, care, and support at all times.

Jinsong Liang
Abstract

Control of Linear Time-Invariant Distributed Parameter Systems: From Integer Order to Fractional Order

by

Jinsong Liang, Master of Science
Utah State University, 2005

Major Professor: Dr. YangQuan Chen
Department: Electrical and Computer Engineering

Control of distributed parameter systems has been an active research area. This thesis presents a few research results in the past three years on different, but related, topics. The studied topics include: a new simulation method for control of distributed parameter systems, control of fractional order wave equations using both integer order and fractional order boundary controllers, control of fractional order wave equations and beam equations subject to large delays, and the development of a simulation platform for simulating control of diffusion process using mobile actuators and sensors.

(119 pages)
# Contents

<table>
<thead>
<tr>
<th>Acknowledgments</th>
<th>iii</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iv</td>
</tr>
<tr>
<td>List of Tables</td>
<td>vii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>viii</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2 A Hybrid Symbolic-Numerical Simulation Method</td>
<td>2</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>2</td>
</tr>
<tr>
<td>2.2 Problem Formulation, Principle, and Implementation</td>
<td>2</td>
</tr>
<tr>
<td>3 Boundary Control of the Wave Equations Subject to Large Delays</td>
<td>10</td>
</tr>
<tr>
<td>3.1 A Brief Introduction to The Smith Predictor</td>
<td>10</td>
</tr>
<tr>
<td>3.2 Problem Formulation</td>
<td>12</td>
</tr>
<tr>
<td>3.3 Simulation Results</td>
<td>13</td>
</tr>
<tr>
<td>3.4 The Damped Wave Equation Case and the Robustness Issue</td>
<td>18</td>
</tr>
<tr>
<td>3.4.1 Boundary Control of the Damped Wave Equation</td>
<td>18</td>
</tr>
<tr>
<td>3.4.2 Stability and Robustness Analysis</td>
<td>19</td>
</tr>
<tr>
<td>4 Boundary Control of the Beam Equation with Large Delays</td>
<td>23</td>
</tr>
<tr>
<td>4.1 Problem Formulation</td>
<td>23</td>
</tr>
<tr>
<td>4.2 Simulation Results and Analysis</td>
<td>25</td>
</tr>
<tr>
<td>5 Diff-MAS2D: a Simulation Platform for Measurement and Actuation Scheduling in Diffusion Process with Mobile Actuators and Sensors</td>
<td>33</td>
</tr>
<tr>
<td>5.1 Background</td>
<td>33</td>
</tr>
<tr>
<td>5.2 Simulation Platform Introduction</td>
<td>35</td>
</tr>
<tr>
<td>5.3 A Simulation Example: Pollution Tracking and Control</td>
<td>37</td>
</tr>
<tr>
<td>6 Boundary Stabilization and Disturbance Rejection of Fractional Wave Equations</td>
<td>43</td>
</tr>
<tr>
<td>6.1 Problem definition</td>
<td>43</td>
</tr>
<tr>
<td>6.2 Boundary Stabilization</td>
<td>44</td>
</tr>
<tr>
<td>6.3 Disturbance Rejection</td>
<td>45</td>
</tr>
<tr>
<td>6.4 Boundary Control of the Fractional Wave Equation: Fractional Order Boundary Controllers</td>
<td>47</td>
</tr>
<tr>
<td>6.4.1 The Fractional Order Controller</td>
<td>47</td>
</tr>
<tr>
<td>6.4.2 Stability Studies</td>
<td>48</td>
</tr>
<tr>
<td>6.4.3 Performance Comparison</td>
<td>49</td>
</tr>
<tr>
<td>6.5 Robustness of the Fractional Order Controller Against Noises</td>
<td>53</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>54</td>
</tr>
<tr>
<td>6.1 Responsiveness and robustness comparison between fractional order controller and integer order controller</td>
<td>54</td>
</tr>
<tr>
<td>8.1</td>
<td>76</td>
</tr>
<tr>
<td>8.1 Estimated parameters, $\alpha = 1.754$, $b = 0.25$, $N = 3$</td>
<td>76</td>
</tr>
<tr>
<td>8.2</td>
<td>78</td>
</tr>
<tr>
<td>8.2 Estimated parameters when $\alpha = 1.1$, $b = 0.25$, $N = 3$</td>
<td>78</td>
</tr>
<tr>
<td>8.3</td>
<td>79</td>
</tr>
<tr>
<td>8.3 Estimated parameters when $\alpha = 2$, $b = 0.25$, $N = 3$</td>
<td>79</td>
</tr>
<tr>
<td>8.4</td>
<td>82</td>
</tr>
<tr>
<td>8.4 Estimated parameters when $\alpha = 1.75$, $b = 0.25$, different $N$s</td>
<td>82</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Tip displacement for <em>Case 1</em></td>
<td>8</td>
</tr>
<tr>
<td>2.2 Tip displacement for <em>Case 2</em></td>
<td>8</td>
</tr>
<tr>
<td>2.3 Displacement of the whole string for <em>Case 1</em></td>
<td>9</td>
</tr>
<tr>
<td>2.4 Displacement of the whole string for <em>Case 2</em></td>
<td>9</td>
</tr>
<tr>
<td>3.1 A feedback control system with a time delay</td>
<td>11</td>
</tr>
<tr>
<td>3.2 The Smith predictor</td>
<td>11</td>
</tr>
<tr>
<td>3.3 An alternative Smith predictor implementation</td>
<td>12</td>
</tr>
<tr>
<td>3.4 Modified Smith predictor with a lead-lag filter</td>
<td>14</td>
</tr>
<tr>
<td>3.5 Tip velocity and displacement without the Smith predictor</td>
<td>15</td>
</tr>
<tr>
<td>3.6 Displacement of the whole string without the Smith predictor</td>
<td>15</td>
</tr>
<tr>
<td>3.7 Tip velocity and displacement, the conventional Smith predictor</td>
<td>16</td>
</tr>
<tr>
<td>3.8 Displacement of the whole string, the conventional Smith predictor</td>
<td>16</td>
</tr>
<tr>
<td>3.9 Tip velocity and displacement, the modified Smith predictor with a lead filter</td>
<td>17</td>
</tr>
<tr>
<td>3.10 Displacement of the whole string, the modified Smith predictor with a lead filter</td>
<td>17</td>
</tr>
<tr>
<td>3.11 System with mis-matched delays</td>
<td>20</td>
</tr>
<tr>
<td>3.12 A feedback system with delay</td>
<td>21</td>
</tr>
<tr>
<td>4.1 Modified Smith predictor with a time advance approximator</td>
<td>24</td>
</tr>
<tr>
<td>4.2 Tip velocity and displacement without Smith predictor</td>
<td>25</td>
</tr>
<tr>
<td>4.3 Displacement of the whole beam without Smith predictor</td>
<td>26</td>
</tr>
<tr>
<td>4.4 Tip velocity and displacement with the conventional Smith predictor added</td>
<td>27</td>
</tr>
</tbody>
</table>
4.5 Displacement of the whole beam with the conventional Smith predictor added .................................................. 27
4.6 Tip velocity and displacement, modified Smith predictor with a lead filter ................................................................. 28
4.7 Displacement of the whole beam, modified Smith predictor with a lead filter ................................................................. 28
4.8 Bode plot of the optimal lead filter ................................................................. 29
4.9 Bode plot of the optimal time-advance approximator ................................................................. 30
4.10 Implementation of the time-advance approximator ................................................................. 30
4.11 Tip velocity and displacement, modified Smith predictor with inverse of time delay approximation ................................................................. 31
4.12 Displacement of the whole beam, modified Smith predictor with inverse of time delay approximation ................................................................. 31
5.1 Flow chart of simulation ................................................................. 37
5.2 Initial layout ................................................................. 38
5.3 $u(x, y, t)$, sensor/actuators/disturbance positions at $t = 0$ ................................................................. 39
5.4 $u(x, y, 0.8)$, sensor/actuators/disturbance positions at $t = 0.8$ ................................................................. 39
5.5 $u(x, y, 7.0)$, sensor/actuators/disturbance positions at $t = 7.0$ ................................................................. 40
5.6 $u(x, y, 7.9)$, sensor/actuators/disturbance positions at $t = 7.9$ ................................................................. 40
5.7 $u(x, y, 19.7)$, sensor/actuators/disturbance positions at $t = 19.7$ ................................................................. 41
5.8 Integration of $u(x, y, t)$ over $x$ and $y$ ................................................................. 41
6.1 Tip end movement over time for different $\alpha$ ................................................................. 45
6.2 Tip end movement over time for different $k_d$. ................................................................. 46
6.3 Tip end movement over time for different $\alpha$ ................................................................. 47
6.4 Tip end movement over time for different $\alpha$ ................................................................. 48
6.5 Displacement of the free end, $\alpha = 1.1$ ................................................................. 49
6.6 Displacement of the free end, $\alpha = 1.5$ ................................................................. 50
6.7 Displacement of the free end, \( \alpha = 1.9 \) ........................................... 50
6.8 Displacement of the free end, \( \alpha = 1, \ k = 1, \ \mu = 1 \) .......................... 52
6.9 Displacement of the whole cable, \( \alpha = 1, \ k = 1, \ \mu = 1 \) .......................... 52
6.10 Comparison between two optimal boundary controllers for 
\( \alpha = 1.5 \) ................................................................. 53
6.11 Comparison between two optimal boundary controllers for 
\( \alpha = 1.1 \) ................................................................. 55
6.12 Tip end displacement, \( \alpha = 1.75, \ \mu = 1 \) ........................................... 55
6.13 Tip end displacement, \( \alpha = 1.75, \ \mu = 0.85 \) ........................................ 56
6.14 Tip end displacement, \( \alpha = 1.75, \ \mu = 0.7 \) ........................................ 56
6.15 Tip end displacement for \( \mu = 1 \) and \( \mu = 0.7 \) ................................. 57
6.16 Displacement of whole string, \( \alpha = 2, \ \mu = 1 \) ................................. 57
6.17 Displacement of whole string, \( \alpha = 2, \ \mu = 0.7 \) ................................. 58
7.1 Displacement of the whole cable, \( k = 1, \ \alpha = 1.75, \ \theta = 0, \ 
\mu = 1. \) ................................................................. 61
7.2 Displacement of the tip end, \( k = 1, \ \alpha = 1.75, \ \theta = 0, \ \mu = 1. \) ............. 61
7.3 Displacement of the whole cable, \( k = 1, \ \alpha = 1.75, \ \theta = 0.5 \) 
sec., \( \mu = 1. \) ................................................................. 62
7.4 Displacement of the tip end, \( k = 1, \ \alpha = 1.75, \ \theta = 0.5 \) sec., \( \mu = 1. \) 
................................................................. 62
7.5 Displacement of the whole cable, \( k = 1, \ \alpha = 1.75, \ \theta = 0.5 \) 
sec., \( \mu = 1 \) with the Smith Predictor. ................................................. 63
7.6 Displacement of the tip end, \( k = 1, \ \alpha = 1.75, \ \theta = 0.5 \) sec., 
\( \mu = 1 \) with the Smith Predictor. ................................................. 63
7.7 Comparison between two optimal boundary controllers for 
\( \alpha = 1.75 \) ................................................................. 65
7.8 Comparison between two optimal boundary controllers for 
\( \alpha = 1.25 \) ................................................................. 65
7.9 A feedback control system with a time delay ................................. 67
7.10 System with mis-matched delays ........................................ 69
8.1 Displacement of the whole cable $u(x, t)$ .............................. 74
8.2 Boundary measurement data with Gaussian noise added,
$SNR = 20dB$ ............................................................. 75
8.3 Actual initial shape and estimated initial shape, $\alpha = 1.75$ ...... 76
8.4 Displacement of the whole cable $u(x, t)$ when $\alpha = 1.01$ ...... 77
8.5 Boundary measurement data for identification with Gaus-
sian noise added when $\alpha = 1.1$, $SNR = 26dB$ ................. 78
8.6 Actual initial shape and estimated initial shape, $\alpha = 1.1$ ...... 78
8.7 Displacement of whole cable when $\alpha = 2$ .......................... 79
8.8 Boundary measurement data with Gaussian noise added
when $\alpha = 2$, $SNR = 26dB$ ....................................... 80
8.9 Actual initial shape and estimated initial shape, $\alpha = 2$ ...... 80
8.10 Actual initial shape and estimated initial shape, different
Ns, $\alpha = 1.75$ ......................................................... 82
Chapter 1
Introduction

In recent years, boundary control of distributed parameter systems (DPS) has become an active research area [1–9], due to the increasing demand on high precision control of many mechanical systems, such as spacecraft with flexible attachment or robots with flexible links, which are governed by partial differential equations (PDE) rather than ordinary differential equations (ODE).

Boundary control of the wave equation with or without small delays has been studied extensively [1, 4, 7, 10–15]. In this thesis, three related problems are studied. The first problem is boundary control of the wave equation and the beam equation subject to large delays using integer order controllers and the Smith predictors, see Chapter 3 and Chapter 4. The second problem is boundary control of the fractional wave equation with or without delays using integer order controllers and fractional order controllers, see Chapter 6 and Chapter 7. Finally, Chapter 8 studies the parameter identification of the fractional wave equations.

To facilitate the study of controlling some typical distributed parameter systems, two simulation platforms are developed. The first is for boundary control of the wave equation and the beam equation, see Chapter 2. The second is for control of the diffusion process using mobile actuators and mobile sensors, see Chapter 5.

To the best of the author’s knowledge, available research results on the above topics are very few. Related publications will be reviewed in each chapter.
Chapter 2
A Hybrid Symbolic-Numerical Simulation Method

2.1 Introduction

Boundary control of PDE has become an important research area in recent years [1–6]. Contrary to the progress in the theoretical analysis, simulation examples in the publications are very few, albeit simulation plays such an important role in verifying the theoretical analysis and design, identifying the potential problems, reducing the investment, and selecting the optimal solution. The reason for this is probably that the difficulty of simulating feedback control of PDEs is beyond the capability of most commonly available mathematical tools, such as Matlab, Maple, even FEMLAB [16]. For example, Matlab PDE Toolbox is only able to solve second order PDEs with Dirichlet and/or generalized Neumann boundary conditions [17], while the PDEs of most boundary control problems are either of higher order, or/and the boundary conditions are much more complicated than what Matlab PDE Toolbox could handle.

In this chapter, an easy-to-implement, yet powerful, boundary control simulation method will be presented. The method combines the analytical method, the numerical method, and the modern symbolic algebra. Although the basic principle is simple, in the future chapters, it will be shown that the simulation method applies to a wide range of boundary control problems. The method is also much easier to implement than Finite Element Method (FEM) or Finite Difference Method (FDM) [18]. No extra software is needed except Matlab and the Matlab Symbolic Math Toolbox.

2.2 Problem Formulation, Principle, and Implementation

In this section, the following example will be used to show how a typical boundary control problem is formulated. The same example will be used to demonstrate the basic principle and implementation details of the simulation method.
Consider a string whose behavior is governed by the wave equation. Denote the displacement of the string by \( u(x, t) \) at \( x \in [0, 1] \) and \( t \geq 0 \). The string is fixed at one end and stabilized by a boundary controller at the other end. The system is represented by

\[
\begin{align*}
    u_{tt}(x, t) - u_{xx}(x, t) &= 0, \\
    u(0, t) &= 0, \\
    u_x(1, t) &= f(t),
\end{align*}
\]

where the subscript, e.g., the \( t \) as in \( u_t \), denotes a partial differential with respect to the corresponding variable. \( f(t) \) is the combination of boundary control force and the disturbance \( n(t) \) applied at the free end of the string. The control objective is to stabilize \( u(x, t) \), given the initial conditions.

Suppose the following dynamic boundary controller is designed

\[
\hat{f}(s) = (d + \frac{ks}{s^2 + \omega^2})\hat{u}_t(1, s) + \hat{n}(s),
\]

where \( \hat{f}(s) \) is the Laplace transform of the combination of boundary control force and disturbance force; \( \hat{n}(s) \) is the Laplace transform of the disturbance force \( n(t) \); \( \hat{u}_t(1, s) \) is the Laplace transform of the velocity of the free end; \( d \) and \( k \) are the control gains; \( \omega \) is the frequency of the noise.

In the following, it will be shown that the dynamic boundary controller (when \( k > 0 \)) is better than the static boundary controller (when \( k = 0 \)) in rejecting the effect of the noise \( n(t) \). This problem was raised in [4], which is among very few papers with simulation examples using FDM to simulate the overall dynamic boundary control system.

It is well-known that such linear PDEs can be solved by means of the Laplace transform [19]. Following is a summary of this method. It is assumed that the solution of a PDE is a function \( u(x, t) \) of the two independent variables \( x \) and \( t \).

1. Transform \( u(x, t) \) with respect to \( t \) by means of the Laplace transform, so an ODE is obtained for the transformed variable \( U(x, s) \):

\[
f(U(x, s), \frac{dU(x, s)}{dx}, \ldots, \frac{d^n U(x, s)}{dx^n}, x, s) = 0.
\]
2. Solve the ODE (2.5) for $U(x,s)$ as a function of $x$, with the transform variable $s$ still appearing as a parameter in the solution, and use the boundary conditions of the original problem to determine the precise form of $U(x,s)$.

3. Take the inverse Laplace transform of $U(x,s)$ with respect to $s$ to find the solution $u(x,t)$.

Several problems make the above method hard to use in practice to solve a PDE boundary control problem. First, if (2.5) is of high order, the general solution is too complicated to obtain. Second, due to the high order of the ODE and the complicated boundary conditions, the arbitrary constants in the general solution of ODE are hard to determine. Third, even if the undefined constants can be determined, usually the inverse Laplace transform can not be performed by looking up a table of transform pairs.

The above problems can be solved using the Matlab Symbolic Math Toolbox [20] and the numerical inverse Laplace transform [21,22]. The detailed implementation procedures are demonstrated in the following example.

Assume the initial conditions are

$$ u(x,0) = -0.5 \sin(0.5\pi x), \quad (2.6) $$

$$ u_t(x,0) = 0. \quad (2.7) $$

The disturbance $n(t)$ is chosen as

$$ n(t) = \cos(10t). \quad (2.8) $$

The following two cases will be simulated to show that the dynamic controller ($k > 0$) is better than the static controller ($k = 0$) in rejecting the noise.

*Case 1*: $d = 1$, $k = 10$, $\omega = 10$,

*Case 2*: $d = 1$, $k = 0$, $\omega = 10$.

The simulation of *Case 1* will be taken as an example to show the simulation steps. First, take the Laplace transform of (2.1), (2.2), and (2.3) with respect to $t$, which gives

$$ \frac{d^2 U(x,s)}{dx^2} - (s^2 U(x,s) - su(x,0) - u_t(x,0)) = 0, \quad (2.9) $$
\[ U(0, s) = 0, \quad (2.10) \]
\[ \frac{dU(1, s)}{dx} = (d + \frac{ks}{s^2 + \omega^2})(sU(1, s) - u(1, 0)) + \frac{s}{s^2 + \omega^2}, \quad (2.11) \]

where \( U(x, s) \) is the Laplace transform of \( u(x, t) \).

