A Hybrid Symbolic-Numerical Simulation Method for Some Typical Boundary Control Problems

Jinsong Liang, *Student Member, IEEE*, YangQuan Chen and Bao-Zhu Guo, *Senior Members, IEEE*

***Abstract*— A new simulation method for some typical boundary control problems, combining symbolic algebra and numerical method, is presented with typical examples. The transfer function is obtained in the intermediate steps of the simulation, which makes it possible and easier to apply more advanced boundary controllers in the future.**

***Index Terms*— Simulation; symbolic; numeric; boundary control; beam equation; numerical inverse Laplace transform.**

I. Introduction

Boundary control of linear partial differential equation (PDE) has become an important research area in recent years [1], [2], [3], [4], [5], [6], due to the increasing demand on the high precision control of many mechanical systems, such as spacecraft with flexible attachment or robots with flexible links, which are governed by PDE's rather than ordinary differential equations (ODE's). Two important research topics are the boundary control of wave equation and beam equation, which are often encountered in the practical engineering design. Tracing the progress in the theoretical analysis, we have found an interesting fact, i.e., simulation examples of the boundary control of PDE's 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, we would like to suggest, is that the difficulty of boundary control problems is far beyond the capability of most commonly available mathematical tools, such as Matlab, Maple, even FEMLAB [7]. For example, Matlab PDE Toolbox is only able to solve second order PDE's with Dirichlet and/or generalized Neumann boundary conditions [8], while the PDE's 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 accept.

In this paper, we present an easy-to-implement, yet powerful, boundary control simulation method, which combines the analytical method, the numerical method, and the modern symbolic algebra in a creative way. The simulation examples show that this 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) [9] with no extra software needed except Matlab and the Matlab Symbolic Math Toolbox.

J. S. Liang and Y. Q. Chen are with the Center for Self-Organizing and Intelligent Systems (CSOIS), Dept. of Electrical and Computer Engineering, 4160 Old Main Hill, Utah State University, Logan, UT 84322-4160, USA. B.-Z. Guo is with the Institute of Systems Sciences, Academy of Mathematics and System Sciences, Academia Sinica, Beijing 100080, P. R. China. Corresponding author: Dr YangQuan Chen. E-mail: yqchen@ece.usu.edu; Tel. 01-435-7970148; Fax: 01-435-7973054. URL: http://www.csois.usu.edu/people/yqchen

II. Principle and Implementation

PDE's can be solved by means of Laplace transform [10]. Following is a summary of this method. We assume the solution of a PDE is a function $u(x,t)$ of the two independent variables x and t.

- Transform $u(x,t)$ with respect to t by means of Laplace transform, so we obtain an ODE for the transformed variable $U(x,s)$:

$$f(U(x,s), \frac{dU(x,s)}{dx}, \ldots, \frac{d^nU(x,s)}{dx^n}, x, s) = 0. \quad (1)$$

- Solve the ODE (1) 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 $U(x,s)$.
- 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 (1) 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 we can determine the undefined constants, usually the inverse Laplace transform can not be performed by looking up a table of transform pairs.

We solve the above problems using the Matlab Symbolic Math Toolbox [11] and the numerical inverse Laplace transform [13], [14].

We take the following three boundary control simulation examples to show the implementation procedures in detail.

A. Boundary control of wave equation

In this section, we will simulate the stabilization and disturbance rejection for the wave equation as discussed in [4], which is one of the very few papers with simulation examples. The FDM was used in [4] to simulate the system.

We 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 dynamic boundary control at the other end. The system is represented by

$$u_{tt}(x,t) - u_{xx}(x,t) = 0, \quad (2)$$

$$u(0,t) = 0, \quad (3)$$

$$u_x(1,t) = -f(t), \quad (4)$$

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. We will show the effect of the following control law:

$$\hat{f}(s) = (d + \frac{ks}{s^2+\omega^2})\hat{u}_t(1,s) + \hat{n}(s), \tag{5}$$

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; ω is the frequency of the noise.

