Georgia Institute of Technology
Mechanical Engineering Department
ME 6404: Advanced Control System Design and Implementation
Dr. Singhose

Lab Report # 6:

Linear Stage Control

Amir Shenouda
Matt Kontz
Joe Frankel
November 22, 2003

Introduction

Evaluating the performance of a control system in a real world environment during the design phase provides critical information about the control system and the plant. System models can be used to gain insight and parameters can be tuned based on simulation results. Rapid control prototyping is a very popular technique for testing control system prototypes on the actual physical hardware or the plant before final implementation. Modern control systems, designed to be optimal in performance with respect to some specified criterion, robust to changes in system and environmental parameters and in some cases adaptive in operation, are usually complex in their structure and require extensive simulation and testing before final implementation. By quickly implementing a control system in software (in the form of code generated from the simulation model) and investigating their operation with physical systems, their effectiveness in real world operating conditions can be studied up front and trade-off decisions relating to their design can be made more prudently. The controller algorithm is implemented in software, which makes changing its parameters an easy task and thus has the potential of reducing the overall design cycle time of the controller at a much lower cost. Recent advances in real time computing software and hardware have made integration of hardware components and software control algorithms fast and easy.

In this project, rapid prototyping was used on an Anorad Linear stage to control a flexible two mass system representing a pick-and-place robot. Many software platforms are available for real-time control implementation. Here MATLAB/ Simulink, xPC Target, and Real-Time Workshop were used.

In the xPC Target system, typically the control system is first designed on a host PC, and simulations in a MATLAB/Simulink environment are necessary as a preliminary step in the design. The xPC target software module available in MATLAB helps establish TCP/IP communication between the host PC and a target PC that is directly connected to the hardware through I/O interface cards. Thus, the host is used to design control schemes and then compile the algorithms into C code, suitable for execution on the embedded controller (in target PC) in real-time.

For this project, a flexible two-mass system is connected to the linear stage as illustrated in figure (1). This situation represents a pick-and-place robot encountered often in industry. During operation, the tip mass alternates between a “loaded” and “unloaded” condition as it picks up a part, moves it to another location, releases the part, and then moves back to the original location for another part. It is necessary for the robot to operate with maximum speed and minimum tip vibration to reduce the time needed to complete the process, and to reduce the stresses in the robot by minimizing relative deflection between the base and tip masses. For this project, input shaping has been applied to minimize both tip mass vibration and relative deflection during point-to-point movement for both the unloaded and loaded cases.

Objective

Design a control system for the Anorad linear stage in Love 225 that will move a flexible load in point-to-point fashion as fast as possible while minimizing tip vibration and deflection.

Overall System Setup

The flexible two-mass system mounted to the Anorad linear stage is illustrated in figure (1). Thin aluminum plates join the two masses, which create a flexible mode when the base mass is displaced.

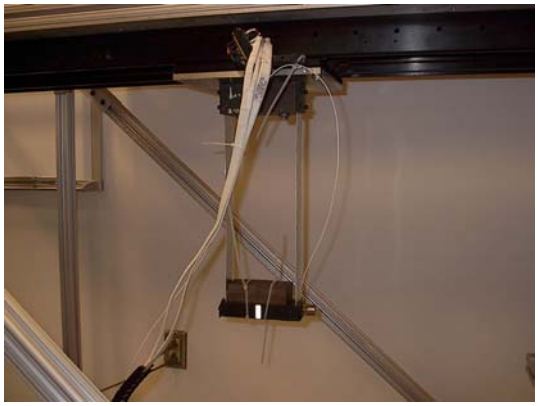


Figure 1: base, tip, and springs

Figure 2: linear track setup

An Anorad DC Brushless Linear Motor is mounted on a Parker aluminum frame, shown in figure (2), which provides actuation for the two masses, hereafter referred to as the base and tip. The motor interfaces with an amplifier shown in figure (3), and the target PC, which is connected to a host PC (a typical xPC target setup), shown in figure (4).