Substituting the initial conditions (2.6) and (2.7) into (2.9) and (2.11), we have
\[ \frac{d^2U(x, s)}{dx^2} - s^2U(x, s) + s(-0.5 \sin(0.5\pi x)) = 0, \quad (2.12) \]
\[ \frac{dU(1, s)}{dx} = (d + \frac{ks}{s^2 + \omega^2})(sU(1, s) + 0.5) + \frac{s}{s^2 + \omega^2}. \quad (2.13) \]

Solving the equations (2.10), (2.12), and (2.13) manually is daunting. However, we can take advantage of the already well developed computer symbolic mathematical tools such as the Matlab Symbolic Math Toolbox, which is used in this thesis. First, we will use \texttt{dsolve()}, which symbolically solves the ODE(s) and the boundary and/or initial condition(s). Although \texttt{dsolve()} is able to determine the arbitrary constants in the solution using the boundary and/or initial condition(s), it is found that \texttt{dsolve()} is in fact not capable enough in handling more complicated cases. So, only (2.12) will be supplied to \texttt{dsolve()} rather than supplying (2.12), (2.10), and (2.13) together. Then, the following solution with two arbitrary constants \( C_1 \) and \( C_2 \) can be obtained.

\[ U(x, s) = C_2 e^{-sx} + C_1 e^{sx} - 2 \frac{s \sin(1/2 \pi x)}{4 s^2 + \pi^2} \quad (2.14) \]

Next, \( U(x, s) \) is differentiated with respect to \( x \) to get the first order derivative of \( U(x, s) \) using the Matlab Symbolic Math Toolbox function \texttt{diff()}, which gives
\[ \frac{dU(x, s)}{dx} = -\frac{C_2 s}{e^{sx}} + C_1 s e^{sx} - \frac{s \cos(1/2 \pi x) \pi}{4 s^2 + \pi^2}. \quad (2.15) \]

Substituting \( U(x, s) \) (2.14) and its first order derivative (2.15) to the boundary conditions (2.10) and (2.11), two boundary conditions (2.16) and (2.17) with two undetermined constants \( C_1 \) and \( C_2 \) are obtained.

\[ C_1 + C_2 = 0, \quad (2.16) \]
\[ C_1 s e^s - \frac{C_2 s}{e^{sx}} + \left(1 + \frac{10s}{s^2 + 100}\right) \left(s \left(e^{-s} C_2 + e^s C_1 - \frac{2s}{4 s^2 + \pi^2}\right) + \frac{1}{2}\right) + \frac{s}{s^2 + 100} = 0. \quad (2.17) \]
Next, the two algebraic equations (2.16) and (2.17) are solved symbolically using the Matlab Symbolic Math Toolbox function `solve()`, which gives

\[ C_1 = -\frac{1}{4} \frac{e^r \left( 12 \pi^2 + 100 \pi^2 + 8 s^3 + s^2 \pi^2 \right)}{s (4 s^2 + \pi^2) \left( -5 s + s^2 (e^r)^2 + 100 (e^r)^2 + 5 s (e^r)^2 \right)} \] (2.18)

\[ C_2 = \frac{1}{4} \frac{e^r \left( 12 \pi^2 + 100 \pi^2 + 8 s^3 + s^2 \pi^2 \right)}{s (4 s^2 + \pi^2) \left( -5 s + s^2 (e^r)^2 + 100 (e^r)^2 + 5 s (e^r)^2 \right)} \] (2.19)

Now, the explicit expression of \( U(x,s) \), shown in (2.20) in Matlab notation, is actually obtained.

\[
U(x,s) = \frac{1}{4} \exp(-s*x) \exp(s) \cdot \frac{12s+2+100+8s^3+s^2}{s(4s^2+\pi^2)(-5s+s^2(e^s)^2+100(e^s)^2+5s(e^s)^2)} - \frac{2s\sin(\frac{1}{2}\pi x)}{4s^2+2778046668940015/281474976710656}
\] (2.20)

Although obtaining the explicit expression of \( U(x,s) \) is just an intermediate step of this simulation method, it will be shown in the subsequent chapters that this is critical to designing more advanced boundary controllers, because if a symbolic boundary controller \( F(s) \) is used instead of equation (2.4) and divide \( U(x,s) \) by \( F(s) \) afterward, \( U(x,s)/F(s) \), the transfer function of this control system, is obtained. The role of the transfer function in control system design cannot be overemphasized. For other numerical methods, such as FEM or FDM, it is impossible to obtain the transfer function. This is actually another advantage of the simulation method developed in this thesis.

To obtain \( u(x,t) \), the inverse Laplace transform of \( U(x,s) \) is required. The Matlab Symbolic Math Toolbox function `ilaplace()`, which takes the inverse Laplace transform symbolically, should not be used, since for such a complicated expression of \( U(x,s) \), the explicit expression of \( u(x,t) \) is usually unavailable. However, the numerical inverse Laplace transform algorithms can be made used of. There are many numerical techniques available for inverse Laplace transform [21]. Among the existing numerical inverse Laplace transform methods, the method introduced in [22] is chosen for its proven accuracy and fastness. The basic idea of this method is summarized as follows [22].

The inverse Laplace transform is defined as

\[
f(t) = \frac{1}{2\pi j} \int_{c-j\infty}^{c+j\infty} F(s) e^{st} ds.
\] (2.21)
After applying the trapezoidal quadrature formula with some rearrangements, an approximate formula in discrete form can be written as

$$\tilde{f}^k = C^k \left\{ 2 \text{Re} \left[ \sum_{n=0}^{\infty} F_n E_n^k \right] - F_0 \right\},$$

(2.22)

for $k = 0, 1, \cdots, N - 1$, with

$$\tilde{f}^k = \tilde{f}(kT), \quad C^k = \frac{\Omega}{2\pi} e^{ckT},$$

$$F_n = F(c - jn\Omega), \quad E_n^k = e^{-jkTn\Omega},$$

where $T$ and $\Omega = 2\pi/(NT)$ are sampling periods in original and transform domains, respectively. It can be proved that (2.22) corresponds to a Fourier series approximation of the original $f(t)$ when the error can theoretically be controlled in the interval $t \in (0, NT)$.

To speed up the convergence of infinite complex Fourier series, the $c$-algorithm is applied, which makes the results from FFT more accurate as if more terms are used to compute the FFT. Interested readers can refer to [24] and [25] for detailed theory and [22] for implementation details with readily available Matlab code.

At this point, the time-domain simulation is finished. In what follows, some simulation results for both Case 1 and Case 2 will be presented.

The plots of tip displacement in Case 1 and Case 2 are shown in Fig. 2.1 and Fig. 2.2, respectively. It shows clearly that the dynamic boundary controller is better than the static boundary controller in rejecting the sinusoidal noise. The two plots are identical to the simulation results reported in [4].

It is also very easy to show the displacement of the whole string as shown in Fig. 2.3 and Fig. 2.4, for Case 1 and Case 2, respectively.
Fig. 2.1: Tip displacement for Case 1

Fig. 2.2: Tip displacement for Case 2
Fig. 2.3: Displacement of the whole string for Case 1

Fig. 2.4: Displacement of the whole string for Case 2
Chapter 3

Boundary Control of the Wave Equations Subject to Large Delays

Two important research topics on control of DPS are boundary control of the wave equation and the beam equation, which are often encountered in the practical engineering design. The robustness of the boundary controllers against delays has been studied extensively [10, 12–15]. It is well-known that although a simple derivative feedback controller applied at the boundaries is sufficient to stabilize the system [7, 8], these systems become unstable when an arbitrary small time delay is introduced into the feedback loop [10, 12].

All the available publications focus on the analysis of systems against a small delay, i.e., under what conditions a very small delay will not cause instability problem and can be therefore neglected? An equally important and very practical issue is, how to synthesize a boundary controller when the delay is large and makes the systems unstable? In this chapter, the Smith predictor and its variant are applied to the boundary control of the wave equation with delayed boundary measurement to compensate the delay.

3.1 A Brief Introduction to The Smith Predictor

The Smith predictor was proposed by Smith in [27] and is probably the most famous method for control of systems with time delays [28, 29]. Consider a typical feedback control system with a time delay in Fig. 3.1, where \( C(s) \) is the controller; \( P(s)e^{-\theta s} \) is the plant with a time delay \( \theta \).

With the presence of the time delay, the transfer function of the closed-loop system relating the output \( y(s) \) to the reference \( r(s) \) becomes

\[
\frac{y(s)}{r(s)} = \frac{C(s)P(s)e^{-\theta s}}{1 + C(s)P(s)e^{-\theta s}}. \tag{3.1}
\]
Fig. 3.1: A feedback control system with a time delay

Obviously, the time delay $\theta$ directly changes the closed-loop poles. Usually, the time delay reduces the stability margin of the control system, or more seriously, destabilizes the system.

The classical configuration of a system containing a Smith predictor is depicted in Fig. 3.2, where $\hat{P}(s)$ is the assumed model of $P(s)$ and $\hat{\theta}$ is the assumed delay. The block $C(s)$ combined with the block $\hat{P}(s) - \hat{P}(s)e^{-\hat{\theta}s}$ is called “the Smith predictor”. If we assume the perfect model matching, i.e., $\hat{P}(s) = P(s)$ and $\theta = \hat{\theta}$, the closed-loop transfer function becomes

$$\frac{y(s)}{r(s)} = \frac{C(s)P(s)e^{-\theta s}}{1 + C(s)P(s)}. \quad (3.2)$$

Fig. 3.2: The Smith predictor

Now, it is clear what the underlying idea of the Smith predictor is. With the perfect model matching, the time delay can be removed from the denominator of the transfer function, making the closed-loop stability irrelevant to the time delay.

An alternative implementation of the Smith predictor is shown in Fig. 3.3. Since this configuration makes the design of the Smith predictor more convenient, this implementation will be used in this thesis.
3.2 Problem Formulation

Consider a string clamped at one end and is free at the other end. Denote the displacement of the string by \( u(x,t) \), where \( x \in (0,1) \) and \( t \geq 0 \). The string is controlled by a boundary control force at the free end. The governing equations are given as

\[
\begin{align*}
    u_{tt} - u_{xx} &= 0, \\
    u(0,t) &= 0, \\
    u_x(1,t) &= f(t),
\end{align*}
\]

where \( f(t) \) is the boundary control force applied at the free end of the string.

It is well-known that the following boundary controller is a stabilizing controller [8]:

\[
f(t) = -ku_t(1,t)
\]

where \( k > 0 \) is the constant gain.

However, in [12], it was shown that the system based on the boundary controller (3.6) becomes unstable when an arbitrary small time delay is introduced into the feedback loop as follows:

\[
f(t) = -ku_t(1,t - \theta),
\]

where \( \theta \) is the time delay.

It should be noted that this delay-induced instability exists as well in the boundary control of the beam equation [10,15].

Fig. 3.3: An alternative Smith predictor implementation
Comparing the equation (3.7) with Fig. 3.2, it can be seen that to apply the Smith predictor, the output $y$ is the tip end displacement, $C(s)$ is the derivative controller $ks$, and $P(s)$ is the transfer function from the control force $f(t)$ to the tip end displacement.

Assuming $\hat{P}(s) = P(s)$ and the time delay $\theta$ is known, the remaining problem is how to get $P(s)$. In Chapter 2, a method combining the symbolic algebra and numerical method was designed to simulate some typical boundary control problems. A by-product of the simulation method is that, in the intermediate steps, $U(x,s)$, the Laplace transform of $u(x,t)$ with respect to $t$, can be explicitly obtained, which can be used to get $P(s)$. Since the wave equation is relatively simple, $P(s)$ can be obtained manually. However, for higher order systems, manual derivation can be very difficult.

Using the method described in Chapter 2, $P(s)$ is obtained as follows:

$$P(s) = \frac{\sinh(s)}{\cosh(s)}.$$  \hspace{1cm} (3.8)

At this point, the controller design based on the configuration shown in Fig. 3.3 is almost finished. As commented in [29], the Smith predictor is best suitable for tracking problems rather than regulation problems, which is the case in the thesis. Even worse is the situation that the initial conditions actually act as disturbances to the Smith predictor shown in Fig. 3.3. To improve the regulation performance, some modified Smith predictors have been designed [28] [30–32]. In this chapter, one of the modified Smith predictor schemes is used: the lead-lag compensation scheme shown in Fig. 3.4. A simple optimization search routine to determine the suitable parameters is also developed.

### 3.3 Simulation Results

In the following simulation, $k = 1$ and $\theta = 0.1$ are chosen. The initial conditions are chosen as

$$u(x,0) = -\sin(0.5\pi x),$$ \hspace{1cm} (3.9)

$$u_t(x,0) = 0.$$ \hspace{1cm} (3.10)

If only the static feedback controller (3.7) is used, the simulation results are shown in Fig. 3.5 and Fig. 3.6. The controller works well at the beginning, driving the tip end to
the zero position. However, the frequency of the vibration is increasing over time. When
the frequency is high enough, the time delay causes the control force to be in phase rather
than out of phase with the tip velocity, thus making the system unstable.

After the Smith predictor shown in Fig. 3.2 is added, the simulation results are shown
in Fig. 3.7 and Fig. 3.8. An obvious improvement over Fig. 3.5 and Fig. 3.6 is that the
velocity and displacement are not increasing to infinity over time. However, the string
vibrates with a relatively large non-decreasing magnitude, rather than converging to zero
position as we expected. As pointed out in [28], the open-loop poles are presented in $G_d$,
the transfer function from the response $y$ to the disturbance $d$. These poles are excited by
input disturbances but not by the reference. Depending on their locations relative to the
closed-loop poles, these poles may dominate the response. In this case, the non-zero initial
conditions act as disturbances, which deteriorate severely the regulation performance of
the Smith predictor.

To get a better performance, a modified Smith predictor [30,31], shown in Fig. 3.4
is introduced. The filter parameters, $k$, $z$, and $p$, are determined by an optimization
search procedure. The optimal values are found to be that $k = 1.1118$, $z = 0.8864$, and
$p = 1.2467$. The optimal filter turns out to be a lead filter. The simulation results are
shown in Fig. 3.9 and Fig. 3.10.

Compared to Fig. 3.7 and Fig. 3.8, the addition of an optimal lead-lag filter attenuates
almost completely the output oscillation.

\[ C(s) P(s) \quad + \quad d \quad + \quad e^{-\theta s} \quad + \quad \]

\[ \begin{align*}
\frac{k(s+z)}{s+p} \\
\end{align*} \]

\[ P_0(s) \]

\[ \begin{align*}
r & \quad \rightarrow \quad C(s) \\
& \quad \rightarrow \quad P(s) \\
& \quad \rightarrow \quad y \\
\end{align*} \]

Fig. 3.4: Modified Smith predictor with a lead-lag filter
Fig. 3.5: Tip velocity and displacement without the Smith predictor

Fig. 3.6: Displacement of the whole string without the Smith predictor
Fig. 3.7: Tip velocity and displace, the conventional Smith predictor

Fig. 3.8: Displacement of the whole string, the conventional Smith predictor
Fig. 3.9: Tip velocity and displace, the modified Smith predictor with a lead filter

Fig. 3.10: Displacement of the whole string, the modified Smith predictor with a lead filter
3.4 The Damped Wave Equation Case and the Robustness Issue

In this section, the robustness of the Smith predictor applied to the boundary control of a general (damped) wave equation subject to large delays is studied. The control scheme is shown to be stable and robust against a small difference between the actual delay and the assumed delay.

3.4.1 Boundary Control of the Damped Wave Equation with Large Delays

Consider a string clamped at one end and is free at the other end. Denote the displacement of the string by \( u(x, t) \), where \( x \in [0, 1] \) and \( t \geq 0 \). The string is controlled by a boundary control force at the free end. The governing equations are given as

\[
\begin{align*}
\ddot{u}(x, t) - u_{xx}(x, t) + 2au_t(x, t) + a^2u(x, t) &= 0, \\
u(0, t) &= 0, \\
u_x(1, t) &= f(t),
\end{align*}
\]

where \( a > 0 \) is the damping constant and \( f(t) \) is the boundary control force applied at the free end of the string.

It is known that the following boundary feedback controller stabilizes the system [7],

\[
f(t) = -ku_t(1, t),
\]

where \( k > 0 \) is the constant boundary control gain.

Consider the presence of a time delay in the feedback loop, shown as follows.

\[
f(t) = -ku_t(1, t - \theta),
\]

where \( \theta \) is the time delay.

In [10] and [14], it was shown that if \( k \) and \( a \) satisfy

\[
k\frac{e^{2a} + 1}{e^{2a} - 1} < 1,
\]

then the delayed feedback systems is stable for all sufficiently small delays.
In this section, the Smith predictor is applied to solve the following problem: what if the time delay $\mu$ is large enough to make the system unstable?

Assuming zero initial conditions of $u(x, 0)$ and $u_t(x, 0)$. Following the procedure presented in Chapter 2, the Laplace transform of the tip end displacement can be obtained as follows.

$$U(1, s) = \frac{F(s)(1 - e^{-2(s+a)})}{(s+a)(1 + e^{-2(s+a)})}.$$

The transfer function of the plant, $P(s)$ in Fig. 3.2, can also be obtained as

$$P(s) = \frac{U(1, s)}{F(s)} = \frac{1 - e^{-2(s+a)}}{(s+a)(1 + e^{-2(s+a)})}.$$

Finally, the Smith predictor, denoted as $C_{sp}(s)$, is obtained as follows:

$$C_{sp}(s) = \frac{ks}{1 + ksP(s)(1 - e^{-\theta s})}.$$

Notice that the controller (3.19) is physically implementable.

### 3.4.2 Stability and Robustness Analysis

In [7], the stability of the controller (3.14) was proved for the boundary control of the damped wave equation without delays. If the assumed delay is equal to the actual delay, the Smith predictor removes the delay term completely from the denominator of the closed-loop transfer function. This means the stability of the controller (3.19) is already proved.

Since the actual delay $\theta$ and the assumed delay $\hat{\theta}$ can not be exactly the same, another important issue is the robustness of the controller (3.19), i.e., what if an unknown small difference $\epsilon$ between the assumed delay and the actual delay is introduced to the system, as shown in Fig. 3.11?

To study the robustness of the controller (3.19), the following theorem [13, 14] is important.

**Theorem 1** Let $H(s)$ be the open-loop transfer function as illustrated in Fig. 3.12 and $\mathcal{D}_H$ the set of all its poles. Define two closed-loop transfer functions $G_0(s)$ and $G_\epsilon(s)$ as

$$G_0(s) = \frac{H(s)}{1 + H(s)},$$
and

\[ G_\epsilon(s) = \frac{H(s)}{1 + e^{-\epsilon s}H(s)}. \]

Define again

\[ C_0 = \{s \in \mathbb{C} | \Re(s) > 0\}, \]

and

\[ \gamma(H(s)) = \limsup_{|s| \to \infty, s \in C_0 \setminus D_H} |H(s)|. \]

Suppose \( G_0 \) is \( L^2 \)-stable. If \( \gamma(H) < 1 \), then there exists \( \epsilon^* \) such that \( G_\epsilon \) is \( L^2 \)-stable for all \( \epsilon \in (0, \epsilon^*) \).

The underlying idea of the above theorem is that the robustness of the closed-loop transfer function \( G_0(s) \) against a small unknown delay can be determined by studying the open-loop transfer function \( H(s) \). Now we can prove the robustness of the controller (3.19).

CLAIM:

If \( \hat{\theta} \) is chosen as the minimum value of the possible delay and \( k \) is chosen to satisfy

\[ k \frac{e^{2\hat{\theta}}}{e^{2\hat{\theta}} - 1} \leq \frac{1}{3}, \]

then the controller (3.19) is robust against a small difference \( \epsilon \) between the assumed delay \( \hat{\theta} \) and the actual delay \( \theta = \hat{\theta} + \epsilon \).