The initial conditions are chosen as

$$u(x,0) = -0.5\sin(0.5\pi x), \tag{6}$$

$$u_t(x,0) = 0. \tag{7}$$

The disturbance $n(t)$ is chosen as

$$n(t) = \cos(10t). \tag{8}$$

We will simulate the following two cases to show that the dynamic controller ($k > 0$) is better than the static controller ($k = 0$) to reject the noise.

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

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

We take the simulation of *Case 1* as an example to show the steps. We first take the Laplace transform of (2), (3), (4) with respect to t which gives

$$\frac{\mathrm{d}^2U(x,s)}{\mathrm{d}x^2} - (s^2U(x,s) - su(x,0) - u_t(x,0)) = 0, \tag{9}$$

$$U(0,s) = 0, \tag{10}$$

$$\frac{\mathrm{d}U(1,s)}{\mathrm{d}x} = (d + \frac{ks}{s^2+\omega^2})(sU(1,s) - u(1,0)) + \frac{s}{s^2+\omega^2}, \tag{11}$$

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

Substituting the initial conditions (6) and (7) into (9) and (11), we have

$$\frac{\mathrm{d}^2U(x,s)}{\mathrm{d}x^2} - s^2U(x,s) + s(-0.5\sin(0.5\pi x)) = 0, \tag{12}$$

$$\frac{\mathrm{d}U(1,s)}{\mathrm{d}x} = (d + \frac{ks}{s^2+\omega^2})(sU(1,s) + 0.5) + \frac{s}{s^2+\omega^2}. \tag{13}$$

Next, we solve the equation (12), (10) and (13) using Matlab Symbolic Math Toolbox function `dsolve()`, which symbolically solves the ODE(s) and the boundary and/or initial condition(s). Although `dsolve()` is able to determine the arbitrary constants in the solution using the boundary and/or initial condition(s), we find that its ability is rather weak. So, we supply only (12) to `dsolve()` rather than supply (12), (10) and (13) together and get the following solution with two arbitrary constants $C1$ and $C2$ in it.

```
U(x,s) = 1/2*(-4*s*sin(1/2*pi*x)*exp(-s*x)+C1*pi^2-C1*pi^2*e
xp(-2*s*x)+4*C1*s^2-4*C1*s^2*exp(-2*s*x)+C2*pi^2+4*C2*s^2+4*
C2*s^2*exp(-2*s*x)+C2*pi^2*exp(-2*s*x))/(4*s^2+pi^2)*exp(s*x
)                                                          (14)
```

Next, we differentiate $U(x,s)$ with respect to x to get the first order derivative of $U(x,s)$ using Matlab Symbolic Math Toolbox function `diff()`. The expression of $\mathrm{d}U(x,s)/\mathrm{d}x$ is

```
dU(x,s)/dx = -1/2*s*exp(s*x)*(2*cos(1/2*pi*x)*pi*exp(-s*x)-C
1*pi^2*exp(-2*s*x)-4*C1*s^2*exp(-2*s*x)+4*C2*s^2*exp(-2*s*x)
+C2*pi^2*exp(-2*s*x)-C1*pi^2-4*C1*s^2-C2*pi^2-4*C2*s^2)/(4*s
^2+pi^2)                                                   (15)
```

Substituting $U(x,s)$ (14) and its first order derivative (15) to the boundary conditions (10) and (11), we have two boundary conditions (16) and (17) with two undetermined constants $C1$ and $C2$.

```
1/2*(2*C2*pi^2+8*C2*s^2)/(4*s^2+pi^2)=0                    (16)
```

```
-1/2*s*exp(s)*(-C1*pi^2*exp(-2*s)-4*C1*s^2*exp(-2*s)+4*C2*s^
2*exp(-2*s)+C2*pi^2*exp(-2*s)-C1*pi^2-4*C1*s^2-C2*pi^2-4*C2*
s^2)/(4*s^2+pi^2)+(1+10*s/(s^2+100))*(1/2*s*(-4*s*exp(-s)+C1
*pi^2-C1*pi^2*exp(-2*s)+4*C1*s^2-4*C1*s^2*exp(-2*s)+C2*pi^2+
4*C2*s^2+4*C2*s^2*exp(-2*s)+C2*pi^2*exp(-2*s))/(4*s^2+pi^2)*
exp(s)+1/2)+s/(s^2+100)=0                                  (17)
```