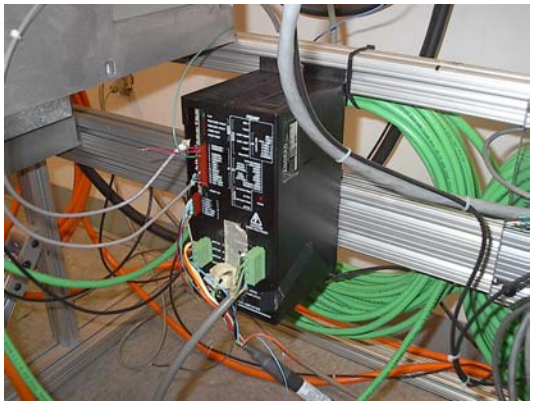


Figure 3: Amplifier

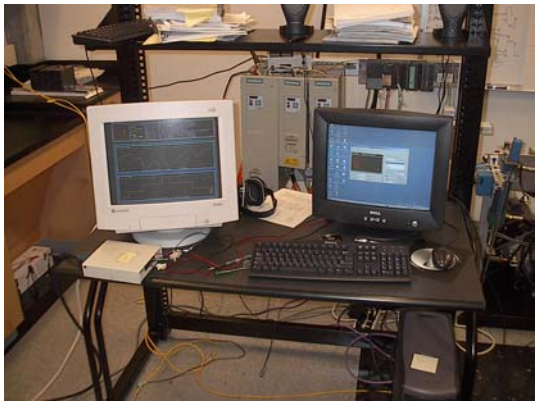


Figure 4: Target PC (L) & Host PC (R)

Position feedback of the base is provided by an encoder along the length of the track. An accelerometer is attached to the tip mass to quantify vibration, but is not used for feedback.

Hardware

Hardware setup was a major undertaking in this project. There were many hardware components that had to be connected together, work synchronously, and communicate with other components in the system. The xPC Target setup had to control of all these components and provide for the algorithm that would control the linear stage and the two masses. The following is a list of the hardware components and some of the problems faced.

- **Linear Stage and Encoder:** Anorad LW-10 Linear Stage. This is based on the LCE-S DC Brushless Linear Motor. This motor has a steel core, a flat design with natural cooling, low cogging, low magnetic attraction, and is excellent for general automation. The encoder on the LW-50 Linear stage has a resolution of 50 lines per millimeter, which outputs a sinusoidal signal that is passed through a 5x logic encoder multiplier. This results in a quadrature signal with 1μm resolution that is passed to an encoder card. The PC-QUAD04 encoder card used in the computer running xPC-target has the ability to pulse at 1.25MHz. This means that the maximum stage speed that this encoder card can handle is1.25m/s compare to 1.5m/s which is the actual hardware limit. When the encoder was tested in real-time using xPC-target, the encoder malfunctioned at a much lower rate than 1.25m/s. After research, it was discovered that the original xPC target drivers only pulsed at 4kHz. Luckily this problem was fixed by downloading a new xPC-target driver for the PC-QUAD04 encoder card from Mathworks.
- **Amplifier:** Anorad High Voltage Brushless DC Servo Amplifier-Sine Hall Mode. The amplifier directly controls the motor. Once the amplifier is enabled, it sends 3-phase current to the motor proportional to the command signal. This current is then converted to DC, which powers the motor and applies a force on the base to move it. It also receives a sine-hall effect sensor signal from the linear stage. The amplifier has 15 switches, 9 potentiometers, and 12 signal lines that have to be configured for correct operation. For example, one of the potentiometers is used to adjust the drift in the base. The switch combination defines how the input signals are processed.
- **Host/Target PC's:** The control system is a typical Host/Target PC setup. The control algorithm is designed on the Host PC in Simulink and then downloaded to the Target PC, which interfaces directly with the hardware. The Host PC is also used to tune parameters, monitor performance, and make any necessary changes in the design. The Target PC interfaces with the amplifier and accelerometer through a National Instruments PCI-6052E A/D card. The encoder is read with a Measurement Computing PCI-Quad 04 Four Channel Quadrature Encoder card.
- **Accelerometer:** A PCB Piezotronics accelerometer was placed on the tip mass to quantify vibration. The accelerometer was connected to both a Tektronix oscilloscope to determine the vibratory characteristics of the tip mass, and also to the A/D card in the target PC to capture the acceleration data in MATLAB.