Proof:

Fig. 3.11: System with mis-matched delays
Let $T(s) = ksP(s)$, then

$$|H(s)| = \frac{1}{|\left(\frac{1}{T(s)} + 1\right)e^{\hat{\theta}s} - 1|}$$  \hspace{1cm} (3.21)$$

Let $Q(s) = (\frac{1}{T(s)} + 1)e^{\hat{\theta}s} - 1$, then

$$|Q(s)| = \left|\left(\frac{1}{T(s)} + 1\right)e^{\hat{\theta}s} - 1\right|$$
\begin{align*}
\geq & \left|\left(\frac{1}{T(s)} + 1\right)e^{\hat{\theta}s}\right| - 1 \\
\geq & \left|\frac{1}{T(s)} + 1\right| \left|e^{\hat{\theta}s}\right| - 1
\end{align*}$$ \hspace{1cm} (3.22)$$

In [14], it was proved that

$$\limsup_{|s| \to \infty, s \in \mathbb{C}_0} |T(s)| = k \frac{e^{2a} + 1}{e^{2a} - 1}$$

So if $k \frac{e^{2a} + 1}{e^{2a} - 1} \leq \frac{1}{3}$, for $|s|$ large enough,

$$\left|\frac{1}{T(s)} + 1\right| \geq \left|\frac{1}{T(s)}\right| - 1$$
\begin{align*}
\geq & 2
\end{align*}$$ \hspace{1cm} (3.23)$$

Considering $|e^{\hat{\theta}s}| > 1$, we have

$$|Q(s)| > 1$$ \hspace{1cm} (3.24)$$

Fig. 3.12: A feedback system with delay
So

\[
\limsup_{|s| \to \infty, s \in \mathbb{C}_0} |H(s)| < 1. \tag{3.25}
\]

**Remarks:**

- In *Theorem 1*, \( \epsilon \) is positive. To satisfy this condition, \( \hat{\theta} \) should be chosen as the minimal value of the possible delay.

- The damping constant \( a \) plays a key role in making the controllers (both the original derivative controller \( ks \) and the Smith predictor) robust. If \( a = 0 \), the damped wave equation becomes the conservative wave equation, which is studied in the first part of this chapter. The transfer function becomes

\[
P(s) = \frac{1 - e^{-2s}}{s(1 + e^{-2s})}. \tag{3.26}
\]

Now \( P(s) \) has infinite number of poles on the imaginary axis. In order to make \( \gamma(H(s)) < 1 \), controllers must cancel these poles completely, which is impossible due to the uncertainty of the plant parameters. This means both the original derivative controller \( ks \) and the Smith predictor are not robust when applied to the boundary control of the conservative wave equation.
Chapter 4
Boundary Control of the Beam Equation with Large Delays

In this chapter, similar to in Chapter 3, the Smith predictor is applied to the boundary control of the beam equation subject to large delays. Due to the difference between the beam equation and the wave equation, it will be shown that a further modified Smith predictor is required to achieve the same performance as in the wave equation case.

4.1 Problem Formulation

Consider a flexible beam clamped at one end and is free at the other end. Denote the displacement of the beam by $u(x, t)$, where $x \in (0, 1)$ and $t \geq 0$. The beam is controlled by a boundary control force at the free end. The governing equations are given as

$$u_{tt} + u_{xxxx} = 0,$$

$$u(0, t) = 0,$$  \hspace{1cm} (4.1)

$$u_x(0, t) = 0,$$  \hspace{1cm} (4.2)

$$u_{xx}(1, t) = 0,$$  \hspace{1cm} (4.3)

$$u_{xxx}(1, t) = f(t),$$  \hspace{1cm} (4.4)

where $f(t)$ is the boundary control force applied at the free end of the beam.

It is well-known that the following boundary controller is an stabilizing controller [8]:

$$f(t) = -k_d u_t(1, t)$$  \hspace{1cm} (4.5)

where $k_d > 0$ is the constant gain and the suffix $d$ shows it is a derivative gain.

In [12], it was shown that the system based on the boundary controller (4.6) becomes unstable when an arbitrary small time delay is introduced into the feedback loop as follows:

$$f(t) = -k_d u_t(1, t - \theta),$$  \hspace{1cm} (4.7)
where \( \theta \) is the time delay.

In this following, the Smith predictor will be introduced to compensate the time delays.

Comparing the equation (4.7) with Fig. 3.2, it is clear that the output \( y \) is the tip end displacement, \( C(s) \) is the derivative controller \( k_d s \), and \( P_0(s) \) is the transfer function from the control force \( f(t) \) to the tip end displacement.

Using the method introduced in Chapter 2, \( P(s) \) can be obtained as follows:

\[
P_0(s) = \frac{(1-i) \sqrt{-s^2} (1+i e^{2i}) \sqrt{-s^2-4e^{-2i}} \sqrt{-s^2+4e^{-2i}(-1+i) \sqrt{-s^2} + 4e^{-2i}(-1+i) \sqrt{-s^2} + 1 + e^{-2i}}}{s (e^{2i} \sqrt{-s^2+e(-2+2i) \sqrt{-s^2+4e^{-2i}(-1+i) \sqrt{-s^2}} + 1 + e^{-2i}) \sqrt{-s^2}}}
\] (4.8)

At this point, controller design is almost finished based on the configuration shown in Fig. 3.3. How, as commented in [29], the Smith predictor is more suitable for tracking problems rather than regulation problems, which is the case in this chapter. Even worse is the situation that the initial conditions actually act as disturbances to the Smith predictor shown in Fig. 3.3. In section 4.2, it will be demonstrated how the initial conditions, acting as disturbances, deteriorate severely the performance of the Smith predictor. To improve the regulation performance, some modified Smith predictors have been designed [28,30–32]. In this chapter, two modified Smith predictor schemes will be used: the lead-lag compensation scheme shown in Fig. 3.4 and the time advance approximator scheme shown in Fig. 4.1. For both of the two schemes, an optimization search routine is developed to determine the suitable parameters by minimizing the maximal amplitude in the last 1/5th part of the simulation time, where the 1/5 is determined by trial-and-error.

![Fig. 4.1: Modified Smith predictor with a time advance approximator](image-url)
4.2 Simulation Results and Analysis

In the following simulation, \( k_d = 0.1 \) and \( \theta = 0.05 \) are chosen. The initial conditions are chosen as

\[
\begin{align*}
  u(x,0) &= x^3 - 3x^2, \\
  u_t(x,0) &= 0.
\end{align*}
\]

(4.9) (4.10)

The initial condition (4.9) is a typical displacement profile when the beam is subject to a static force \( f = -1 \) at the free end [33].

If only the static feedback controller (4.7) is used, the simulation results are shown in Fig. 4.2 and Fig. 4.3. The controller works well at the beginning, driving the tip end to the zero position. However, the frequency of the vibration is increasing over time. When the frequency is high enough, the time delay causes the control force to be in phase rather than out of phase with the tip velocity, making the system unstable.

![Tip velocity and displacement without Smith predictor](image)

Fig. 4.2: Tip velocity and displacement without Smith predictor

After the Smith predictor shown in Fig. 3.2 is added, the simulation results are shown in Fig. 4.4 and Fig. 4.5. An obvious improvement over Fig. 4.2 and Fig. 4.3 is that the velocity and displacement are not increasing to infinity over time. However, the beam
vibrates with a relatively large yet non-decreasing magnitude, rather than converging to zero position. As pointed out in [28], the open-loop poles are presented in $G_d$, the transfer function from the response $y$ to the disturbance $d$. These poles are excited by input disturbances but not by the reference. Depending on their locations relative to the closed-loop poles, these poles may dominate the response. The non-zero initial conditions act as disturbances, which deteriorate severely the regulation performance of the Smith predictor.

To get a better performance, the modified Smith predictor [30,31], shown in Fig. 3.4, will be applied. The filter parameters, $k$, $z$, and $p$, are determined by the optimization search procedure. The optimal values are found to be that $k = 1.0681$, $z = 0.6283$, and $p = 1.2681$. The optimal filter turns out to be a lead filter. The simulation results are shown in Fig. 4.6 and Fig. 4.7.

We can see that although the lead filter is not able to attenuate completely the vibration caused by the time delay, it does reduce the magnitude of the velocity and displacement to a much smaller value, compared to the results in Fig. 4.4 and Fig. 4.5. To help further understand how the lead filter is working, its Bode plot is shown in Fig. 4.8, which is a high pass filter. Studying Fig. 4.4 carefully, it can be observed that both the velocity and displacement of the tip end are composed of a low frequency component with a bigger
Fig. 4.4: Tip velocity and displacement with the conventional Smith predictor added

Fig. 4.5: Displacement of the whole beam with the conventional Smith predictor added
Fig. 4.6: Tip velocity and displacement, modified Smith predictor with a lead filter

Fig. 4.7: Displacement of the whole beam, modified Smith predictor with a lead filter
magnitude and a high frequency component with a smaller magnitude. In Fig. 4.6, the low frequency component is in fact filtered out by the high pass filter, making the magnitude of the vibration much smaller.

![Bode plot of the optimal lead filter](image)

**Fig. 4.8: Bode plot of the optimal lead filter**

To attenuate the magnitude of the vibration further, another more advanced modified Smith predictor configuration [32], shown in Fig. 4.1, is applied, where the low pass filter $B(s)$ is expressed as

$$B(s) = \frac{k}{1 + \tau s}.$$  \hspace{1cm} (4.11)

The underlying idea is that, when $k$ is large and the frequency is low, the filter in Fig. 4.1 approximates $e^{\theta s}$, which is impossible to realize physically. If $e^{\theta s}$ can be approximated closely enough, the effect of time delay, $e^{-\theta s}$, can actually be canceled. As before, the parameters of the filter are optimized and they are turned out to be: $k = 1.0364 \times 10^4$ and $\tau = 7.2710$. The Bode plot of this time-advance approximator is shown in Fig. 4.9. We can see that it does approximate $e^{\theta s}$ very well at low frequencies. Unlike $e^{\theta s}$, this time advance approximator is implementable physically. An implementation method is shown in Fig. 4.10. The simulation results are shown in Fig. 4.11 and Fig. 4.12.
Fig. 4.9: Bode plot of the optimal time-advance approximator

Fig. 4.10: Implementation of the time-advance approximator
Fig. 4.11: Tip velocity and displace, modified Smith predictor with inverse of time delay approximation

Fig. 4.12: Displacement of the whole beam, modified Smith predictor with inverse of time delay approximation
It can be seen that this modified Smith filter with a time-advance approximator generates much better results than the modified Smith predictor with a lead filter does. The magnitudes of both velocity and displacement are very small.
Chapter 5

Diff-MAS2D: a Simulation Platform for Measurement and Actuation Scheduling in Diffusion Process with Mobile Actuators and Sensors

5.1 Background

Sensor networks are drawing increased attention from research communities, industry sectors, and government agencies. As stated in [34], sensor networks will “have significant impact on a broad range of applications relating to national security, health care, the environment, energy, food safety, and manufacturing. The convergence of the Internet, communications, and information technologies with techniques for miniaturization has placed sensor technology at the threshold of a period of major growth.” Recent surveys on sensor networks [35–39] also indicate the importance of sensor networks research. Many on-going efforts are focused on various specific issues in sensor networks such as sensor structures [40–42], communication [36,43], data processing and sensor fusion methods [35, 37,44], sensor deployment and localization [35,37,45], calibration [46,47], etc.

However, from dynamic systems control point of view, the sensor networks should be part of a complete system, especially a control system, with a specific mission defined. Although a few systems consisting sensor networks have been proposed [48–50] to carry out specific missions, neither of the missions chooses control as the final goal. So far, there is no such real-time, closed-loop distributed feedback control system involving networked actuators and sensors [51,52]. However, there are applications that require the use of mobile actuators and sensor networks, denoted as MAS-Net [54]. These include the following motivating examples.

- Application Scenario 1 (land): In this case, each networked sensor is mounted on a ground mobile robot. The mission is to determine the safe radiation boundary of
the radiation field from possibly multiple radiation sources. Each robot is actuated according to spatial and temporal sensed information (radiation gradient, spatial position, etc.) from more than one actuated or mobile sensors.

- Application Scenario 2 (water): This case is similar to Application Scenario 1 if the toxic diffusion source is a one-time pouring and the diffusion is in steady state. However, the boundary may be dynamically evolving as the toxic source keeps polluting the reservoir. The actuated or mobile sensors are autonomous boats mounted with toxic chemical concentration sensors. The boats are commanded according to the spatial-temporal sensed information from more than one sensor. Furthermore, assume that some of the boats (not all of the boats) are equipped with relevant neutralizing chemicals to make the water detoxified. By proper design of the distributed sensing and actuation/control strategies, it is possible to control the zone or shape of the toxic region to match the given desirable zone/shape. Now we have a complex distributed feedback control system that is more challenging than the networked actuators and sensors themselves.

- Application Scenario 3 (air): This scenario is similar to the above water case, but it is more complicated since 3D space must be explored. Here, the actuated or mobile sensors are unmanned aerial vehicles (UAVs) equipped with concentration detectors and anti-contamination chemical agent(s) distributors. For this case, a more detailed description can be found in [53].

Motivated by these examples, in [54], two high-level tasks or missions were introduced, i.e., the diffusion boundary determination and zone control via mobile actuator-sensor networks. As a means of exploring the algorithmic, computational, and practical issues of the problems, a 2-D MAS-Net hardware testbed, introduced in [55]. It is well-known that software simulation plays an equally important (in some situations, more important) role as hands-on experiments. There are lots of readily available mathematical tools for solving PDE problems, such as MATLAB PDE Toolbox [56], FEMLAB [57], Nastran [58],
ANSYS [59]. However, if studying these tools carefully, we may find that neither is suitable for simulating the problems mentioned above due to the following reasons:

- Neither is designed with movable sensors in mind. Although this can be worked around with careful post-process, it makes code writing for on-line sensor scheduling hard and error-prone.
- Neither is designed with movable actuators in mind. This shortcoming is vital, since it makes simulation of closed-loop control of PDEs impossible.

In view of the above problems, a software platform was developed by the Author, named **Diff-MAS2D**, for simulation of measurement scheduling and controls in diffusion process with moving sensors and moving actuators.

### 5.2 Simulation Platform Introduction

**Diff-MAS2D** is able to simulate closed-loop control of the diffusion process and the wave equation, respectively, using movable sensors and movable actuators. Specifically, **Diff-MAS2D** is used to solve the following PDE:

\[
\frac{\partial u}{\partial t} = k \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + f(\bar{u}, x, y, t), \tag{5.1}
\]

where \( u = u(x,y,t) \) is the variable to be controlled; \( 0 \leq x \leq 1 \) and \( 0 \leq y \leq 1 \) is the spatial domain; \( t \geq 0 \) is the time domain; \( k \) is a positive real constant related to system parameters; \( f(\bar{u}, x, y, t) \) is a combination of control and disturbances

\[
f(\bar{u}, x, y, t) = f_c(\bar{u}(x,y,t), x, y, t) + f_d(x, y, t),
\]

where \( \bar{u}(x,y,t) \) is the measured data of \( u(x,y,t) \) from the movable sensors; \( f_c(\bar{u}(x,y,t), x, y, t) \) is the control applied by the movable actuators; \( f_d(x, y, t) \) is the disturbance. The exact format of \( f_c(\bar{u}(x,y,t), x, y, t) \) depends on the closed-loop control law designed by the user.

**Diff-MAS2D** is designed as complete simulation environments within Matlab for the end user. Internally, \( u(x,y,t) \) is discretized in spatial domain manually using finite difference method; while the discretization and integration in time domain are left to Matlab.
There are two main advantages with the different treatments of spatial domain and time domain. First, the coding is greatly simplified, since $u(x, y, t)$ is only discretized manually in spatial domain. Second, since the discretization and integration in time domain are handled by Matlab itself, for the end user, the large amount of Matlab functions can be utilized in the path planning of the sensors/actuators and in the control law design.

For the end user, arbitrary combination of the following two types of boundary conditions can be used as boundary conditions for each boundary ($x = 0$, $x = 1$, $y = 0$, and $y = 1$).

- **Dirichlet boundary condition**
  \[ u = C \]  
  where $C$ is a real constant.

- **Neumann boundary condition**
  \[ \frac{\partial u}{\partial n} = C_1 + C_2 u \]  
  where $C_1$ and $C_2$ are two real constants; $n$ is the outward direction normal to the boundary.

The main features of **Diff-MAS2D** also include:

- Any number of sensors and actuators can be used.
- Sensors and actuators can be collocated (bound together) or non-collocated (separated).
- Disturbances can be movable and time-varying.
- Movement of sensors and actuators can be open-loop (designed by the user as functions of time only) or closed-loop (designed by the user as functions of $t$, $\tilde{u}(x, y, t)$, sensor position/velocity, and actuator position/velocity).
- Arbitrary control algorithms can be applied in $f_e(\tilde{u}(x, y, t), x, y, t)$. 
• **Diff-MAS2D** is designed with easy-to-use in mind. The user needs only consider the high-level problems, such as the control law design and on-line sensor/actuator scheduling, without knowing the internal implementation details of **Diff-MAS2D**.

• **Diff-MAS2D** is designed as a platform to simulate distributed control systems, *i.e.*, the movement of each sensor/actuator can be designed individually based on the information (position and velocity) of other sensors/actuators and the measurement of the sensors.

The flow chart of **Diff-MAS2D** is shown in Fig. 5.1.

![Flow chart of simulation](image)

**Fig. 5.1: Flow chart of simulation**

### 5.3 A Simulation Example: Pollution Tracking and Control

To show the capabilities and the unique features of **Diff-MAS2D**, the following problem is simulated.

The diffusion process (5.1) will be controlled. The initial condition is 
\[
u(x, y, 0) = 0
\]
and the boundary condition for all four boundaries is 
\[
\frac{\partial u}{\partial n} = 0.
\]

Assume there is a point disturbance \( f_d = 5 \), which can be viewed as a pollution source, with following sinusoidal movement in \( x \) direction

\[
x = 0.3 \sin(0.628t) + 0.5, \quad y = 0.25.
\]

The objective is to control \( u(x, y, t) \) to be as close to zero as possible using a small number of actuators and sensors. To achieve this goal, the best strategy is probably to let the actuators track and catch the pollution source and apply the control \( f_c \). By doing this,
most part of the pollution can be eliminated as soon as the pollution is emitted, before it spreads and pollutes the whole area.

![Fig. 5.2: Initial layout](image)

To track the pollution source, the actuators/sensors are designed to move in the direction of $\nabla \tilde{u}(x, y, t)$, the gradient of $u(x, y, t)$ around the position of sensors/actuators. To calculate the gradient, we use five sensors and five actuators and choose the collocated scheme. The sensors/actuators are formed in a cross shape, shown in Fig. 5.2, the initial layout ($t = 0$) of the simulation. The gradient is approximately calculated by (denoting the three sensors in horizontal position as $s_1$, $s_2$, $s_3$, and the other two sensors in vertical position as $s_4$ and $s_5$)

$$\nabla \tilde{u}(x, y, t) \approx \left[ \frac{\tilde{u}_{s3} - \tilde{u}_{s1}}{x_{s3} - x_{s1}}, \frac{\tilde{u}_{s4} - \tilde{u}_{s5}}{x_{s4} - x_{s5}} \right].$$

A simple proportional control law is chosen for the actuators to eliminate the pollution. Denote the five actuators as $a_1$ through $a_5$, the control applied by each actuator is formulated as

$$f_{ai} = -45 \tilde{u}_{si}, \quad i = 1, \ldots, 5$$

(5.4)

The evolution of $u(x, y, t)$ can be viewed through an animation generated by the post-process program. We will use a few pictures to shown the control results.
Figure 5.3 shows $u(x, y, t)$ and the positions of the sensors, actuators, and the disturbance at $t = 0$. The left part of the figure is a 3D plot of $u(x, y, 0)$ and the right part of the figure shows the current positions of the sensors, actuators, and the disturbance.