Feeding (16) and (17) to the Matlab Symbolic Math Toolbox function `solve()`, the above two algebraic equations can be solved symbolically to give the following expressions for the two constants $C1$ and $C2$:

```
C1 = 1/2*(12*s*pi^2+100*pi^2+s^2*pi^2+8*s^3)/s/exp(s)/(5*s*p
i^2*exp(-2*s)-100*pi^2-s^2*pi^2+20*s^3*exp(-2*s)-5*s*pi^2-40
0*s^2-20*s^3-4*s^4)                                        (18)
```

```
C2 = 0                                                     (19)
```

Now, we have actually obtained the explicit expression of $U(x,s)$, which is shown in (20).

```
U(x,s) = 1/(4*s^2+2778046668940015/281474976710656)*(-2*s*si
n(1/2*pi*x)-2778046668940015/1125899906842624*(s^2*pi^2+8*s^
3+100*pi^2+12*s*pi^2)/s/exp(s)/(s^2*pi^2+4*s^4+400*s^2+100*p
i^2+5*s*pi^2-20*s^3*exp(-2*s)-5*pi^2*exp(-2*s)*s+20*s^3)*exp
(s*x)+2778046668940015/1125899906842624*(s^2*pi^2+8*s^3+100*
pi^2+12*s*pi^2)/s/exp(s)/(s^2*pi^2+4*s^4+400*s^2+100*pi^2+5*
s*pi^2-20*s^3*exp(-2*s)-5*pi^2*exp(-2*s)*s+20*s^3)*exp(-s*x)
-(s^2*pi^2+8*s^3+100*pi^2+12*s*pi^2)*s/exp(s)/(s^2*pi^2+4*s^
4+400*s^2+100*pi^2+5*s*pi^2-20*s^3*exp(-2*s)-5*pi^2*exp(-2*s
)*s+20*s^3)*exp(s*x)+(s^2*pi^2+8*s^3+100*pi^2+12*s*pi^2)*s/e
xp(s)/(s^2*pi^2+4*s^4+400*s^2+100*pi^2+5*s*pi^2-20*s^3*exp(-
2*s)-5*pi^2*exp(-2*s)*s+20*s^3)*exp(-s*x))                 (20)
```

Although obtaining the explicit expression of $U(x,s)$ is just an intermediate step of this simulation method, it is proven in the subsequent paper [12] that this is critical to designing more advanced boundary controllers, since $U(x,s)$ is actually the transfer function of this boundary control system.

To obtain $u(x,t)$, we need to take the inverse Laplace transform of $U(x,s)$. We should *not* use the Matlab Symbolic Math Toolbox function `ilaplace()`, which takes the inverse Laplace transform symbolically, since for such a complicated expression of $U(x,s)$, the explicit expression of $u(x,t)$ is usually unavailable. However, we can make use of the numeric inverse Laplace transform. Among the existing numeric inverse Laplace transform methods, the FFT (Fast Fourier Transform) method seems to be both accurate and fast [13]. So, we choose the program in [14] to take the inverse Laplace transform of $U(x,s)$.

At this point, we have actually finished the time-domain simulation. In what follows, we present some simulation results for both *Case 1* and *Case 2*.

The tip displacement in *Case 1* is shown in Fig. 1. The tip displacement in *Case 2* is shown in Fig. 2. It shows clearly that the dynamic controller is better than the static

controller in rejecting the noise. The two plots are very close to the simulation results reported in [4]. The simulation code developed in this paper has been validated using this example.

It is also very easy to show the displacement of the whole string, which is shown in Fig. 3 and Fig. 4, for *Case 1* and *Case 2*, respectively.