Figure 5.4 shows $u(x, y, 0.8)$ and the positions of the sensors, actuators, and the disturbance at $t = 0.8$. We can observe a peak in the 3D plot, caused by the pollution source. Since the pollution has not spread to the position of the sensors/actuators, the sensors and the actuators are at almost the same position as in 5.3.
Figure 5.5 shows $u(x, y, 7.0)$ and the positions of the sensors, actuators, and the disturbance at $t = 7.0$. We can see that the sensors/actuators have started to track the pollution source, although not catching it yet. From the 3D plot, the environment can be seen as having been polluted (the color changed from blue to green).

Figure 5.6 shows $u(x, y, 7.9)$ and the positions of the sensors, actuators, and the disturbance at $t = 7.9$. We can see that the sensors/actuators have caught the pollution source. The peaks in Fig. 5.4 and Fig. 5.5 disappeared, showing the effect of the control.
law (5.4). After $t = 7.9$, the pollution source is caught till the end of simulation, as shown in Fig. 5.7.

Fig. 5.7: $u(x, y, 19.7)$, sensor/actuators/disturbance positions at $t = 19.7$

Finally, to compare the effects of different path planning and control algorithms, a possible benchmark is to compare the time profile of total pollution over the whole area:

$$u_I(t) = \int_0^1 \int_0^1 u(x, y, t) dx dy$$

(5.5)
$u_t(t)$ is shown in Fig. 5.8. Before the pollution source is captured by the actuators at $t \approx 10$, the total pollution is always increasing. After the pollution source is captured, not only the instant pollution emission is eliminated, due to the diffusion process, the total pollution level is also dropping.

The above simulation example shows some of the capabilities and features of Diff-MAS2D. To the best of the authors’ knowledge, Diff-MAS2D is the only available software package capable of simulating control of the diffusion process using movable sensors and actuators. With Diff-MAS2D, some hard questions might be answered, such as

- Given a specification, what are the minimal number of sensors and the minimal number of actuators required?
- What are the advantages of disadvantages of the collocated scheme and the non-collocated scheme?
- Any better control laws than (5.4)?
- Any better schemes to track the disturbances?
- ……
Chapter 6
Boundary Stabilization and Disturbance Rejection of Fractional Wave Equations

Fractional diffusion and wave equations are obtained from the classical diffusion and wave equations by replacing the first and second order time derivative term by a fractional derivative \((0, 1)\) and \((1, 2)\), respectively. There has been a growing interest in investigating the solutions and properties of these equations because many of the universal phenomenons can be modeled accurately using the fractional diffusion and wave equations [60]. Research has been focused on the analytical solution to the fractional diffusion and wave equations. In [61], the solution to the fractional diffusion equation was given in closed form in terms of Fox functions. In [62], the fractional diffusion and wave equations were reformulated as integrodifferential equations. Analytical solutions for the Green’s functions of the latter were then found. In [63], the fractional wave equation was shown to govern the propagation of stress waves in viscoelastic solids. The transition from a pure diffusion process to a pure wave process was also shown when the time derivative increases from 1 to 2. In [64], a general solution was given for a fractional diffusion-wave equation defined in a bounded space domain.

In this chapter, the boundary control of a string governed by the fractional wave equation will be studied.

6.1 Problem definition

Consider a string governed by the fractional wave equation, fixed at one end, and stabilized by a boundary control at the other end. The system can be represented by

\[
\frac{\partial^\alpha u}{\partial t^\alpha} = \frac{\partial^2 u}{\partial x^2}, \quad 1 < \alpha < 2, \quad x \in [0, 1], \quad t \geq 0
\]  \hspace{1cm} (6.1)

\[
u(0, t) = 0, \quad t \geq 0.
\]  \hspace{1cm} (6.2)
where \( u(x, t) \) is the displacement of the string at \( x \in [0, 1] \) and \( t \geq 0 \), \( f(t) \) is the boundary control force at the free end of the string, \( u_0(x) \) and \( v_0(x) \) are the initial conditions of displacement and velocity, respectively.

The control objective is to stabilize \( u(x, t) \), given the initial conditions (6.4) and (6.5).

The following definition is adopted for the fractional derivative of \( \alpha \) of function \( f(t) \) [63,65],

\[
\frac{d^\alpha}{dt^\alpha} f(t) \doteq \begin{cases} 
  f^{(n)}(t) & \text{if } \alpha = n \in N, \\
  \frac{t^{n-\alpha-1}}{\Gamma(n-\alpha)} * f^{(n)}(t) & \text{if } n - 1 < \alpha < n,
\end{cases}
\]

where the * denotes the time convolution between two causal functions.

Based on the definition of (6.6), the Laplace transform of the fractional derivative is

\[
\mathcal{L} \left\{ \frac{d^\alpha}{dt^\alpha} \right\} = s^\alpha F(s) - \sum_{k=0}^{n-1} f^k(0^+) s^{\alpha-1-k}
\]

### 6.2 Boundary Stabilization

In this section, the performance and properties of the following controller is studied:

\[
f(t) = k_d u_t(x, t)
\]

where \( k_d \) is the controller gain and the suffix \( d \) means it is a derivative gain.

Although the control law (6.8) has been widely used in the boundary control of wave equation and beam equation [3,7,8], its effectiveness when applied to the boundary control of fractional wave equation is still unknown.

The initial conditions are chosen as

\[
u(x, 0) = -0.5 \sin(0.5\pi x),
\]

\[
u_t(x, 0) = 0.
\]

First, we choose \( k_d = 1 \) and study the response for \( \alpha = 1.25, 1.50, 1.75, 2.00 \).
The tip end movement over time is shown in Fig. 6.1.

We can see that controller (6.8) still stabilizes the fractional wave equation. The fixed controller gain \( k_d = 1 \) provides the shortest convergence time on the system with \( \alpha = 2 \). The response for \( \alpha = 2 \) (wave equation) becomes zero for \( t > 2 \), an already well-known result [66].

Next, for a fixed \( \alpha = 1.5 \), we study the response for different controller gains \( k_d = 0.25, 0.50, 1.00, 2.00 \), among which \( k_d = 1.00 \) is already studied.

The tip end movement over time is shown in Fig. 6.2.

We can see that when \( k_d \) increases from 0.25 to 2.00, the system changes from underdamped to overdamped, because \( k_d \) is a derivative gain. Once again, the simulation results show that controller (6.8) still works for the boundary control of the fractional wave equation.

### 6.3 Disturbance Rejection

In this section, a disturbance force \( n(t) \) is added at the same point where the boundary control signal enters. It is further assumed that \( n(t) \) is a sinusoidal disturbance signal with unknown amplitude and phase but with a known frequency \( \omega \). Together with the boundary
control signal \([4]\) \((k_d + \frac{ks}{s^2 + \omega^2})\hat{u}_t(1, s)\), the overall applied boundary force is given by

\[
\hat{f}(s) = (k_d + \frac{ks}{s^2 + \omega^2})\hat{u}_t(1, s) + \hat{n}(s),
\]

where \(\hat{f}(s)\) is the Laplace transform of the combination of boundary control force and disturbance force \(n(t)\); \(\hat{n}(s)\) is the Laplace transform of \(n(t)\); \(\hat{u}_t(1, s)\) is the Laplace transform of the velocity of the free end; \(k_d\) and \(k\) are the control gains.

The above boundary controller (6.11) was proposed in [4] to reject the noise for the boundary control of wave equation. The effectiveness of (6.11) when applied to the boundary control of the fractional wave equation is still unknown. Here, again, we use simulation results to verify the feasibility. In the simulation, the disturbance \(n(t)\) is chosen as

\[
n(t) = \sin(10t).
\]

The first example to be simulated is the response for \(k_d = 1\) and \(k = 0\), i.e., the dynamic part of the controller is turned off, for \(\alpha = 1.25, 1.50, 1.75, 2.00\).

The tip end movement over time is shown in Fig. 6.3.

We can see that although the performance is severely degraded with the presence of the noise for all \(\alpha\), the smaller \(\alpha\) is, the less the response is affected by the noise. This is because the lower order derivatives help reduce the level of noise [11].
Next we will study the response for $k_d = 1$ and $k = 10$. The tip end movement over time is shown in Fig. 6.4.

Simulation results show that controller (6.11) still applies to the boundary control of the fractional wave equation. Since the lower order derivatives reduce the affect of the noise, better performance than in the wave equation case can be obtained.

6.4 Boundary Control of the Fractional Wave Equation: Fractional Order Boundary Controllers

In this section, the fractional order boundary controller will be applied to the boundary control of the fractional wave equation. The objective is to study the feasibility of the fractional order boundary controller and make a performance comparison between the integer order controller and the fractional order controller.

6.4.1 The Fractional Order Controller

The problem formulation is the same as in Section 6.1 except that the following fractional order boundary controller is used.

\[
f(t) = -k \frac{d^\mu u(t)}{dt^\mu}, \quad 0 < \mu \leq 1
\]  

(6.13)
where $k$ is the controller gain, $\mu$ is the order of fractional derivative of the displacement at the free end of the cable.

When $\mu = 1$, the controller (6.13) is called integer order controller and has been widely used in the boundary control of wave equations and beam equations [3, 7, 8]. The effectiveness has also been verified when applied to the boundary control of fractional wave equation in [67]. When $0 < \mu < 1$, can controller (6.13) stabilize the system? What advantages does a fractional order controller have over integer order controllers? These are the questions this chapter tries to answer.

### 6.4.2 Stability Studies

To test if fractional order boundary controllers can be used to stabilize the fractional wave equation, the following three different systems were simulated:

- **Case 1:** $\alpha = 1.1$, $k = 0.1$, $\mu = 0.5$,
- **Case 2:** $\alpha = 1.5$, $k = 0.1$, $\mu = 0.7$,
- **Case 3:** $\alpha = 1.9$, $k = 0.2$, $\mu = 0.9$.
All cases have the same initial conditions

\[ u_0(x) = -\sin(0.5\pi x), \quad v_0(x) = 0. \quad (6.14) \]

In Case 3, \( k = 0.2 \) were chosen rather than \( k = 0.1 \), because when the fractional wave equation is closer to the wave equation (\( \alpha \to 1 \)), the response tends to oscillate and needs a larger damping to stabilize quicker.

The descriptions of the simulation results are summarized as follows:

- For \( \alpha = 1.1 \), the displacement of the free end is shown in Fig. 6.5.
- For \( \alpha = 1.5 \), the displacement of the free end is shown in Fig. 6.6.
- For \( \alpha = 1.9 \), the displacement of the free end is shown in Fig. 6.7.

![Fig. 6.5: Displacement of the free end, \( \alpha = 1.1 \)](image)

The simulation results show that all simulated boundary controllers can stabilize the systems. It is also shown that smaller \( \mu \) means smaller overshoot and longer rise time, and vice versa.

### 6.4.3 Performance Comparison

In this section, the performance of fractional order boundary controllers and integer order boundary controllers will be compared, which can only be achieved when the optimal
Fig. 6.6: Displacement of the free end, $\alpha = 1.5$

Fig. 6.7: Displacement of the free end, $\alpha = 1.9$
fractional order controller is compared with the optimal integer order controller. The following objective function, equivalent to comparing the settling time which is easier to implement, is defined:

For integer order boundary controllers \((\mu = 1)\), we seek the best gain \(k\) to

\[
\min_k J(k) = \max(\text{abs}((u(1, t))), \quad t \in [t_f - T, t_f]) \quad (6.15)
\]

Subject to: \(k > 0\). For fractional order boundary controllers \((0 < \mu \leq 1)\), the task to find the best gain \(k\) and the fractional order \(\mu\) to

\[
\min_{k, \mu} J(k, \mu) = \max(u(1, t)), \quad t \in [t_f - T, t_f] \quad (6.16)
\]

Subject to: \(k > 0\) and \(0 < \mu \leq 1\). In the above optimization tasks, \(u(1, t)\) is the displacement of the free end of the cable; \(t_f\) is the total time of simulation; \(T\) is the time period to optimize within the time interval \([t_f - T, t_f]\) which is determined by trial-and-error.

The optimization program chosen is \texttt{SolvOpt} (see [69]), a free program for local non-linear optimization problems.

Next, the performance comparison between the fractional order boundary controller and the integer order boundary controller will be studied.

First, the optimal fractional boundary controller applied to the classical wave equation \((\alpha = 1)\) will be obtained using the initial condition (6.14). \(k\) and \(\mu\) are initialized as \(k = 1\) and \(\mu = 0.5\). The optimal values of \(k\) and \(\mu\) turn out to be \(k^* = 1\), \(\mu^* = 1\), which means the integer order boundary controller achieves the best performance. The displacement of the free end and the whole cable are shown in Fig.6.8 and Fig.6.9, respectively.

We can see that the response becomes zero for \(t > 2\), an already well-known result [66].

Is the integer order boundary controller always better than the fractional order boundary controller? Let us try the fractional wave equation with \(\alpha = 1.5\). The optimal fractional order controller turns out to be \(k^* = 0.7608\) and \(\mu^* = 0.9275\). The optimal integer order boundary controller is with gain \(k^* = 1.1453\).
Fig. 6.8: Displacement of the free end, $\alpha = 1$, $k = 1$, $\mu = 1$

Fig. 6.9: Displacement of the whole cable, $\alpha = 1$, $k = 1$, $\mu = 1$
The comparison of the free end displacement between the optimal fractional order boundary controller and optimal integer order boundary controller is shown in Fig. 6.10. From Fig. 6.10 we can see that the response to the optimal fractional order boundary controller not only has shorter rise time and settling time, but also has no overshoot.

![Figure 6.10: Comparison between two optimal boundary controllers for $\alpha = 1.5$.](image)

Finally, we study the case of $\alpha = 1.1$, in which the fractional wave is much closer to the diffusion equation than to the wave equation. The optimal fractional order controller is $k^* = 0.2455$, $\mu^* = 0.8882$ and optimal integer order controller is $k^* = 0.6787$. The comparison of the free end displacement is shown in Fig. 6.11. We can see that the optimal fractional order boundary controller is again much better than the optimal integer order boundary controller.

### 6.5 Robustness of the Fractional Order Controller Against Noises

The performance advantages of the fractional order control over the integer order controller are shown in Sec. 6.4.3. In this section, we will investigate the robustness of the fractional order controller compared with the integer order controller, specifically, the robustness against the measurement noises. For some practical implementations that the
Table 6.1: Responsiveness and robustness comparison between fractional order controller and integer order controller

<table>
<thead>
<tr>
<th>µ</th>
<th>response time</th>
<th>vibration magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.68s</td>
<td>0.06</td>
</tr>
<tr>
<td>0.85</td>
<td>13.30s</td>
<td>0.04</td>
</tr>
<tr>
<td>0.7</td>
<td>59.32s</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Tip end velocity is calculated with finite differentiation, the possible high level of measurement noises will reduce the accuracy of the calculated velocity, which may make the control performance unacceptable. In these situations, the lower order fractional derivative is expected to reduce the noise level (see [11]).

In the following simulations, the measurement of \( u(1, t) \) is added with a five Hz sinusoidal noise

\[
u_s = 0.1 \cos(10\pi t)\] (6.17)

Other parameters in the simulations are choose as

\[\alpha = 1.75, \quad k = 1\]

Three different derivative control laws, \( \mu = 1, \mu = 0.85, \) and \( \mu = 0.7 \) are tested. The tip end displacements for the three cases are shown in Fig. 6.12, Fig. 6.13, and Fig. 6.14, respectively. We can see that although the response time is slower, the fractional order controllers do have robustness advantages over the integer order controller. The exact comparison of response time (defined here as time to reach \( u = -0.02 \)) and magnitude of vibration after \( t = 70 sec \) is shown in Tbl. 6.1.

An interesting phenomenon occurs if we re-make the above robustness comparisons for the standard wave equation. Although the fractional order controller is still more robust than the integer order controller, the advantage can not be observed from the plots of the tip end displacement, as shown in Fig. 6.15. Rather, the advantages are shown in the 3D plots of displacement of the whole string, Fig. 6.16 and Fig. 6.17, for cases of \( \mu = 1 \) and \( \mu = 0.7 \), respectively.
Fig. 6.11: Comparison between two optimal boundary controllers for $\alpha = 1.1$

Fig. 6.12: Tip end displacement, $\alpha = 1.75$, $\mu = 1$
Fig. 6.13: Tip end displacement, $\alpha = 1.75$, $\mu = 0.85$

Fig. 6.14: Tip end displacement, $\alpha = 1.75$, $\mu = 0.7$
Fig. 6.15: Tip end displacement for $\mu = 1$ and $\mu = 0.7$

Fig. 6.16: Displacement of whole string, $\alpha = 2$, $\mu = 1$
Fig. 6.17: Displacement of whole string, $\alpha = 2$, $\mu = 0.7$
Chapter 7

Boundary Control of Fractional Wave Equations with Delays

Similar in Chapter 6, in this chapter, the boundary control of a “special” cable governed by the fractional wave equation will be studied, except that a time delay is introduced in the feedback loop.

It will be shown that a small delay can destabilize the overall system. The Smith predictor will then be used to compensate the effect of the time delay. Furthermore, a fractional order boundary controller is studied for possible improvement in the transient response performance.

The mathematical analysis will show that different from the conservative wave equation, boundary control of the fractional wave equation is robust again a small enough time delay in the feedback loop. Similarly, the Smith predictor is robust against a small enough difference between the assumed delay and actual delay.

7.1 Problem Formulation

Consider a “special” cable, possibly made with special smart materials, governed by the fractional wave equation, fixed at one end, and stabilized by a boundary controller at the other end. The system can be represented by

\[
\frac{\partial^\alpha u(x, t)}{\partial t^\alpha} = \frac{\partial^2 u(x, t)}{\partial x^2}, \quad 1 < \alpha \leq 2, \quad x \in [0, 1], \quad t \geq 0 \tag{7.1}
\]

\[
u(0, t) = 0, \tag{7.2}
\]

\[
u_x(1, t) = f(t), \tag{7.3}
\]

\[
u(x, 0) = u_0(x), \tag{7.4}
\]

\[
u_t(x, 0) = v_0(x), \tag{7.5}
\]
where \( u(x, t) \) is the displacement of the cable at \( x \in [0, 1] \) and \( t \geq 0 \), \( f(t) \) is the boundary control force at the free end of the cable, \( u_0(x) \) and \( v_0(x) \) are the initial conditions of displacement and velocity, respectively.

The control objective is to stabilize \( u(x, t) \), given the initial conditions (7.4) and (7.5) by using a proper boundary control law using boundary measurements \( f(t) = f(u(1, t), u_t(1, t), \ldots) \).

In Chapter 6, it is shown that following boundary controller can be used to stabilize the fractional wave equation (7.1),

\[
f(t) = -ku_t(1, t),
\]

where \( k \) is the boundary controller gain.

In practice, the time delay is unavoidable in the feedback loop, due to measurement lags and computational lags. In this case, the boundary controller (7.6) becomes

\[
f(t) = -ku_t(1, t - \theta),
\]

where \( \theta \) is the time delay. As shown in Sec. 7.2, the boundary controller (7.7) will not be able to stabilize the overall boundary control system. In this section, the Smith predictor will be applied to compensate the time delay.

Another objective of this section is to study whether a fractional order boundary controller can improve the boundary control performance. The fractional order boundary controller is expressed as follows:

\[
f(t) = -k\frac{d^\mu}{dt^\mu}u(1, t - \theta), \quad \mu \in (0, 1].
\]

### 7.2 Time Delay Effect and Its Compensation: Integer Boundary Control

In this section, some simulation results will be presented to illustrate the time delay effect in boundary control and fractional order wave equation. The Smith predictor scheme is then applied to show the time delay compensation.

All the simulation examples in this paper have the same initial conditions

\[
u_0(x) = -\sin(0.5\pi x), \quad v_0(x) = 0.
\]
7.2.1 Destabilizing Effect of Time Delay

Choose $\alpha = 1.75$ and $k = 1$. If there is no time delay ($\theta = 0$), the displacement of the whole cable and of the tip end are plotted in Fig. 7.1 and Fig. 7.2, respectively.

When $\theta = 0.5$ sec. is introduced, the displacement of the whole cable and of the tip end are plotted in Fig. 7.3 and Fig. 7.4, respectively. We can see that the time delay makes the originally stable system unstable.

Fig. 7.1: Displacement of the whole cable, $k = 1$, $\alpha = 1.75$, $\theta = 0$, $\mu = 1$.

Fig. 7.2: Displacement of the tip end, $k = 1$, $\alpha = 1.75$, $\theta = 0$, $\mu = 1$. 
Fig. 7.3: Displacement of the whole cable, $k = 1, \alpha = 1.75, \theta = 0.5$ sec., $\mu = 1$.

Fig. 7.4: Displacement of the tip end, $k = 1, \alpha = 1.75, \theta = 0.5$ sec., $\mu = 1$. 
7.2.2 Results Using the Smith Predictor

After the introduction of the Smith predictor, the displacement of the whole cable and of the tip end for $\alpha = 1.75$ are plotted in Fig. 7.5 and Fig. 7.6, respectively. We can see that the Smith predictor effectively compensates the time delay and makes the system stable.

Fig. 7.5: Displacement of the whole cable, $k = 1, \alpha = 1.75, \theta = 0.5 \text{ sec.}, \mu = 1$ with the Smith Predictor.

Fig. 7.6: Displacement of the tip end, $k = 1, \alpha = 1.75, \theta = 0.5 \text{ sec.}, \mu = 1$ with the Smith Predictor.

7.3 Integer Order Controller vs. Fractional Order Controller
A fractional order wave equation naturally reminds us of the fractional order boundary controller. The objective of this section is to compare the performance of fractional order boundary controllers and integer order boundary controllers. The conclusion can only be drawn when the optimized fractional order controller is compared with the optimized integer order controller under the same optimization criterion. Here, we define the following objective function, equivalent to comparing the settling time but it is much easier to implement. Of course, other types of objective functions could be used.

For integer order boundary controllers ($\mu = 1$), we seek the best gain $k$ to

$$
\min_k J(k) = \max(|u(1,t)|), \quad t \in [t_f - T, t_f]
$$

(7.10)

Subject to: $k > 0$.

For fractional order boundary controllers ($0 < \mu \leq 1$), the task is to find the best gain $k$ and the fractional order $\mu$ to

$$
\min_{k,\mu} J(k,\mu) = \max(|u(1,t)|), \quad t \in [t_f - T, t_f]
$$

(7.11)

Subject to: $k > 0$ and $0 < \mu \leq 1$.

In the above optimization tasks, $u(1,t)$ is the displacement of the free end of the cable; $t_f$ is the total time of simulation; $T$ is the time period to optimize within the time interval $[t_f - T, t_f]$ which is determined by trial-and-error.

The optimization program chosen is SolvOpt [69], a free program for local nonlinear optimization problems.

In the first example, we choose $\alpha = 1.75$ and $\theta = 0.5$. The optimal gain and the optimal fractional order derivative of the optimal fractional order controller are $k^* = 0.9385$ and $\mu^* = 0.9023$, respectively. The optimal gain of the optimal integer order controller is $k^* = 0.7037$. The comparison of the free end displacement between the optimal fractional order boundary controller and optimal integer order boundary controller is shown in Fig. 7.7. We can see that the response to the optimal fractional order boundary controller not only has a shorter settling time, but also has a smaller overshoot.
In the second example, we choose $\alpha = 1.25$ and $\theta = 0.5$ sec. The optimal gain and the optimal fractional order derivative of the optimal fractional order controller are $k^* = 0.3048$ and $\mu^* = 0.8469$, respectively. The optimal gain of the optimal integer order controller is $k^* = 0.7926$. The comparison of the free end displacement between the optimal fractional order boundary controller and optimal integer order boundary controller is shown in Fig. 7.8. We can see that the response under the optimal fractional order boundary control settles faster.
7.4 Robustness Issue

In this section, two robust stabilization problems of the fractional wave equations subject to delays will be studied. First, under what conditions a very small delay in boundary measurement will not cause instability problems, so that the application of the Smith predictor is not necessary. Second, how to stabilize the system when the delay is large enough and makes the system unstable.

7.4.1 Problem Formulation

Consider a cable made with special smart materials governed by the fractional wave equation, fixed at one end, and stabilized by a boundary controller at the other end. Omitting the mass of the cable, the system can be represented by

\[
\frac{\partial^\alpha u}{\partial t^\alpha} = \frac{\partial^2 u}{\partial x^2}, \quad 1 < \alpha \leq 2, \quad x \in [0, 1], \quad t \geq 0
\] (7.12)

\[
u(0, t) = 0,
\]

\[
u_x(1, t) = f(t),
\]

\[
u(x, 0) = u_0(x),
\]

\[
u_t(x, 0) = v_0(x),
\]

where \(u(x, t)\) is the displacement of the cable at \(x \in [0, 1]\) and \(t \geq 0\), \(f(t)\) is the boundary control force at the free end of the cable, \(u_0(x)\) and \(v_0(x)\) are the initial conditions of displacement and velocity, respectively. The control objective is to stabilize \(u(x, t)\), given the initial conditions (7.15) and (7.16).

In this section, the robustness of the controllers in the following format will be studied:

\[
f(t) = -k \frac{d^\mu u(1, t)}{dt^\mu}, \quad 0 < \mu \leq 1
\] (7.17)

where \(k\) is the controller gain, \(\mu\) is the order of fractional derivative of the displacement at the free end of the cable.
Using the method presented in Chapter 2, the Laplace transform of \( u(x, t) \), \( U(x, s) \), can be obtained. Substituting \( x = 1 \) into \( U(x, s) \) and divide \( U(x, s) \) by \( F(s) \), we obtain the following transfer function of the fractional wave equation \( P(s) \):

\[
P(s) = \frac{U(1, s)}{F(s)} = \frac{1 - e^{-2s^\frac{\alpha}{2}}}{s^\frac{\alpha}{2} \left(1 + e^{-2s^\frac{\alpha}{2}}\right)}.
\]  

(7.18)

### 7.4.2 Small Delay Case

We consider the presence of a very small time delay \( \theta \) in boundary measurement, shown as follows

\[ f(t) = -ku_t^{(\mu)}(1, t - \theta), \]  

(7.19)

where \( \theta \) is the time delay.

The situation is also illustrated in Fig.7.9, where \( P(s) \) is the transfer function of the plant and \( C(s) \) is the Laplace transform of the controller. In our case, \( P(s) \) is (7.18) and \( C(s) \) is

\[
C(s) = k s^\mu
\]  

(7.20)

![Fig. 7.9: A feedback control system with a time delay](image)

In [10,12–14], it was shown that an arbitrarily small delay in boundary measurement causes the instability problem in boundary control of wave equations using integer order controllers \( f(t) = -ku_t(1, t) \). Does this problem exist in boundary control of the fractional wave equation? Since fractional order controllers are chosen in this chapter, will this additional tuning knob bring us any benefits of robustness against the small delay?

CLAIM:
If the derivative order $\mu$ of controller (7.17) and the fractional order $\alpha$ in the fractional
wave equation (7.12) satisfy
\[ \mu < \frac{\alpha}{2}, \] (7.21)
then the system is stable for a small enough delay $\theta$ in boundary measurement.

**Proof:**

For $s \in \mathbb{C}_0$,
\[
|H(s)| = |C(s)P(s)| = \left| \frac{ks^\mu(1 - e^{-2s^{\frac{\alpha}{2}}})}{s^{\alpha} \left(1 + e^{-2s^{\frac{\alpha}{2}}}\right)} \right|
\]
\[
= \left| \frac{k(1 - e^{-2s^{\frac{\alpha}{2}}})}{s^{(\frac{\alpha}{2} - \mu)} \left(1 + e^{-2s^{\frac{\alpha}{2}}}\right)} \right|
\]
\[
= \frac{k|1 - e^{-2s^{\frac{\alpha}{2}}}|}{s^{(\frac{\alpha}{2} - \mu)}|1 + e^{-2s^{\frac{\alpha}{2}}}|}
\]
Since $\frac{\alpha}{2} > \mu$, $|s^{(\frac{\alpha}{2} - \mu)}| \to \infty$ for $|s| \to \infty$.

Since $\frac{1}{2} < \frac{\alpha}{2} < 1$, for $|s|$ large enough, $|1 - e^{-2s^{\frac{\alpha}{2}}}|$ is bounded and $|1 - e^{-2s^{\frac{\alpha}{2}}}| > \eta > 0$, where $\eta$ is a positive number.

So
\[
\limsup_{|s| \to \infty, s \in \mathbb{C}_0} |H(s)| = 0 < 1.
\]
Following the above proof, it can be easily proved that an integer order controller $f(t) = -ku_t(1,t)$ is not robust against an arbitrarily small delay.

### 7.4.3 Compensation of Large Delays Using the Smith Predictor

In the last section, it is shown that an fractional order controller is robust against a small delay under the condition (7.21). In this section, we investigate the problem that what if the delay is large and makes the system unstable? We will apply the Smith predictor to solve this problem.

In Sec. 3.1, it is shown that if the assumed delay is equal to the actual delay, the Smith predictor removes the delay term completely from the denominator of the closed-loop. However, the actual delay is not exactly known. In this section, we will investigate
what if an unknown small difference \( \epsilon \) between the assumed delay and the actual delay is introduced to the system, as shown in Fig. 7.10.

\[
\begin{align*}
\text{CLAIM:} \\
\text{If } \hat{\theta} \text{ is chosen as the minimum value of the possible delay and } \mu \text{ is chosen to satisfy (7.21), then the controller (3.19) is robust against a small difference } \epsilon \text{ between the assumed delay } \hat{\theta} \text{ and the actual delay } \theta = \hat{\theta} + \epsilon. \\
\text{Proof:} \\
\text{For } s \in \mathbb{C}_0,
\end{align*}
\]

\[
|H(s)| = \left| \frac{ks^\mu P(s)e^{-\hat{\theta}s}}{1 + ks^\mu P(s)(1 - e^{-\hat{\theta}s})} \right| \\
< \frac{k|1 - e^{-2s\frac{\mu}{2}}||e^{-\theta s}|}{|s^{(\frac{\mu}{2} - \mu)}(1 + e^{-2s\frac{\mu}{2}}) - k(1 - e^{-2s\frac{\mu}{2}})(1 - e^{-\theta s})|} \\
\text{When } |s| \to \infty,
\]

\[
|s^{(\frac{\mu}{2} - \mu)}(1 + e^{-2s\frac{\mu}{2}})| \to \infty,
\]

while both \( |1 - e^{-2s\frac{\mu}{2}}| \) and \( |(1 - e^{-2s\frac{\mu}{2}})(1 - e^{-\theta s})| \) are bounded.

So

\[
\limsup_{|s| \to \infty, s \in \mathbb{C}_0} |H(s)| = 0 < 1.
\]

\text{Remarks:}

In \textit{Theorem 1}, \( \epsilon \) is positive. To satisfy this condition, \( \hat{\theta} \) should be chosen as the minimal value of the possible delay.
Chapter 8

Identification of a Fractional Linear Wave Equation

In this chapter, using the simulation method presented in Chapter 2, an identification method is developed to identify the unknown wave constant, fractional order and initial profile of a fractional order diffusion-wave equation, based on boundary measurements possibly corrupted with measurement noise.

8.1 Problem Formulation

Consider a cable made from special materials, with one end fixed and the other end free, governed by the following fractional diffusion-wave equation

\[
\frac{\partial^\alpha u(x,t)}{\partial t^\alpha} = b^2 \frac{\partial^2 u(x,t)}{\partial x^2},
\]

\[1 < \alpha < 2, \ x \in [0,1], \ t \geq 0, \] (8.1)

where \( u(x,t) \) is the displacement of the cable at \( x \in [0,1] \) and \( t \geq 0 \), \( \alpha \) is the parameter describing the order of the fractional derivative, \( b \) is a constant decided by the tension and the mass per unit length of the cable. Clearly, this is a special type of fractional order PDE system in-between the diffusion equation (\( \alpha = 1 \)) and the wave equation (\( \alpha = 2 \)).

Equation (8.1) is subject to the following initial and boundary conditions

\[ u(0,t) = 0, \] (8.2)

\[ u_x(1,t) = f(t), \] (8.3)

\[ u(x,0) = u_0(x), \] (8.4)

\[ u_t(x,0) = v_0(x), \] (8.5)

where \( f(t) \) is the boundary control force at the free end, \( u_0(x) \) and \( v_0(x) \) are the initial displacement condition (initial boundary profile) and the initial velocity profile, respectively.
Assuming that the values of $\alpha$ and $b$ are not exactly known and need to be estimated. Furthermore, initial profiles $u_0(x)$ and $v_0(x)$ may not be exactly known. It is also assumed that the displacement of the free end $u(1, t)$ can be measured for the identification task.

8.2 The Proposed Identification Method

For simplicity of our presentation and for practical reasons, we consider a simplified scenario. That is, with the initial shape $u_0(x)$, not exactly known, initial velocity $v_0(x) = 0$ (cable is initially at rest), and no boundary force $f(t) = 0$, we can measure the displacement of the free end $u(1, t)$ as the system output measurement data for identification.

If equations (8.1)-(8.5) can be solved, we can estimate $\alpha$ and $b$ through an optimization program to make the solution fit the measurement data as closely as possible.

There are following problems with the above idea.

1. How to solve equations (8.1)-(8.5)?

2. Since we are designing the parameter estimation algorithm via simulation, how to generate the measurement data?

3. If we want to do a real hands-on experiment rather than generating the “measured data” via simulation, it is hard to make the initial shape $u(x, 0)$ exactly as desired, i.e., the actual initial shape is not exactly known.

The first two problems are very closely related. The first problem can be solved by numerically solving equations (8.1)-(8.5), since the analytical solution is still an unsolved problem. Using the method presented in Chapter 2, the fractional wave equation can be solved, effective even if $f(t)$, a boundary feedback controller, is included. The solution plus Gaussian noise, which is unavoidable in the actual experiments, can be used as the measured data. This is how we solve the second problem. It is illustrated below how to solve (8.1)-(8.5), assuming $\alpha = 1.75$, $b = 0.5$, $f(x) = 0$, and the initial conditions

$$u_0(x) = -\frac{1}{2} \sin\left(\frac{1}{2} \pi x\right), \quad (8.6)$$
Taking the Laplace transform of (8.1)- (8.3) with respect to \( t \), we obtain the following ODE (Ordinary Differential Equation) and boundary conditions

\[
\frac{d^2 U(x, s)}{dx^2} - 4s^7 U(x, s) = 2s^\frac{3}{2} \sin\left(\frac{1}{2} \pi x\right) \quad (8.8)
\]

\[
U(0, s) = 0 \quad (8.9)
\]

\[
\left. \frac{dU(x, s)}{dx} \right|_{x=1} = 0 \quad (8.10)
\]

where \( U(x, s) \) is the Laplace transform of \( u(x, t) \).

Solving (8.8), we have

\[
U(x, s) = e^{-2s^\frac{7}{8}x} C_1 + e^{2s^\frac{7}{8}x} C_2 - 8 s^{3/4} \sin\left(\frac{1}{2} \pi x\right) \frac{\sin \left(\frac{1}{2} \pi x\right)}{16 s^7 + \pi^2} \quad (8.11)
\]

where \( C_1 \) and \( C_2 \) are arbitrary constants. By taking derivative of (8.11) with respect to \( x \), we have

\[
\frac{dU(x, s)}{dx} = -2s^\frac{7}{8} e^{-2s^\frac{7}{8}x} C_1 + 2s^\frac{7}{8} e^{2s^\frac{7}{8}x} C_2 - 4s^\frac{3}{4} \cos \left(\frac{1}{2} \pi x\right) \frac{\pi}{16 s^7 + \pi^2} . \quad (8.12)
\]

Substituting (8.11) and (8.12) into (8.9) and (8.10), respectively, yields

\[
C_1 + C_2 = 0, \quad (8.13)
\]

\[
-e^{-2s^\frac{7}{8}} C_1 + e^{2s^\frac{7}{8}} C_2 = 0. \quad (8.14)
\]

Solving (8.13) and (8.14) simultaneously, we obtain

\[
C_1 = C_2 = 0. \quad (8.15)
\]

So, finally,

\[
U(x, s) = -8 s^{3/4} \sin \left(\frac{1}{2} \pi x\right) \frac{\sin \left(\frac{1}{2} \pi x\right)}{16 s^7 + \pi^2} . \quad (8.16)
\]
So far, all the calculation steps illustrated above can be automated via computer symbolic algebra, such as Matlab Symbolic Math Toolbox [20]. Now, \( u(x,t) \) can be obtained by taking the numerical inverse Laplace transform of (8.16), as presented in Chapter 2.

We can solve the third problem by treating the initial shape \( u_0(x) \) as the extra parameters to be estimated such that the parameter estimation algorithm does not depend on the exact knowledge of \( u_0(x) \). Specifically, we assume the initial shape to be parameterized by the following polynomial, which equals to zero at \( x = 0 \),

\[
\tilde{u}_0(x) = \sum_{n=1}^{N} a_n x^n
\]  

(8.17)

where \( a_i \) is the parameter to be estimated. By increasing \( N \), we expect that the estimated initial shape \( \tilde{u}_0(x) \) will converge to \( u_0(x) \), the real initial shape.

Now the parameter estimation problem can be formulate as the following nonlinear programming problem

\[
\min_{a_0, \ldots, a_N, \tilde{a}, \tilde{b}} J(a_0, \ldots, a_N, \tilde{a}, \tilde{b}) = 
\min_{a_0, \ldots, a_N, \tilde{a}, \tilde{b}} \sum_{n=0}^{N_s-1} (u(1, n\Delta t) - \tilde{u}(1, n\Delta t))^2,
\]  

(8.18)

where \( a_0, \ldots, a_N, \tilde{a}, \tilde{b} \) are the parameters to be estimated; \( u(1, n\Delta t) \) are the measured boundary response data at time \( n\Delta t \) with the sampling time \( \Delta t \); \( \tilde{u}(1, n\Delta t) \) are the solution to (8.1)-(8.5) based on parameters \( a_0, \ldots, a_N, \tilde{a}, \tilde{b} \) and \( N_s \) is the total number of samples.

At this step, the identification problem has been converted to a numerical optimization problem which can be solved by various existing optimization codes. The optimization program we chose for this study is \texttt{SolvOpt} [69], a free program for local nonlinear optimization problems.

Another source of errors in this algorithm is from the mismatched time between the measured data and the numerical solution. In (8.18), \( u(1, n\Delta t) \) is desired to be measured at \( t = n\Delta t \). However, due to various reasons in practice, especially the inaccuracy of the starting time, the actual time at which \( u(1, t) \) is sampled can be slightly different from the
desired time instant $n\Delta t$. We simulated the effect of this time mismatch problem by using
the following objective function, a slightly modified one from (8.18):

$$
\min J(a_0, \ldots, a_N, \tilde{\alpha}, \tilde{\beta}) = \sum_{n=0}^{N_s-k-1} (u(1, (n+k)\Delta t) - \tilde{u}(1, n\Delta t))^2,
$$
(8.19)
i.e., the simulated mismatched time is $k\Delta t$.