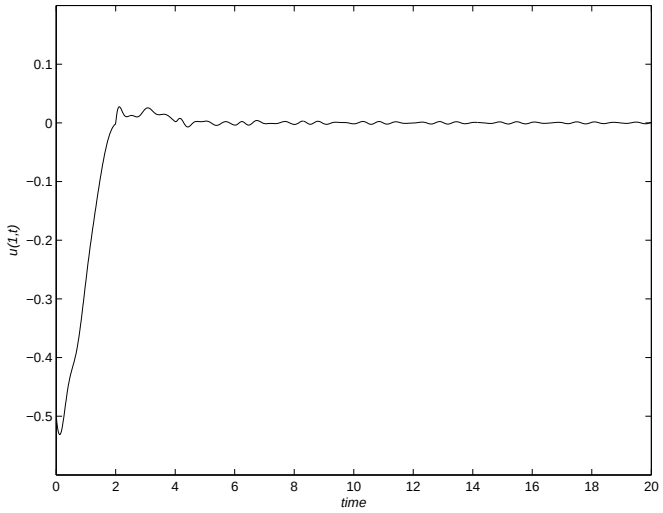

Fig. 1. Tip displacement for *Case 1*

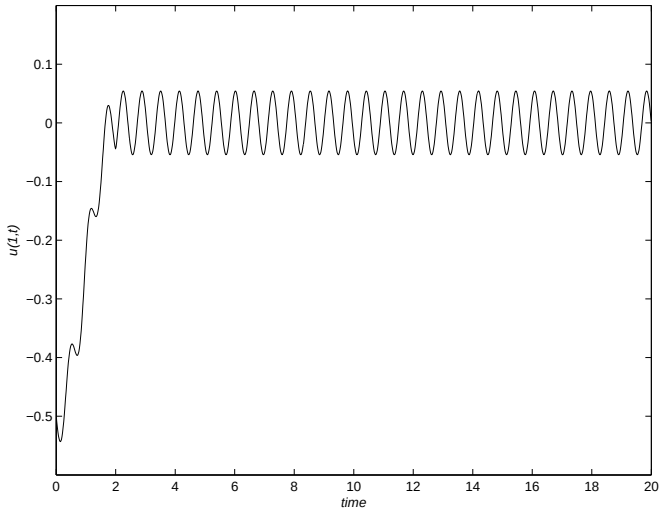

Fig. 2. Tip displacement for *Case 2*

B. Boundary control of beam equation

In this section, the boundary control of beam equation, a fourth order PDE, will be simulated.