8.3 Simulation Results for Algorithm Validation

To generate the simulated measurement data, the following parameters and initial
conditions are used

$$
\alpha = 1.75, \quad b = 0.25, \quad \Delta t = 0.078s, \quad N_s = 512
$$

$$
u_0(x) = -\frac{1}{2} \sin\left(\frac{1}{2} \pi x\right)
(8.20)$$

$$
v_0(x) = 0.
$$

The corresponding solution for $u(x, t)$ obtained from (8.1)-(8.5) is plotted in Fig. 8.1.
From Fig. 8.1 we can see that the fractional diffusion-wave equation have mixed properties
of the diffusion equation, i.e., damped solution, and of the wave equation, i.e., oscillated
solution.

Fig. 8.1: Displacement of the whole cable $u(x, t)$
To simulate the practical measured signal, Gaussian noise with $SNR = 26dB$ is added to the displacement of the free end and shown in Fig. 8.2.

![Figure 8.2: Boundary measurement data with Gaussian noise added, $SNR = 20dB$](image)

We choose $N = 3$ and the parameters to be estimated are initialized in optimization as follows

$$a_0 = -0.79, \ a_1 = 0.07, \ a_2 = 0.21,$$  \hspace{1cm} (8.21)

and

$$\tilde{a} = 1.5, \ \tilde{b} = 1.$$  \hspace{1cm} (8.22)

In (8.21), $a_0, \ a_1, \ \text{and} \ a_2$ are initialized to make the polynomial (8.17) close to (8.20). This is reasonable since the actual initial profile is actually roughly known.

We used different values of $k$ to simulate different amount of mismatched time. After the optimization process, the estimated values of $\alpha$ and $b$ are shown in Table 8.1. The estimated initial shapes are plotted in Fig. 8.3.

We can see that the unknown system parameters and the initial profile have been successfully estimated. As expected, smaller mismatched time generates more accurate results. However, for relatively large mismatched time, the estimation accuracy is still satisfactory. In the sequel, we only report the results for noisy measurement cases if not otherwise stated.
Table 8.1: Estimated parameters, $\alpha = 1.754$, $b = 0.25$, $N = 3$

<table>
<thead>
<tr>
<th>$k$</th>
<th>time mismatch</th>
<th>$\bar{\alpha}$ and rel. error</th>
<th>$b$ and rel. error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0s</td>
<td>1.7520, 0.11%</td>
<td>0.2497, 0.12%</td>
</tr>
<tr>
<td>1</td>
<td>0.078s</td>
<td>1.7557, 0.32%</td>
<td>0.2504, 0.16%</td>
</tr>
<tr>
<td>2</td>
<td>0.156s</td>
<td>1.7582, 0.46%</td>
<td>0.2513, 0.52%</td>
</tr>
<tr>
<td>5</td>
<td>0.390s</td>
<td>1.7655, 0.88%</td>
<td>0.2539, 1.56%</td>
</tr>
</tbody>
</table>

Fig. 8.3: Actual initial shape and estimated initial shape, $\alpha = 1.75$
8.4 Simulation Studies on Two Extreme Cases

In this section, we study two extreme cases, i.e., when $\alpha$ is close to 1 and when $\alpha = 2$. In many existing schemes, in extreme cases, the estimation accuracy is usually degraded, or even worse, the algorithm may fail. It is meaningful to check the robustness of our proposed algorithms in these extreme cases.

First let us study the parameter estimation when $\alpha = 1$. In this case, the fractional wave equation is closer to a diffusion equation than to a wave equation. All parameters and initial conditions are the same as in Sec. 8.3 except $\alpha$, the fractional order.

The solution to (8.1)-(8.5), to be taken as measurement data for system identification, is plotted in Fig. 8.4. We can see that the solution is over-damped, close to the solution of the diffusion equation.

![Fig. 8.4: Displacement of the whole cable $u(x, t)$ when $\alpha = 1.01$](image)

The boundary measurement data with Gaussian noise added is plotted in Fig. 8.5. The estimated parameters are shown in Table 8.2. The estimated initial shapes are plotted in Fig. 8.6.

Simulation results show that the algorithm works well even if $\alpha$ is close to 1. However, comparing Tbl. 8.2 and Tbl. 8.1, we can see that the estimation accuracy degraded when $\alpha$ is close to 1. We can see the reason if we study Fig. 8.5, where after $t = 20$sec, $u(1, t)$ is
Fig. 8.5: Boundary measurement data for identification with Gaussian noise added when $\alpha = 1.1$, $SNR = 26dB$

Table 8.2: Estimated parameters when $\alpha = 1.1$, $b = 0.25$, $N = 3$

<table>
<thead>
<tr>
<th>$k$</th>
<th>time mismatch</th>
<th>$\bar{\alpha}$ and $\bar{b}$</th>
<th>$\bar{\alpha}$ and $\bar{b}$</th>
<th>rel. error</th>
<th>rel. error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0s</td>
<td>1.1011, 0.10%</td>
<td>0.2483, 0.68%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.078s</td>
<td>1.1060, 0.54%</td>
<td>0.2461, 1.56%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.156s</td>
<td>1.1064, 0.58%</td>
<td>0.2463, 1.48%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.390s</td>
<td>1.1185, 1.68%</td>
<td>0.2398, 4.08%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 8.6: Actual initial shape and estimated initial shape, $\alpha = 1.1$
almost zero and there is only noise left. So Fig. 8.5 actually contains less useful information than in Fig. 8.2, making the estimation accuracy lower.

Next we study the parameter estimation when $\alpha = 2$, i.e., the fractional wave equation becomes the wave equation.

The solution to (8.1)-(8.5) is plotted in Fig. 8.7. This is the solution to the standard wave equation

The boundary measurement data with Gaussian noise added is plotted in Fig. 8.8. The estimated parameters are shown in Tbl. 8.3 and the estimated initial shapes are plotted in Fig. 8.9.

In the case of $\alpha = 2$, the algorithm works even better than in the case of $\alpha = 1.75$. In Fig. 8.9, the estimated initial shapes are almost identical to the actual initial shape, even if the mismatched time is large. This confirms our reasoning why the estimation

<table>
<thead>
<tr>
<th>$k$</th>
<th>time mismatch</th>
<th>$\tilde{\alpha}$ and rel. error</th>
<th>$b$ and rel. error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0s</td>
<td>2.0001, 0.00%</td>
<td>0.2500, 0.00%</td>
</tr>
<tr>
<td>1</td>
<td>0.078s</td>
<td>2.0002, 0.01%</td>
<td>0.2507, 0.28%</td>
</tr>
<tr>
<td>2</td>
<td>0.156s</td>
<td>2.0004, 0.02%</td>
<td>0.2514, 0.56%</td>
</tr>
<tr>
<td>5</td>
<td>0.390s</td>
<td>2.0006, 1.63%</td>
<td>0.2537, 1.48%</td>
</tr>
</tbody>
</table>
Fig. 8.8: Boundary measurement data with Gaussian noise added when $\alpha = 2$, $SNR = 26dB$

Fig. 8.9: Actual initial shape and estimated initial shape, $\alpha = 2$
results for $\alpha = 1.75$ are better than the results for $\alpha = 1.1$. Since the solution for the standard wave equation oscillates forever without being damped, Fig. 8.8 contains more useful information than in Fig. 8.2, which leads to better estimation results.

It can be concluded that even in the extreme cases, the performance are still satisfactory by using our proposed identification algorithm.

### 8.5 Relationship Between Polynomial Order and Estimation Accuracy

In the previous simulation examples, we use the third order polynomial to estimate the initial profile, which leads to satisfactory estimation results. In this section, we will investigate the relation between the polynomial order and the estimation accuracy. Lower order polynomial results in less parameters to identify and faster computation, which is important for tasks requiring on-line parameter estimation. So lower order polynomial is highly preferred if it generates equally high, or lower yet acceptable, estimation accuracy.

To study only the relationship between the polynomial order and the estimation accuracy, in the following simulations, all parameters and initial conditions are the same as in Sec. 8.3, except that there is no measurement noise and no mismatched time ($k = 0$).

We tested three different cases: $N = 3$, $N = 2$, and $N = 1$. The estimated parameters for each case are listed in Tbl. 8.4. The estimated initial profiles for each case are plotted in Fig. 8.10. Since we assume there is no noise and mismatched time in the simulations, the estimation accuracy in the $N = 3$ case is much higher than in the simulations in previous sections. We can see that the estimation accuracy for $N = 2$ is almost as high as for $N = 3$.

We conclude that unless extremely high estimation accuracy is required, the second order polynomial can replace the third order polynomial to estimate the initial profile. The first order polynomial is also a candidate when the measurement noise is small, the mismatched time is small, and the estimation accuracy requirement is low. We can also conclude that this estimation algorithm is not sensitive to the difference between the actual initial profile and the desired initial profile.
Table 8.4: Estimated parameters when $\alpha = 1.75$, $b = 0.25$, different $N$s

<table>
<thead>
<tr>
<th>$N$</th>
<th>$\alpha$ and rel. error</th>
<th>$b$ and rel. error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3$</td>
<td>$1.7500, 0.00%$</td>
<td>$0.2500, 0.00%$</td>
</tr>
<tr>
<td>$2$</td>
<td>$1.7501, 0.006%$</td>
<td>$0.2500, 0.00%$</td>
</tr>
<tr>
<td>$1$</td>
<td>$1.7543, 0.24%$</td>
<td>$0.2491, 0.36%$</td>
</tr>
</tbody>
</table>

Fig. 8.10: Actual initial shape and estimated initial shape, different $N$s, $\alpha = 1.75$
Chapter 9
Conclusions and Future Work

Several topics are studied in the thesis. The main contributions include the following:

- A new simulation method is developed to simulate control of some typical distributed parameter systems.
- The Smith predictor is applied to the boundary control of distributed parameter systems, including the wave equation and the fractional wave equation.
- The robustness of the Smith predictor is analyzed.
- The performance of the integer order and the fractional order controller is compared to show the advantages of the fractional order controller.
- A simulation platform is developed to simulate control of the diffusion process using mobile actuators and mobile sensors.

Control of distributed parameter systems is a huge topic and this thesis touches only the surface of a few topics. The possible future work following the thesis includes:

- The stability of boundary control of the fractional wave equation is only studied via simulation method developed in the thesis. The rigorous proof is still an unsolved problem.
- The robustness of the Smith predictor applied to the boundary control of the wave equation and the fractional wave equation is proved rigorously in the thesis. However, only the robustness against the difference between the assumed delay and actual delay is considered. For the distributed parameter systems studied in the thesis, the Smith predictor is actually a fractional order controller, which can only be approximated in practice. Will this approximation cause stability problems? This is another aspect of the robustness issue.
References


Appendices
Appendix A
Source Code for Each Chapter

The source code for each chapter is listed in the following. The follow routine [22] will be required by all the simulation code.

```matlab
% NILT numerical inverse Laplace transform
% except from Programs for Fast Numerical Inversion of Laplace
% Transforms in Matlab Language Environment, Lubomir Brancik,
% Konference MATLAB 99 ZCU, Plzen, 1999, pp. 27-39
%
% function [ft, t] = nilt(F, tm)
% % F: file name of the transfer function
% % tm: time range in which to calculate the inverse Laplace transform
% % ft: vector of the numerical value of the inverse Laplace transform
% % t: vector of the time, t(1) = 0, t(end) = tm

function [ft,t]=nilt(F,tm);

alfa=0; M=1024; P=2;
N=2*M; wyn=2*P+1;
t=linspace(0,tm,M);

NT=2*tm*N/(N-2); omega=2*pi/NT;
c=alfa+25/NT; s=c-i*omega*(0:N+wyn-2);
Fsc=feval(F,s);
ft=fft(Fsc(1:N)); ft=ft(1:M);
for n=N:N+wyn-2
    ft(n-N+2,:)=Fsc(n+1)*exp(-i*n*omega*t);
end
ft1=cumsum(ft); ft2=zeros(wyn-1,M);
for I=1:wyn-2
    ft=ft2+1./diff(ft1);
    ft2=ft1(2:wyn-I,:); ft1=ft;
end
ft=ft2+1./diff(ft1); ft=2*real(ft)-Fsc(1);
ft=exp(c*t)/NT.*ft; ft(1)=2*ft(1);

A.1 Source Code for Chapter 3

% wave_smith.m
% WAVE_SMITH calculating the transfer function of boundary control of wave
% equation using Smith predictor and its variants
%
% function U_xs = wave_smith(m, theta_real, theta_guess, ctrl_type,
% % ctrl_para, noise_para, u_0, v_0)
% ```
function U_xs = wave_smith(a, m, theta_real, theta_guess, ctrl_type, ctrl_para, noise_para, u_0, v_0)

  syms x s C1 C2 Ff_s Ff_s_smith k_flt z_flt p_flt tao_flt alfa flt_s Ff_s_smith
  % s: Laplace transform variable
  % C1, C2: undetermined constants in the solution of the fourth order ODE
  % Ff_s: Laplace transform of boundary control force
  % m: tip mass
  % alfa: controller gain, see Francis Conrad and Omer Morgul,
  % On the stabilization of a flexible beam with a tip mass,
  % theta: tip velocity feedback delay

  % first to get the transfer function of the tip
  u_0_bak = u_0;
  v_0_bak = v_0;
  u_0 = 0;
  v_0 = 0;
  u_xxx_0 = diff(u_0, 'x', 3);
  ode = strcat('D2U-(s+a)^2*U','+(', char(sym(s*u_0)),')+(', char(sym(v_0)), ')',' = 0');
  U_ud = simple(dsolve(ode, 'x')); % U(x,s) with undefined constants
  dU_ud = simple(diff(U_ud, 'x', 1)); % first order derivative, with undefined constants
bd1 = strcat(char(subs(U_ud, x, 0)), '0'); % boundary condition \$u(0, t)=0$\nbd2 = subs(dU_ud, x, 1) - Ff_s;
bd2 = strcat(char(bd2), '0');

[C1, C2] = solve(bd1, bd2, 'C1', 'C2');

U_tip_s = subs(subs(U_ud), x, 1);
G_u_tip_s = simple(U_tip_s/Ff_s);
if (findstr('Ff_s', char(G_u_tip_s)))
    error('failed to get transfer function of the tip.');
end
G_v_tip_s = G_u_tip_s*s; % we need transfer function of velocity rather than displacement

% simulate smith predictor
u_0 = u_0_bak;
v_0 = v_0_bak;

ode = strcat('D2U-(s+a)^2*U','+(', char(sym(s*u_0)),')+(', char(sym(v_0)),')', '0');

U_ud = simple(dsolve(ode, 'x')); % U(x,s) with undefined constants
dU_ud = simple(diff(U_ud, 'x', 1)); % first order derivative with undefined constants

bd1 = strcat(char(subs(U_ud, x, 0)), '0'); % boundary condition \$u(0, t)=0$\n
switch ctrl_type,
case 0 % no Smith predictor at all
    d_gain = ctrl_para(1); % static controller gain
    k_gain = ctrl_para(2); % dynamic controller gain
    if k_gain < 1e-10, % if k_gain is zero, freq_ctrl is not
        freq_ctrl = 0; % necessary to be supplied by caller
    else
        freq_ctrl = ctrl_para(3);
    end

    mag_noise = noise_para(1);
    if mag_noise < 1e-10,
        freq_noise = 0;
    else
        freq_noise = noise_para(2);
    end

    bd2 = simple(subs(dU_ud, x, 1) - (-d_gain - k_gain*s/(s^2 + freq_ctrl^2))* ... 
                (s*subs(U_ud, x, 1) - subs(u_0, x, 1))*exp(-s*theta_real) + mag_noise*s/(s^2 + freq_noise^2));
    bd2 = strcat(char(bd2), '0');

[C1, C2] = solve(bd1, bd2, 'C1', 'C2');

end % plain Smith predictor

case 1 % modified Smith predictro with lead-lag filter

d_gain = ctrl_para(1);
Ff_s = -d_gain*((s*subs(U_ud, x, 1)-subs(u_0, x, 1))*exp(-thet
\[ d_{\text{gain}} = \text{ctrl\_para}(1); \]
\[ k_{\text{flt}} = \text{ctrl\_para}(2); \]
\[ z_{\text{flt}} = \text{ctrl\_para}(3); \]
\[ p_{\text{flt}} = \text{ctrl\_para}(4); \]
\[ F_f_s = -d_{\text{gain}}*\text{flt\_s}*((s*\text{subs}(U_{ud}, x, 1)-\text{subs}(u_0, x, 1))*\exp(-\theta_{\text{real}}*s))/F_f_s_{\text{smith}}; \]
\[ bd2 = \text{subs}(dU_{ud}, x, 1) - F_f_s; \]
\[ bd2 = \text{strcat}\left(\text{char}(bd2), '='0'\right); \]
\[ [C1, C2] = \text{solve}(bd1, bd2, 'C1', 'C2'); \]
\[ \text{flt\_s} = k_{\text{flt}}*(s + z_{\text{flt}})/(s + p_{\text{flt}}); \]
\[ F_f_s_{\text{smith}} = 1 + d_{\text{gain}}*G_v_{\text{tip\_s}}*(1-\text{flt\_s}*\exp(-\theta_{\text{guess}}*s)); \]
\[ \text{case 3 } \% \text{ modified Smith predictor with time advance approximator} \]
\[ \alpha = \text{ctrl\_para}(1); \]
\[ k_{\text{flt}} = \text{ctrl\_para}(2); \]
\[ tao_{\text{flt}} = \text{ctrl\_para}(3); \]
\[ F_f_s = -\alpha*\text{flt\_s}*((s*\text{subs}(U_{ud}, x, 1) - \text{subs}(u_0, x, 1))*\exp(-\theta_{\text{real}}*s))/F_f_s_{\text{smith}}; \]
\[ bd4 = -\text{subs}(ddU_{ud}, x, 1) \ldots \% \text{ boundary condition } \quad -u_{xxx}(1,t)+\text{my}_{tt}(1,t)=f(t) \]
\[ + m*(s^2*\text{subs}(U_{ud}, x, 1) - s*\text{subs}(u_0, x, 1) - \text{subs}(v_0, x, 1)) - F_f_s; \]
\[ bd4 = \text{strcat}\left(\text{char}(bd4), '='0'\right); \]
\[ [C1, C2, C3, C4] = \text{solve}(bd1, bd2, bd3, bd4, 'C1', 'C2', 'C3', 'C4'); \]
\[ B_s = k_{\text{flt}}/(1 + tao_{\text{flt}}*s); \]
\[ \text{flt\_s} = (1 + B_s)/(1 + B_s*\exp(-\theta_{\text{guess}}*s)); \]
\[ F_f_s_{\text{smith}} = 1 + \alpha*(G_v_{\text{tip\_s}}*(1-\text{flt\_s}*\exp(-\theta_{\text{guess}}*s))); \]
\[ \text{otherwise} \]
\[ \text{error('wrong controller type.');} \]
\[ \text{end} \]
\[ C1 = \text{subs}(C1); \]
\[ C2 = \text{subs}(C2); \]
\[ U_{\text{tip\_s}} = \text{subs}(\text{subs}(U_{ud}), x, 1); \]
\[ \text{if (findstr('Ff\_s', char(U_{\text{tip\_s}})))} \]
\[ \quad \text{error('There is Ff\_s in U_{\text{tip\_s}}');} \]
\[ \text{end} \]
\[ v_{\text{tip\_s}} = s*U_{\text{tip\_s}}-\text{subs}(u_0, x, 1); \]
\[ v_{\text{tip\_s\_str}} = \text{char}(v_{\text{tip\_s}}); \]
\[ v_{\text{tip\_s\_str}} = \text{strrep}(v_{\text{tip\_s\_str}}, '[*, ', '); \% \text{ for future use, capable of accepting vector input} \]
\[ v_{\text{tip\_s\_str}} = \text{strrep}(v_{\text{tip\_s\_str}}, '/', ' '); \]
\[ v_{\text{tip\_s\_str}} = \text{strrep}(v_{\text{tip\_s\_str}}, '^^', ' '); \]
\[ \text{fid\_lap\_v} = \text{fopen('F\_lap\_v.m', 'Wt');} \]
\[ \text{fprintf(fid\_lap\_v, 'function F = F\_lap\_v(s)\n');} \]
\[ \text{fprintf(fid\_lap\_v, 'F = %s;\n', v_{\text{tip\_s\_str}});} \]
\[ \text{fclose(fid\_lap\_v);} \]
\[ \text{clear F\_lap\_v; } \% \text{ force flushing, work-around against the Matlab I/O bug} \]
\[ U_{\text{xs}} = \text{subs}(U_{ud}); \]
\[ U_{\text{xs\_str}} = \text{char}(U_{\text{xs}}); \]
\[ U_{\text{xs\_str}} = \text{strrep}(U_{\text{xs\_str}}, '[*, ', '); \]
\[ U_{\text{xs\_str}} = \text{strrep}(U_{\text{xs\_str}}, '\/', ' '); \]
\[ U_{\text{xs\_str}} = \text{strrep}(U_{\text{xs\_str}}, '^\^', ' '); \]
\[ \text{fid\_lap} = \text{fopen('F\_lap\_m.m', 'Wt');} \]
\[ \text{fprintf(fid\_lap, 'function F = F\_lap(s)\n');} \]
\[ \text{fprintf(fid\_lap, 'F = %s;\n', U_{\text{xs\_str}});}
fclose(fid_lap);
clear F_lap;

A.2 Source Code for Chapter 4

% beam_smith.m
% BEAM_SMITH calculating the transfer function of boundary control of beam
% equation using Smith predictor and its variants
%
% function U_xs = beam_smith(m, theta_real, theta_guess, ctrl_type, ctrl_para, u_0, v_0)
%
% m: tip mass, currently only m = 0 is tested
% theta_real: real time delay of tip velocity measurement
% theta_guess: guessed time delay of tip velocity measurement, currently
%     only theta_guess = theta_real is tested
% ctrl_type: controller type, can be 0, 1, 2, or 3
% 0: no Smith controller, only static controller
% 1: plain Smith controller, no additional filter
% 2: modified Smith controller with lead-lag filter
% 3: modified Smith controller with time advance approximator
% ctrl_para: controller parameters
%     if ctrl_type == 0 or ctrl_type == 1,
%         ctrl_para = [alfa], where alfa is the static controller gain
%     if ctrl_type == 2,
%         ctrl_para = [alfa, k, z, p], the transfer function of
%             lead-lag filter is k(s+z)/(s+p)
%     if ctrl_type == 3,
%         ctrl_para = [alfa, k, tao], the transfer function of
%             time advance approximator is
%             (1+B(s))/(1+B(s)exp(-theta_guess*s))
% u_0: initial displacement condition, u_0 = u_0(x)
% v_0: initial velocity condition, v_0 = v_0(x)

function U_xs = beam_smith(m, theta_real, theta_guess, ctrl_type, ctrl_para, u_0, v_0)

syms x s C1 C2 C3 C4 F_f_s Ff_s_smith k_flt z_flt p_flt tao_flt alfa flt_s Ff_s_smith
s: Laplace transform variable
C1, C2, C3, C4: undetermined constants in the solution of the fourth order ODE
F_f_s: Laplace transform of boundary control force
m: tip mass
alfa: controller gain, see Francis Conrad and Omer Morgul,
     On the stabilization of a flexible beam with a tip mass,
theta: tip velocity feedback delay

% first to get the transfer function of the tip
keyboard
u_0_bak = u_0;
v_0_bak = v_0;
u_0 = 0;
v_0 = 0;
u_xxx_0 = diff(u_0, 'x', 3);

ode = strcat('D4U+s^2*U','+(', char(sym(-s*u_0)),')+(', char(sym(-v_0)),')', '=0');

U_ud = simple(dsolve(ode, 'x'));

bd1 = strcat(char(subs(U_ud, x, 0)), '=0'); % boundary condition $u(0, t)=0$
bd2 = strcat(char(subs(U_ud, x, 0)), '=0'); % boundary condition $u_x(0,t)=0$
bd3 = strcat(char(subs(U_ud, x, 1)), '=0'); % boundary condition, no control torque
bd4 = -subs(dddU_ud, x, 1) ... % boundary condition $-u_{xxx}(1,t)+m_y_{tt}(1,t)=f(t)$
  + m*(s^2*subs(U_ud, x, 1) - s*subs(u_0, x, 1) - subs(v_0, x, 1)) - Ff_s;
bd4 = strcat(char(simple(bd4)), '=0');

[C1, C2, C3, C4] = solve(bd1, bd2, bd3, bd4, 'C1', 'C2', 'C3', 'C4');

U_tip_s = subs(subs(U_ud), x, 1);
G_u_tip_s = simple(U_tip_s/Ff_s);
if (findstr('Ff_s', char(G_u_tip_s))
  error('failed to get transfer function of the tip.');
end

G_v_tip_s = G_u_tip_s*s; % we need transfer function of velocity rather than displacement

% simulate smith predictor
u_0 = u_0_bak;
v_0 = v_0_bak;
u_xxx_0 = diff(u_0, 'x', 3);

ode = strcat('D4U+s^2*U','+(', char(sym(-s*u_0)),')+(', char(sym(-v_0)),')', '=0');

U_ud = simple(dsolve(ode, 'x'));

bd1 = strcat(char(subs(U_ud, x, 0)), '=0'); % boundary condition $u(0, t)=0$
bd2 = strcat(char(subs(U_ud, x, 0)), '=0'); % boundary condition $u_x(0,t)=0$
bd3 = strcat(char(simple(subs(U_ud, x, 1))), '=0'); % boundary condition, no control torque
bd4 = -subs(dddU_ud, x, 1) ... % boundary condition $-u_{xxx}(1,t)+m_y_{tt}(1,t)=f(t)$
  + m*(s^2*subs(U_ud, x, 1) - s*subs(u_0, x, 1) - subs(v_0, x, 1)) - Ff_s;
bd4 = strcat(char(simple(bd4)), '=0');
```matlab
[C1, C2, C3, C4] = solve(bd1, bd2, bd3, bd4, 'C1', 'C2', 'C3', 'C4');