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

$$u_{tt} + u_{xxxx} = 0, \tag{21}$$

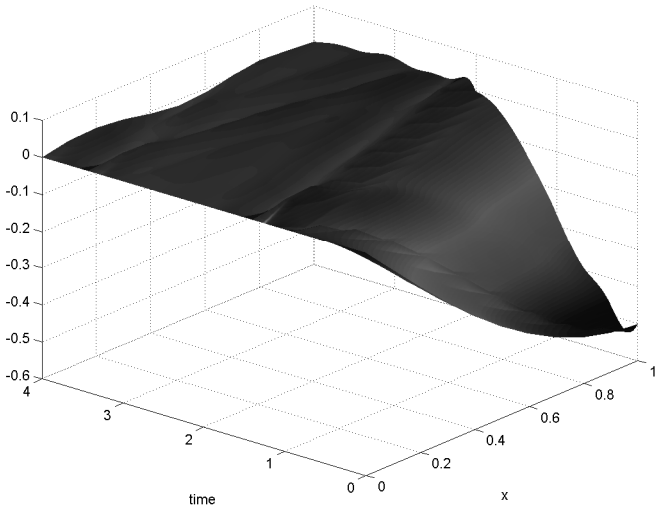

Fig. 3. Displacement of the whole string for *Case 1*

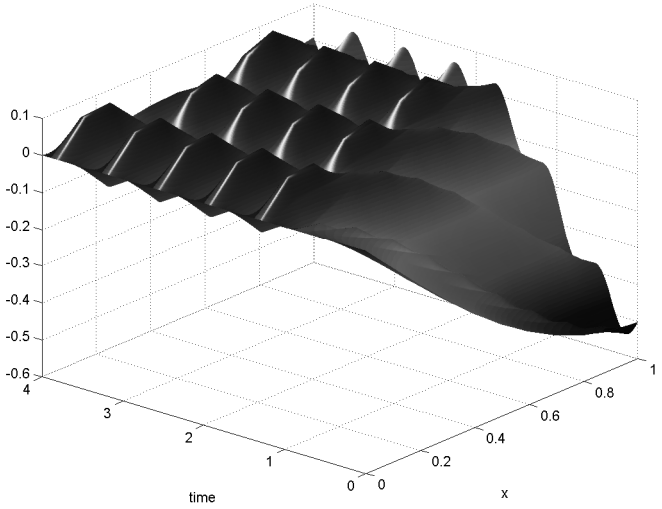

Fig. 4. Displacement of the whole string for *Case 2*

$$u(0,t) = 0, \tag{22}$$

$$u_x(0,t) = 0, \tag{23}$$

$$u_{xx}(1,t) = 0, \tag{24}$$

$$u_{xxx}(1,t) = f(t), \tag{25}$$

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

It is well-known that the following controller stabilizes the displacement of the beam [15]:

$$f(t) = k u_t(1,t), \tag{26}$$

where $k > 0$ is the constant gain.

The initial conditions are chosen as

$$u(x,0) = x^3 - 3x^2, \tag{27}$$

$$u_t(x,0) = 0. \tag{28}$$

The initial condition (27) is a typical displacement profile when the beam is subject to an static force $f = -1$ at the free end [16].

Since the basic simulation procedure are the same as that in Sec. II-A, except that the intermediate results and expressions are much more complicated, only the final simulation results will be shown below.

Figure 5 shows the displacement of the free end of the beam. The displacement of the whole beam is shown in Fig. 6. We comment that the numerical results are correct at least from the quantitative point of view.