case 1 % plain Smith predictor
alfa = ctrl_para(1);
Ff_s = -alfa*((s*subs(U_ud, x, 1)-subs(u_0, x, 1))*exp(-theta_real*s))/Ff_s_smith;
bd4 = -subs(dddU_ud, x, 1) ... % boundary condition $-u_{xxx}(1,t)+my_{tt}(1,t)=f(t)$
   + m*(s^2*subs(U_ud, x, 1) - s*subs(u_0, x, 1) - subs(v_0, x, 1)) - Ff_s;
bd4 = strcat(char(simple(bd4)), '=0');
[C1, C2, C3, C4] = solve(bd1, bd2, bd3, bd4, 'C1', 'C2', 'C3', 'C4');
Ff_s_smith = 1 + alfa*(G_v_tip_s*(1-exp(-theta_guess*s)));

case 2 % modified Smith predictor with lead-lag filter
alfa = ctrl_para(1);
k_flt = ctrl_para(2);
z_flt = ctrl_para(3);
p_flt = ctrl_para(4);
Ff_s = -alfa*flt_s*((s*subs(U_ud, x, 1)-subs(u_0,x,1))*exp(-theta_real*s))/Ff_s_smith;
bd4 = -subs(dddU_ud, x, 1) ... % boundary condition $-u_{xxx}(1,t)+my_{tt}(1,t)=f(t)$
   + m*(s^2*subs(U_ud, x, 1) - s*subs(u_0, x, 1) - subs(v_0, x, 1)) - Ff_s;
bd4 = strcat(char(simple(bd4)), '=0');
[C1, C2, C3, C4] = solve(bd1, bd2, bd3, bd4, 'C1', 'C2', 'C3', 'C4');
flt_s = k_flt*(s + z_flt)/(s + p_flt);
Ff_s_smith = 1 + alfa*(G_v_tip_s*(1-flt_s*exp(-theta_guess*s)));

case 3 % modified Smith predictor with time advance approximator
alfa = ctrl_para(1);
k_flt = ctrl_para(2);
tao_flt = ctrl_para(3);
Ff_s = -alfa*flt_s*((s*subs(U_ud, x, 1) - subs(u_0, x, 1))*exp(-theta_real*s))/Ff_s_smith;
bd4 = -subs(dddU_ud, x, 1) ... % boundary condition $-u_{xxx}(1,t)+my_{tt}(1,t)=f(t)$
   + m*(s^2*subs(U_ud, x, 1) - s*subs(u_0, x, 1) - subs(v_0, x, 1)) - Ff_s;
bd4 = strcat(char(simple(bd4)), '=0');
[C1, C2, C3, C4] = solve(bd1, bd2, bd3, bd4, 'C1', 'C2', 'C3', 'C4');
Bs = k_flt/(1 + tao_flt*s);
flt_s = (1 + Bs)/(1 + Bs*exp(-theta_guess*s));
Ff_s_smith = 1 + alfa*(G_v_tip_s*(1-flt_s*exp(-theta_guess*s)));

otherwise
   error('wrong controller type.');
end

C1 = subs(C1);
C2 = subs(C2);
C3 = subs(C3);
C4 = subs(C4);

U_tip_s = subs(subs(U_ud), x, 1);
if (findstr('Ff_s', char(U_tip_s)))
   error('There is Ff_s in U_tip_s');
end

v_tip_s = s*U_tip_s- subs(u_0, x, 1);
v_tip_s_str = char(v_tip_s);
v_tip_s_str = strrep(v_tip_s_str, '\*', '.*'); % for future use, capable of accepting vector input
v_tip_s_str = strrep(v_tip_s_str, '/', './');
v_tip_s_str = strrep(v_tip_s_str, '^', '.^');

fid_lap_v = fopen('F_lap_v.m', 'Wt');
```
fprintf(fid_lap_v, 'function F = F_lap_v(s)\n');
fprintf(fid_lap_v, 'F = %s;\n', v_tip_s_str);
fclose(fid_lap_v);
clear F_lap_v; % force flushing, work-around against the Matlab I/O bug

U_xs = subs(U_ud);
U_xs_str = char(U_xs);
U_xs_str = strrep(U_xs_str, '*', '.*');
U_xs_str = strrep(U_xs_str, '/', './');
U_xs_str = strrep(U_xs_str, '^', '.^');

fid_lap = fopen('F_lap.m', 'Wt');
fprintf(fid_lap, 'function F = F_lap(s)\n');
fprintf(fid_lap, 'global x;\n');
fprintf(fid_lap, 'F = %s;\n', U_xs_str);
fclose(fid_lap);
clear F_lap;

A.3 Source Code for Chapter 6

% wave_smith.m
% WAVE_SMITH calculating the transfer function of boundary control of wave
% equation using Smith predictor and its variants
%
% function U_xs = wave_smith(m, theta_real, theta_guess, ctrl_type,
% ctrl_para, noise_para, u_0, v_0)
% % m: tip mass, currently only m = 0 is tested
% % theta_real: real time delay of tip velocity measurement
% % theta_guess: guessed time delay of tip velocity measurement, currently
% % only theta_guess = theta_real is tested
% % ctrl_type: controller type, can be 0, 1, 2, or 3
% % 0: no Smith controller, only static controller
% % 1: plain Smith controller, no additional filter
% % 2: modified Smith controller with lead-lag filter
% % 3: modified Smith controller with time advance approximator
% % ctrl_para: controller parameters
% if ctrl_type == 0
% ctrl_para = [d, k], where d, k are the controller gain
% if ctrl_type == 1
% ctrl_para = [d], where d is the static control gain
% if ctrl_type == 2,
% ctrl_para = [alfa, k, z, p], the transfer function of
% lead-lag filter is k(s+z)/(s+p)
% if ctrl_type ==3,
% ctrl_para = [alfa, k, tao], the transfer function of
% time advance approximator is
% (i+B(s))/(i+B(s)exp(-theta_guess*s)
% u_0: initial displacement condition, u_0 = u_0(x)
% v_0: initial velocity condition, v_0 = v_0(x)
%
% Copyright: Jinsong Liang and YangQuan Chen
% Department of Electrical and Computer Engineering
% Utah State University
function U_xs = wave_smith(alfa, m, theta_real, theta_guess, ctrl_type, ctrl_para, noise_para, u_0, v_0)

syms x s C1 C2 Ff_s Ff_s_smith k_flt z_flt p_flt tao_flt alfa flt_s Ff_s_smith

keyboard

% alfa: fractional order of pde
% s: Laplace transform variable
% C1, C2: undetermined constants in the solution of the fourth order ODE
% Ff_s: Laplace transform of boundary control force
% m: tip mass
% alfa: controller gain, see Francis Conrad and Omer Morgul,
% On the stabilization of a flexible beam with a tip mass,
% theta: tip velocity feedback delay

% first to get the transfer function of the tip
u_0_bak = u_0;
v_0_bak = v_0;
u_0 = 0;
v_0 = 0;
u_xxx_0 = diff(u_0, 'x', 3);

%ode = strcat('D2U-s^2*U','+(', char(sym(s*u_0)),')+(', char(sym(v_0)), ')',' = 0');
%n_order = ceil(alfa); % n_order-1 < alfa <= n_order
ode = strcat('D2U', '-', char(s^alfa), '*U', '+(', char(s^(alfa-1)*sym(u_0)),')+(', char(s^(alfa-2)*sym(v_0)), ')',' = 0');

U_ud = simple(dsolve(ode, 'x')); % U(x,s) with undefined constants
dU_ud = simple(diff(U_ud, 'x', 1)); % first order derivative, with undefined constants

bd1 = strcat(char(subs(U_ud, x, 0)), '=0'); % boundary condition $u(0, t)=0$
bd2 = subs(dU_ud, x, 1) - Ff_s;
bd2 = strcat(char(bd2), '=0');

[C1, C2] = solve(bd1, bd2, 'C1', 'C2');

%keyboard
U_tip_s= subs(subs(U_ud), x, 1);
G_u_tip_s = simple(U_tip_s/Ff_s);
if (findstr('Ff_s', char(G_u_tip_s)))
    error('failed to get transfer function of the tip.');
end
G_v_tip_s= G_u_tip_s*s; % we need transfer function of velocity rather than displacement

% simulate smith predictor
u_0 = u_0_bak;
v_0 = v_0_bak;

%ode = strcat('D2U-s^2*U','+(', char(sym(s*u_0)),')+(', char(sym(v_0)), ')',' = 0');
ode = strcat('D2U', '-', char(s^alfa), '*U', '+(', char(s^(alfa-1)*sym(u_0)),')+(', char(s^(alfa-2)*sym(v_0), ')',' = 0');
U_ud = simple(dsolve(ode, 'x')); % U(x,s) with undefined constants
dU_ud = simple(diff(U_ud, 'x', 1)); % first order derivative with undefined constants

bd1 = strcat(char(subs(U_ud, x, 0)), '=0'); % boundary condition $u(0, t)=0$

switch ctrl_type,
    case 0 % no Smith predictor at all
        d_gain = ctrl_para(1); % static controller gain
        k_gain = ctrl_para(2); % dynamic controller gain
        if k_gain < 1e-10, % if k_gain is zero, freq_ctrl is not
            freq_ctrl = 0; % necessary to be supplied by caller
        else
            freq_ctrl = ctrl_para(3);
        end
        mag_noise= noise_para(1);
        if mag_noise < 1e-10,
            freq_noise = 0;
        else
            freq_noise = noise_para(2);
        end
        % noise should be chosen as sin(freq_noise*t), rather than
        % cos(freq_noise*t), because cos contradicts with the initial
        % conditions
        bd2 = simple(subs(dU_ud, x, 1) - (-d_gain - k_gain*s/(s^2 + freq_ctrl^2))* 
            (s*subs(U_ud, x, 1) - subs(u_0, x, 1))*exp(-s*theta_real) + mag_noise*freq_noise/(s^2 + freq_noise^2));
        bd2 = strcat(char(bd2), '=0');
    end

    case 1 % plain Smith predictor
        d_gain = ctrl_para(1);
        Ff_s = -d_gain*((s*subs(U_ud, x, 1)-subs(u_0, x, 1))*exp(-theta_real*s))/Ff_s_smith;
        bd2 = subs(dU_ud, x, 1) - Ff_s;
        bd2 = strcat(char(bd2), '=0');
    end

    case 2 % modified Smith predictor with lead-lag filter
        d_gain = ctrl_para(1);
        k_flt = ctrl_para(2);
        z_flt = ctrl_para(3);
        p_flt = ctrl_para(4);
        Ff_s = -d_gain*flt_s*((s*subs(U_ud, x, 1)-subs(u_0,x,1))*exp(-theta_real*s))/Ff_s_smith;
        bd2 = subs(dU_ud, x, 1) - Ff_s;
        bd2 = strcat(char(bd2), ' =0');
    end

    case 3 % modified Smith predictor with time advance approximator
        alfa = ctrl_para(1);
        k_flt = ctrl_para(2);
        tao_flt = ctrl_para(3);
        Ff_s = -alfa*flt_s*((s*subs(U_ud, x, 1) - subs(u_0, x, 1))*exp(-theta_real*s))/Ff_s_smith;
bd4 = -subs(dddU_ud, x, 1) ... % boundary condition $-u_{xxx}(1,t)+m\gamma_{tt}(1,t)=f(t)$
 bd4 = strcat(char(simple(bd4)), '0');
 [C1, C2, C3, C4] = solve(bd1, bd2, bd3, bd4, 'C1', 'C2', 'C3', 'C4');
 Bs = k_flt/(1 + tao_flt*s);
 flt_s = (1 + Bs)/(1 + Bs*exp(-theta_guess*s));
 Ff_s_smith = 1 + alfa*(G_v_tip_s*(1-flt_s*exp(-theta_guess*s)));
 otherwise
   error('Wrong controller type.');
 end

C1 = subs(C1);
C2 = subs(C2);
U_tip_s = subs(subs(U_ud), x, 1);
if (findstr('Ff_s', char(U_tip_s)))
   error('There is Ff_s in U_tip_s');
end
v_tip_s = s*U_tip_s- subs(u_0, x, 1);
v_tip_s_str = char(v_tip_s);
v_tip_s_str = strrep(v_tip_s_str, '*', '.*'); % for future use, capable of accepting vector input
v_tip_s_str = strrep(v_tip_s_str, '/', './');
v_tip_s_str = strrep(v_tip_s_str, '^', '.^');

fid_lap_v = fopen('F_lap_v.m', 'wt');
fprintf(fid_lap_v, 'function F = F_lap_v(s)
');
fprintf(fid_lap_v, 'F = %s;
', v_tip_s_str);
fclose(fid_lap_v);
clear F_lap_v; % force flushing, work-around against the Matlab I/O bug

U_xs = subs(U_ud);
U_xs_str = char(U_xs);
U_xs_str = strrep(U_xs_str, '*', '.*');
U_xs_str = strrep(U_xs_str, '/', './');
U_xs_str = strrep(U_xs_str, '^', '.^');

fid_lap = fopen('F_lap.m', 'wt');
fprintf(fid_lap, 'function F = F_lap(s)
');
fprintf(fid_lap, 'global x;
');
fprintf(fid_lap, 'F = %s;
', U_xs_str);
fclose(fid_lap);
clear F_lap;

A.4 Source Code for Chapter 7

% fracwave_fractrl_smith.m
% WAVE_SMITH calculating the transfer function of boundary control of wave
% equation using Smith predictor and its variants
%
% function U_xs = wave_smith(m, theta_real, theta_guess, ctrl_type,
%       ctrl_para, noise_para, u_0, v_0)
% % m: tip mass, currently only m = 0 is tested
% theta_real: real time delay of tip velocity measurement
% theta_guess: guessed time delay of tip velocity measurement, currently
% only theta_guess = theta_real is tested
% ctrl_type: controller type, can be 0, 1, 2, or 3
% 0: no Smith controller, only static controller
% 1: plain Smith controller, no additional filter
% 2: modified Smith controller with lead-lag filter
% 3: modified Smith controller with time advance approximator
% ctrl_para: controller parameters
% if ctrl_type == 0
% ctrl_para = [d, k], where d, k are the controller gain
% if ctrl_type == 1
% ctrl_para = [d], where d is the static control gain
% if ctrl_type == 2,
% ctrl_para = [alfa, k, z, p], the transfer function of
% lead-lag filter is k(s+z)/(s+p)
% if ctrl_type == 3,
% ctrl_para = [alfa, k, tao], the transfer function of
% time advance approximator is
% (1+B(s))/(1+B(s)exp(-theta_guess*s))
% u_0: initial displacement condition, u_0 = u_0(x)
% v_0: initial velocity condition, v_0 = v_0(x)

function U_xs = fracwave_fractrl_smith(alfa_order, m, theta_real, theta_guess, ctrl_type, ctrl_para, noise_para, u_0, v_0)

syms x s C1 C2 Ff_s Ff_s_smith k_flt z_flt p_flt tao_flt alfa flt_s Ff_s_smith

% alfa_order: fractional order of pde
% s: Laplace transform variable
% C1, C2: undetermined constants in the solution of the fourth order ODE
% Ff_s: Laplace transform of boundary control force
% m: tip mass
% alfa: controller gain, see Francis Conrad and Omer Morgul,
% On the stabilization of a flexible beam with a tip mass,
% theta: tip velocity feedback delay

d_gain = ctrl_para(1); % static controller gain
k_gain = ctrl_para(2); % dynamic controller gain
mu = ctrl_para(3); % fractional controller order
freq_ctrl = ctrl_para(4);

% first to get the transfer function of the tip
u_0_bak = u_0;
v_0_bak = v_0;
u_0 = 0;
v_0 = 0;

%keyboard
ode = strcat('D2U', ' - ', char(s^alfa_order), '*U', '+(', char(s^(alfa_order-1)*sym(u_0)), ')', '+(', char(s^(alfa_order-2)*sym(v_0)), ')', '= 0');

U_ud = simple(dsolve(ode, 'x')); % U(x,s) with undefined constants
dU_ud = simple(diff(U_ud, 'x', 1)); % first order derivative, with undefined constants

bd1 = strcat(char(subs(U_ud, x, 0)), '=0'); % boundary condition $u(0, t)=0$
bd2 = subs(dU_ud, x, 1) - Ff_s;
bd2 = strcat(char(bd2), '=0');

[C1, C2] = solve(bd1, bd2, 'C1', 'C2');

U_tip_s = subs(subs(U_ud), x, 1);
G_u_tip_s = simple(U_tip_s/Ff_s);
if (findstr('Ff_s', char(G_u_tip_s)))
    error('failed to get transfer function of the tip.');
end
G_v_tip_s= G_u_tip_s*s^mu; % we need transfer function of fractional derivative

% keyboard
% simulate smith predictor
u_0 = u_0_bak;
v_0 = v_0_bak;

% keyboard
% ode = strcat('D2U-s^2*U','+(', char(sym(s*u_0)),')+(', char(sym(v_0)), ')',' = 0');
ode = strcat('D2U', '-', char(s^alfa_order), '*U', '+(', char(s^(alfa_order-1)*sym(u_0)),')+(', char(s^(alfa_order-2)*sym(v_0)), ')',' = 0');

U_ud = simple(dsolve(ode, 'x')); % U(x,s) with undefined constants
dU_ud = simple(diff(U_ud, 'x', 1)); % first order derivative with undefined constants

bd1 = strcat(char(subs(U_ud, x, 0)), '=0'); % boundary condition $u(0, t)=0$

switch ctrl_type,
    case 0 % no Smith predictor at all
        %if k_gain < 1e-10, % if k_gain is zero, freq_ctrl is not
            % freq_ctrl = 0; % necessary to be supplied by caller
        elseif
        % freq_ctrl = ctrl_para(3);
    end

    mag_noise = noise_para(1);
    freq_noise = noise_para(2);
    %if mag_noise < 1e-10,
        % freq_noise = 0;
    elseif
        % freq_noise = noise_para(2);
    end

    % noise should be chosen as sin(freq_noise*t), rather than
    % cos(freq_noise*t), because cos contradicts with the initial
    % conditions
    bd2 = simple(subs(dU_ud, x, 1) - (-d_gain - k_gain*s/(s^2 + freq_ctrl^2))* ... 
            (s^mu*subs(U_ud, x, 1) - s^(mu-1)*subs(u_0, x, 1)))*exp(-s*theta_real) + mag_noise*freq_noise/(s^2 + freq_noise^2); 
            (s*subs(U_ud, x, 1) - subs(u_0, x, 1))*exp(-s*theta_real) + mag_noise*s/(s^2 + freq_noise^2));
bd2 = strcat(char(bd2), '=0');

[C1, C2] = solve(bd1, bd2, 'C1', 'C2');
case 1 % plain Smith predictor
d_gain = ctrl_para(1);
Ff_s = -d_gain*(((s*subs(U_ud, x, 1)-subs(u_0, x, 1))*exp(-theta_real*s))/Ff_s_smith;
Ff_s = -d_gain*(s^mu*subs(U_ud, x, 1) - s^(mu-1)*subs(u_0, x, 1))*exp(-theta_real*s)/Ff_s_smith;

bd2 = subs(dU_ud, x, 1) - Ff_s;
bd2 = strcat(char(bd2), '=0');

[C1, C2] = solve(bd1, bd2, 'C1', 'C2');
Ff_s_smith = 1 + d_gain*(G_v_tip_s*(1-exp(-theta_guess*s)));
otherwise
error('wrong controller type.');
end

C1 = subs(C1);
C2 = subs(C2);

U_tip_s = subs(subs(U_ud), x, 1);
if (findstr('Ff_s', char(U_tip_s))
error('There is Ff_s in U_tip_s');
end

v_tip_s = s*U_tip_s- subs(u_0, x, 1);
v_tip_s_str = char(v_tip_s);
v_tip_s_str = strrep(v_tip_s_str, '*', '.*'); % for future use, capable of accepting vector input
v_tip_s_str = strrep(v_tip_s_str, '/', './');
v_tip_s_str = strrep(v_tip_s_str, '^', '.^');

fid_lap_v = fopen('F_lap_v.m', 'Wt');
fprintf(fid_lap_v, 'function F = F_lap_v(s)
');
fprintf(fid_lap_v, 'global x d_gain mu;
');
fprintf(fid_lap_v, 'F = %s;
', v_tip_s_str);
close(fid_lap_v);
clear F_lap_v; % force flushing, work-around against the Matlab I/O bug

U_xs = subs(U_ud);
U_xs_str = char(U_xs);
U_xs_str = strrep(U_xs_str, '*', '.*');
U_xs_str = strrep(U_xs_str, '/', './');
U_xs_str = strrep(U_xs_str, '^', '.^');

fid_lap = fopen('F_lap.m', 'Wt');
fprintf(fid_lap, 'function F = F_lap(s)
');
fprintf(fid_lap, 'global x d_gain mu;
');
fprintf(fid_lap, 'F = %s;
', U_xs_str);
close(fid_lap);
clear F_lap;

A.5 Source Code for Chapter 8

% frac_wave.m
% FRAC_WAVE calculating the Laplace transform of the solution of
% fractional diffusion-wave equation w.r.t boundary controller
%
% The Laplace transform results are saved to files F_lap.m and F_lap_v.m.
% F_lap.m saves the Laplace transform of displacement.
% F_lap_v.m saves the Laplace transform of velocity.
%
% Currently, the time-derivative can be fractional order and the boundary
% controller can be zero (no controller), static, plain Smith predictor,
% or modified Smith predictor
%
% In the future, the space-derivative can expected be fractional order.
%
% function U_xs = wave_smith(pde_para, m, theta_real, theta_guess,
% ctrl_type, ctrl_para, noise_para, u_0, v_0)
% % pde_para: parameters describing fractional diffusion-wave equation
% pde_para = [torder, b]
% % pde is described as (LaTeX notation):
% \[
% \frac{\partial^{torder}u}{\partial t^{torder}} = b^2 \frac{\partial^2 u}{\partial x^2}
% \]
% m: tip mass, currently only m = 0 is tested
% theta_real: real time delay of tip velocity measurement
% theta_guess: guessed time delay of tip velocity measurement, currently
% only theta_guess = theta_real is tested
% ctrl_type: controller type, can be 0, 1, 2, or 3
% 0: no Smith controller, only static controller
% 1: plain Smith controller, no additional filter
% 2: modified Smith controller with lead-lag filter
% 3: modified Smith controller with time advance approximator
% ctrl_para: controller parameters
% if ctrl_type == 0{
% ctrl_para = [d, k], where d, k are the controller gain
% % For meanings of d and k, refer to Stabilization and
% % disturbance rejection for the wave equation, Omer Morgul,
% % equation (52)
% }
% if ctrl_type == 1
% ctrl_para = [d], where d is the static control gain
% if ctrl_type == 2,
% ctrl_para = [alfa, k, z, p], the transfer function of
% lead-lag filter is k(s+z)/(s+p)
% if ctrl_type ==3,
% ctrl_para = [alfa, k, tao], the transfer function of
% time advance approximator is
% \[
% (1+B(s))/(1+B(s)\exp(-theta_guess*s))
% \]
% u_0: initial displacement condition, u_0 = u_0(x)
% v_0: initial velocity condition, v_0 = v_0(x)

% Copyright: Jinsong Liang and YangQuan Chen
% Department of Electrical and Computer Engineering
% Utah State University
% email: jsliang@ieee.org
% yqchen@ieee.org
% Last Modified: 11/09/2003
function U_xs = wave_smith(pde_para, m, theta_real, theta_guess, ctrl_type, ctrl_para, noise_para, u_0, v_0)

    % s: Laplace transform variable
    % C1, C2: undetermined constants in the solution of the second order ODE
    % Ff_s: Laplace transform of boundary control force
    % m: tip mass
    % alfa: controller gain, see Francis Conrad and Omer Morgul,
    %       On the stabilization of a flexible beam with a tip mass,
    % theta: tip velocity feedback delay

    u_0_bak = u_0;
    v_0_bak = v_0;
    u_0 = 0;
    v_0 = 0;
    u_xxx_0 = diff(u_0, 'x', 3);

    torder = pde_para(1);
    b_sqr = pde_para(2)^2;
    ode = strcat(char(sym(b_sqr)), '*D2U', '-', char(s^torder), '*U', '+(', char(s^(torder-1)*sym(u_0)),')+(', char(s^(torder-2)*sym(v_0)), ')', ' = 0');
    U_ud = simple(dsolve(ode, 'x')); % U(x,s) with undefined constants
    dU_ud = simple(diff(U_ud, 'x', 1)); % first order derivative, with undefined constants
    bd1 = strcat(char(subs(U_ud, x, 0)), '=0'); % boundary condition $u(0, t)=0$
    bd2 = subs(dU_ud, x, 1) - Ff_s;
    bd2 = strcat(char(bd2), '=0');
    [C1, C2] = solve(bd1, bd2, 'C1', 'C2');
    U_tip_s = subs(subs(U_ud), x, 1);
    G_u_tip_s = simple(U_tip_s/Ff_s); % we need transfer function of velocity rather than displacement
    if (findstr('Ff_s', char(G_u_tip_s)))
        error('failed to get transfer function of the tip.');
    end
    G_v_tip_s = G_u_tip_s*s; % we need transfer function of velocity rather than displacement

    u_0 = u_0_bak;
    v_0 = v_0_bak;

    ode = strcat('D2U-s^2*U','+(', char(sym(s*u_0)),')+(', char(sym(v_0)), ')', ' = 0');
    ode = strcat(char(sym(b_sqr)), '*D2U', '-', char(s^torder), '*U', '+(', char(s^(torder-1)*sym(u_0)),')+(', char(s^(torder-2)*sym(v_0)), ')', ' = 0');
    U_ud = simple(dsolve(ode, 'x')); % U(x,s) with undefined constants
    dU_ud = simple(diff(U_ud, 'x', 1)); % first order derivative with undefined constants
bd1 = strcat(char(subs(U_ud, x, 0)), ' = 0'); % boundary condition $u(0, t)=0$

switch ctrl_type,
case 0  % no Smith predictor at all
    d_gain = ctrl_para(1); % static controller gain
    k_gain = ctrl_para(2); % dynamic controller gain
    if k_gain < 1e-10, % if k_gain is zero, freq_ctrl is not
        freq_ctrl = 0; % necessary to be supplied by caller
    else
        freq_ctrl = ctrl_para(3);
    end
    mag_noise = noise_para(1);
    if mag_noise < 1e-10,
        freq_noise = 0;
    else
        freq_noise = noise_para(2);
    end
    % noise should be chosen as sin(freq_noise*t), rather than
    % cos(freq_noise*t), because cos contradicts with the initial
    % conditions
    bd2 = simple(subs(dU_ud, x, 1) - (-d_gain - k_gain*s/(s^2 + freq_ctrl^2))* ...
                 (s*subs(U_ud, x, 1) - subs(u_0, x, 1))*exp(-s*theta_real) + mag_noise*freq_noise/(s^2 + freq_noise^2));
    bd2 = strcat(char(bd2), ' = 0');
    [C1, C2] = solve(bd1, bd2, 'C1', 'C2');

case 1  % plain Smith predictor
    d_gain = ctrl_para(1);
    Ff_s = -d_gain*((s*subs(U_ud, x, 1)-subs(u_0, x, 1))*exp(-theta_real*s))/Ff_s_smith;
    bd2 = subs(dU_ud, x, 1) - Ff_s;
    bd2 = strcat(char(bd2), ' = 0');
    [C1, C2] = solve(bd1, bd2, 'C1', 'C2');
    Ff_s_smith = 1 + d_gain*G_v_tip_s*(1-exp(-theta_guess*s));

case 2  % modified Smith predictor with lead-lag filter
    d_gain = ctrl_para(1);
    k_flt = ctrl_para(2);
    z_flt = ctrl_para(3);
    p_flt = ctrl_para(4);
    Ff_s = -d_gain*k_flt*s*((s*subs(U_ud, x, 1)-subs(u_0, x, 1))*exp(-theta_real*s))/Ff_s_smith;
    bd2 = subs(dU_ud, x, 1) - Ff_s;
    bd2 = strcat(char(bd2), ' = 0');
    [C1, C2] = solve(bd1, bd2, 'C1', 'C2');
    flt_s = k_flt*s+z_flt)/(s+p_flt);
    Ff_s_smith = 1 + d_gain*G_v_tip_s*(1-flt_s*exp(-theta_guess*s));

case 3  % modified Smith predictor with time advance approximator
    alfa = ctrl_para(1);
    k_flt = ctrl_para(2);
    tao_flt = ctrl_para(3);
    Ff_s = -alfa*k_flt*s*((s*subs(U_ud, x, 1)-subs(u_0, x, 1))*exp(-theta_real*s))/Ff_s_smith;
    bd4 = -subs(dddU_ud, x, 1) ... % boundary condition $-u_{xxx}(1,t)+my_{tt}(1,t)=f(t)$
         + m*s*subs(U_ud, x, 1) - s*subs(u_0, x, 1) - subs(v_0, x, 1) - Ff_s;
    bd4 = strcat(char(simple(bd4)), ' = 0');
\[ [C_1, C_2, C_3, C_4] = \text{solve}(b_1, b_2, b_3, b_4, 'C_1', 'C_2', 'C_3', 'C_4'); \]
\[
B_s = k_{\text{flt}}/(1 + tao_{\text{flt}}s);
\]
\[
\text{flt}_s = (1 + B_s)/(1 + B_s*\exp(-\theta_{\text{guess}}s));
\]
\[
\text{Ff}_s_{\text{smith}} = 1 + \alpha\cdot(G_{v_{\text{tip}}}(1-\text{flt}_s\cdot\exp(-\theta_{\text{guess}}s)));
\]

otherwise
\[
\text{error('wrong controller type.');} \]

end

\[
C_1 = \text{subs}(C_1);
\]
\[
C_2 = \text{subs}(C_2);
\]

\[
\text{U}_{\text{tip}} = \text{subs}((\text{U}_{ud}, x, 1);
\]
if (findstr('Ff_s', char(U_tip_s)))
\[
\text{error('There is Ff_s in U_tip_s');}
\]
end

\[
\text{v}_{\text{tip}} = s*\text{U}_{\text{tip}}- \text{subs}(u_0, x, 1);
\]
\[
\text{v}_{\text{tip}}_{\text{str}} = \text{char}(\text{v}_{\text{tip}});
\]
\[
\text{v}_{\text{tip}}_{\text{str}} = \text{strrep}(	ext{v}_{\text{tip}}_{\text{str}}, '*', '.*'); \quad \%	ext{for future use, capable of accepting vector input}
\]
\[
\text{v}_{\text{tip}}_{\text{str}} = \text{strrep}(	ext{v}_{\text{tip}}_{\text{str}}, '/', './');
\]
\[
\text{v}_{\text{tip}}_{\text{str}} = \text{strrep}(	ext{v}_{\text{tip}}_{\text{str}}, '^', '.^');
\]

\[
fid_{\text{lap}} = \text{fopen('F_lap.m', 'Wt')};
\]
fprintf(fid_lap_v, 'function F = F_lap_v(s)\n');
fprintf(fid_lap_v, 'F = %s;\n', v_tip_s_str);
fclose(fid_lap_v);
clear F_lap_v; \quad \%	ext{force flushing, work-around against the Matlab I/O bug}

\[
\text{U}_{xs} = \text{subs}(\text{U}_{ud});
\]
\[
\text{U}_{xs} = \text{char}(\text{U}_{xs});
\]
\[
\text{U}_{xs} = \text{strrep}(	ext{U}_{xs}, '*', '.*');
\]
\[
\text{U}_{xs} = \text{strrep}(	ext{U}_{xs}, '/', './');
\]
\[
\text{U}_{xs} = \text{strrep}(	ext{U}_{xs}, '^', '.^');
\]

\[
fid_{\text{lap}} = \text{fopen('F_lap.m', 'Wt')};
\]
fprintf(fid_lap, 'function F = F_lap(s)\n');
fprintf(fid_lap, 'global x;\n');
fprintf(fid_lap, 'global a1 a2 a3 a4 b torder;\n');
fprintf(fid_lap, 'F = %s;\n', U_xs_str);
fclose(fid_lap);
clear F_lap;