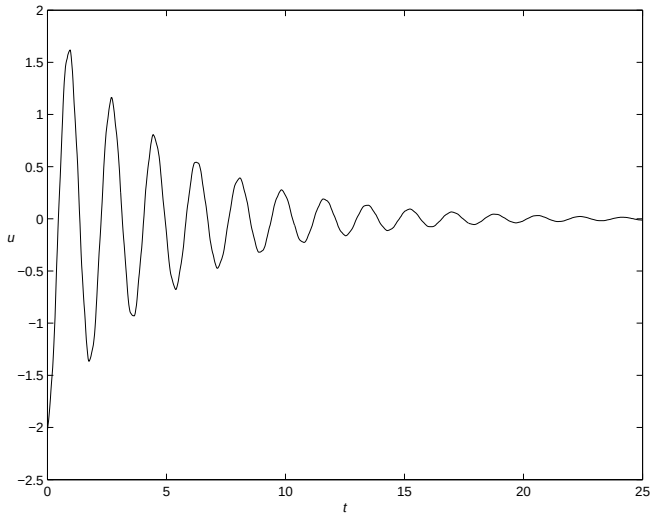

Fig. 5. Displacement of the free end

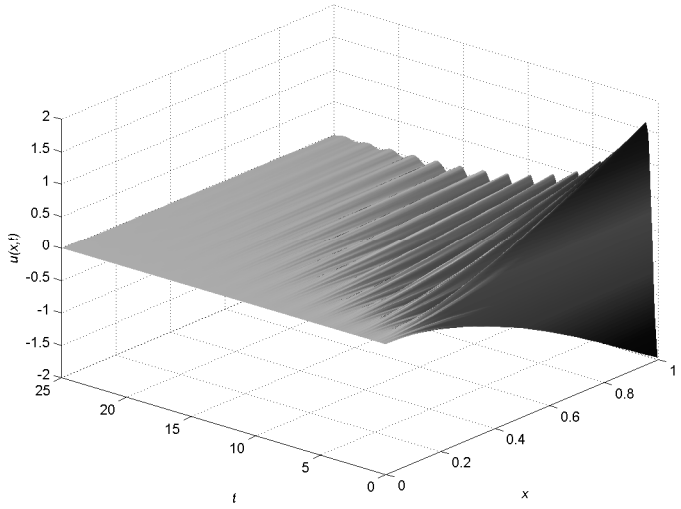

Fig. 6. Displacement of the whole beam

C. Boundary control of beam equation with time delay

In this section, we simulate the boundary control of beam equation with a time delay from the boundary velocity feedback loop due to measurement lag or computation lag.

The PDE, boundary conditions, initial conditions and control gain are the same as in Sec. II-B with the exception of the expression of control input which is changed as follows:

$$f(t) = ku_t(1, t - \theta) \tag{29}$$

where θ is the time delay.

In [17] and [18], it was shown that both boundary control of wave equation and boundary control of beam equation become unstable when an arbitrary small time delay is introduced into the feedback loop. We will simulate this phenomenon to see how it happens. To understand the reason for the instability, it helps to plot both the tip velocity and the tip displacement. To calculate the velocity profile at any point is easy. After the expression of $U(x, s)$ is obtained, $sU(x, s)$ will be the Laplace transform of the velocity at any point x.

Since the simulation procedure is the same as in Sec. II-A and Sec. II-B, only the simulation results will be presented. We can see from Fig. 7 and Fig. 8 that the controller is working 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. Although this phenomenon has been discovered for more than ten years, no solution has been proposed so far. In the subsequent paper [12], a new boundary controller is presented to solve this problem with the help of the simulation method developed in this paper.

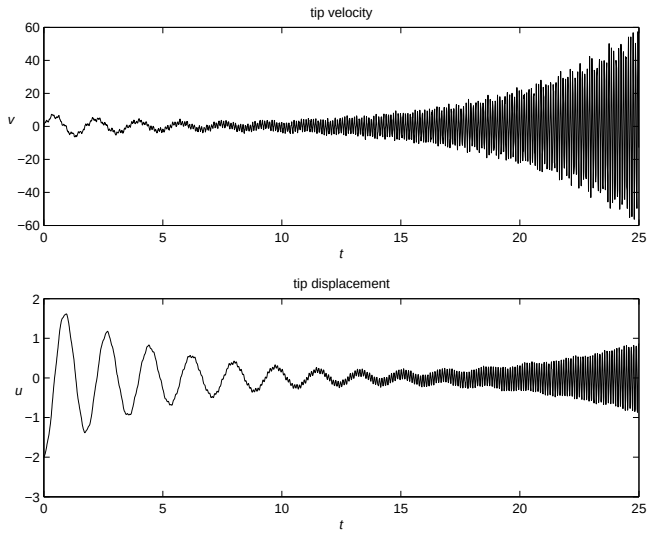

Fig. 7. Tip velocity and displacement, $\theta = 0.05$

III. Concluding Remarks

A hybrid symbolic and numerical method based on MATLAB Symbolic Math Toolbox is developed in this paper to simulate the typical boundary control problems. For illustration and validation, three different boundary control problems are simulated using the proposed method. The

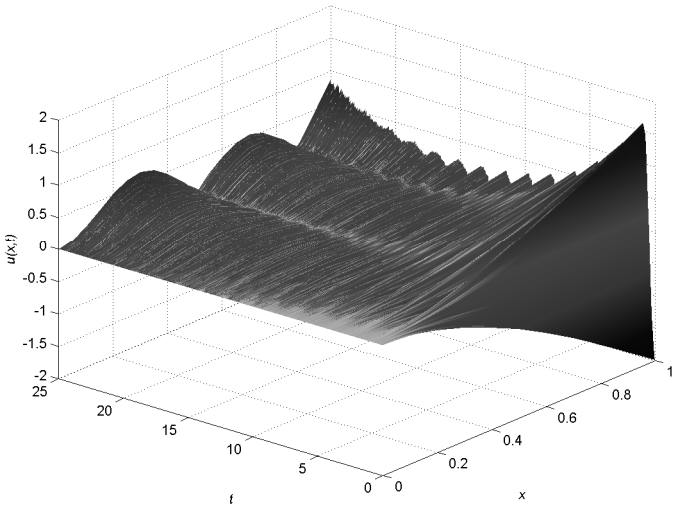

Fig. 8. Displacement of the whole beam, $\theta = 0.05$

proposed simulation method can be applied to a wide range of boundary control problems. Furthermore, the transfer function can be calculated in the intermediate steps of the simulation, which can be used to design some more advanced boundary controllers. We hope that this method is helpful for researchers to study the boundary control problems with easy simulation explorations in the future.

IV. REFERENCES

[1] Ömer Morgül, "An exponential stability result for the wave equation," *Automatica*, vol. 38, pp. 731–735, 2002.

[2] Ömer Morgül, "Stabilization and disturbance rejection for the beam equation," *IEEE Transactions on Automatic Control*, vol. 46, no. 12, pp. 1913–1918, 2001.

[3] Francis Conrad and Ömer Morgül, "On the stability of a flexible beam with a tip mass," *SIAM Journal of Control and Optimization*, vol. 36, no. 6, pp. 1962–1986, 1998.

[4] Ömer Morgül, "Stabilization and disturbance rejection for the wave equation," *IEEE Transactions on Automatic Control*, vol. 43, no. 1, pp. 89–95, 1998.

[5] Bao-Zhu Guo, "Riesz basis approach to the stabilization of a flexible beam with a tip mass," *SIAM J. Control Optim.*, vol. 39, no. 6, pp. 1736–1747, 2001.

[6] Bao-Zhu Guo, "Riesz basis property and exponential stability of controlled euler-bernoulli beam equations with variable coefficients," *SIAM J. Control Optim.*, vol. 40, no. 6, pp. 1905–1923, 2002.

[7] COMSOL Inc., `www.femlab.com`

[8] Comsol AB, *Partial Differential Equation (PDE) Tool Box User's guide*, The Mathworks, Inc., 2002.

[9] Andrew R. Mitchell, David Griffiths, *The Finite Difference Method in Partial Differential Equations* John Wiley & Sons, 1980.

[10] Alan Jeffery, *Advanced Engineering Mathematics*, Harcourt/Academic Press, 2002.

[11] The Mathworks, Inc., *Symbolic Math Toolbox User's Guide*, 2002.

[12] Jinsong Liang, YangQuan Chen and Bao-Zhu Guo "A new boundary control method for beam equation with delayed boundary measurement using modified smith predictors," in Proceedings of the IEEE Conference on Decision and Control (CDC) 2003, Hawaii, USA.

[13] Dean G. Duffy, "On the numerical inversion of Laplace transforms: comparison of three new methods on characeristic problems from applications," *ACM Transactions on Mathematical Software*, vol. 19, pp. 333–359, 1993.

[14] Lubomír Brančík, "Programs for fast numerical inversion of Laplace transforms in matlab language enviroment," in *Konference MATLAB ́99 ZCU Plzen*, 1999, pp. 27–39.

[15] G. Chen, M. C. Delfour, A. M. Krall, and G. Payre, "Modelling, stabilization and control of serially connected beams," *SIAM J. Contr. Optimiz.*, vol. 25, pp. 526–546, 1987.

[16] Aslam Kassimali, *Structural Analysis*, PWS Publishing, 1999.

[17] R. Datko, J. Lagnese, and M. P. Polis, "An example on the effect of time delays in boundary feedback stabilization of wave equations," *SIAM J. Control Optim.*, vol. 24, pp. 152–156, 1986.

[18] R. Datko, "Two examples of ill-posedness with respect to small time delays in stabilized elastic systems," *IEEE Transactions on Automatic Control*, vol. 38, no. 1, pp. 163–166, 1993.

APPENDIX

The code and its demo can be downloaded from `http://mechatronics.ece.usu.edu/jinsong/boundsim.zip`.

```
% WAVE_TF calculating the transfer function of boundary control of wave
% equation
%
% function U_xs = wave_tf(m, d, k, w, w_noise u_0, v_0)
%
% m:    tip mass, currently only m = 0 is tested
% d:    controller gain, refer to Omer Morgul, "Stabilization and
%       disturbance rejection for the wave equation", IEEE
%       Transactions on Automatic Control, Vol. 43, No. 1,
%       pp. 89-95, 1998
% k:    controller gain
% w:    controller parameter
% w_noise:  frequency of the noise, assume noise = cos(w_noise*t)
% 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:  jinsongliang@cc.usu.edu
%                       yqchen@ieee.org
% Last Modified: 02/16/2003

function U_xs = wave_tf(m, d, k, w, w_noise, u_0, v_0)

syms s x C1 C2

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

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

%keyboard
eq1 = strcat(char(subs(U_xs_ud, x, 0)), '=0');
eq2 = simple(subs(dU_ud, x, 1) + (d + k*s/(s^2 + w^2))* ...
    (s*subs(U_xs_ud, x, 1) + 0.5) + s/(s^2 + w_noise^2));
eq2 = strcat(char(eq2), '=0');

[C1, C2] = solve(eq1, eq2, 'C1', 'C2');

U_xs = subs(U_xs_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;
% end of wave_tf.m

% Main program to simulate the examples in
% Omer Morgul, "Stabilization and disturbance rejection for
% the wave equation", IEEE Transactions on Automatic Control,
% Vol. 43, No. 1, pp. 89-95, 1998
% Copyright:    Jinsong Liang and YangQuan Chen
% Last modified: 02/15/2003

clear all
syms x

% initialization
PI = sym(pi);
m = 0;          % tip mass, only m=0 is tested
d = 1;          % controller gain
k = 10;         % controller gain
%k = 0;
w = 10;         % another controller parameter
w_noise = 10;   % frequency of noise, assume noise = cos(w_noise*t)
u_0 = -0.5*sin(PI/2*x);     % initial displacement condition
v_0 = 0;                    % initial velocity condition
t_sim = 20;         % simulation time
steps_x = 20;       % number of points in x direction for simulation
% end of initialization

wave_tf(m, d, k, w, w_noise, u_0, v_0);

clear x;    % syms x is not needed any more
global x;   % global parameters needed in function F_lap.m,
            % generated in beam_smith.m

x = 1;
[u_xt_new, t] = nilt('F_lap', t_sim);

figure
plot(t, u_xt_new)
axis([0, 20, -0.6, 0.2])
xlabel('\ittime')
ylabel('\itu(1,t)')

% for some unknown reasons, displacement at x=0, which should be zero
% at all times, can not be simulated, so we choose a very small x
u_xt = [];
for x = [0.0001, 1/steps_x:1/steps_x:1],
    [u_xt_new, t] = nilt('F_lap', t_sim);
    u_xt = [u_xt, (real(u_xt_new))'];
end

x = [0.0001, 1/steps_x:1/steps_x:1];
figure
[X, T] = meshgrid(x, t);

surf(X(2:end,:), T(2:end,:), u_xt(2:end,:), 'FaceColor','interp', ...
    'EdgeColor','none', 'FaceLighting','phong')
%daspect([5 5 1])
%axis tight
view(-50,30)
camlight left

%surf(X, T, u_xt);
%grid on;
xlabel('x');
ylabel('time')

% 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);
% In boundary control problems, if the solution includes very high
% frequency components, such as boundary control of beam equation with
% time delays in the tip velocity measurement using static controller
% only, experience shows M should be at least 2048. Otherwise M = 1024
% should be enough. The values of alfa and P are not tuned so far.
% Commented by Jinsong Liang

%alfa=0; M=256; P=2;
%alfa=0; M=512; P=2;
alfa=0; M=1024; P=2;
%alfa=0; M=2048; 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);
```