Preliminary System ID and Feedback Controller Design

Once all the hardware and software was debugged to a point where open loop control was possible, a simple model was used to design a feedback controller. By moving the stage up and down the track, it was observed that there was significant viscous friction in the track. Therefore our base line controller was simply a mass damper model.

$$\frac{Y_b(s)}{u_V(s)} = \frac{K_m}{Ms^2 + bs} \tag{1}$$

The model parameters M and b, and K_m in equation (1) were estimated by applying a constant voltage to the amplifier, recording the output from the encoder, and then running a MATLAB minimization routine to minimize the error between the data and the model. The data and model prediction are illustrated in Figure 5.

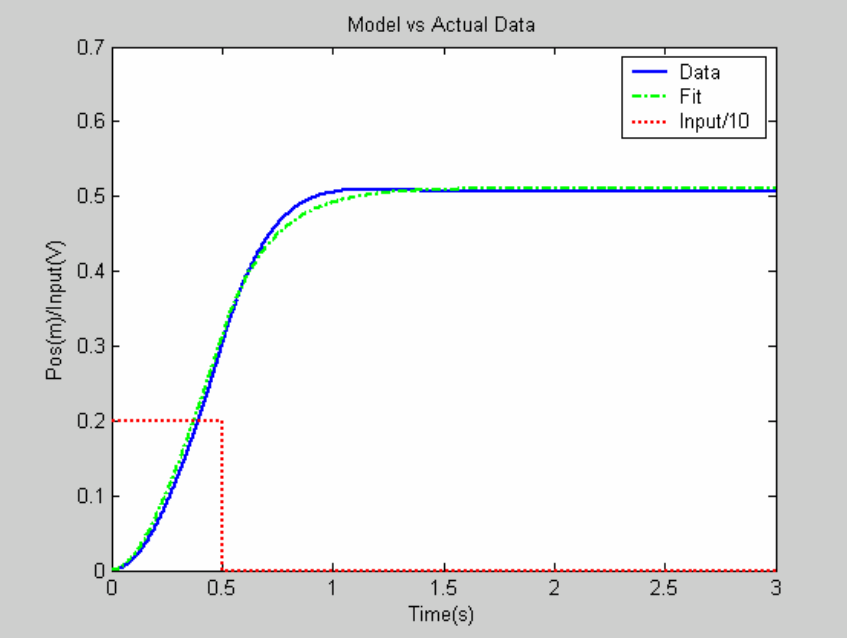


Figure (5): Actual data and model fit data.

This simplified model ignores flexibility, but can still be used to design a feedback controller. This is essentially an estimate of either the base position or the rigid body mode of the system. Table 1 shows the estimated parameters.

Table 1 Parameters in preliminary model

Variable	Description	Value
K_m	amplifier/force gain	31N/V
M	total mass	13kg
b	damping	61N*s/m

For the compensator design, a PD controller was chosen in order to speed up the dynamics and add damping to the system. Initially the design was performed in the s-plane, but soon moved to the z-plane in order to accurately predict the pole locations in the digital domain since this controller was to be implemented digitally at 1000Hz. This was an important step because it allowed a controller to be designed before a more detailed model of the system was developed. The z-domain PD controller is given by equation (2).

$$C(z) = \frac{U_V(z)}{E(z)} = \frac{2750(z - 0.95)}{z} \tag{2}$$

Once the feedback controller was functional, it was possible create controlled point-to-point motions and excite the flexible mode. It was also found later that the root locus has a minimal interaction with the poles of the flexible mode. Figure (5) illustrates the root locus of the rigid body system. The closed loop poles are denoted by '+' while the open loop poles and zeros use the traditional 'x' and 'o'. As can be seen from this root locus, it is possible to choose the natural frequency and damping of the dominant dynamics by setting the location of the zero and adjusting the gain. The closed loop pole contributed by the PD controller ends up very close to the origin, meaning that is extremely fast and has little affect on the system. The final closed-loop poles were chosen relatively slow to avoid excessive actuator saturation.

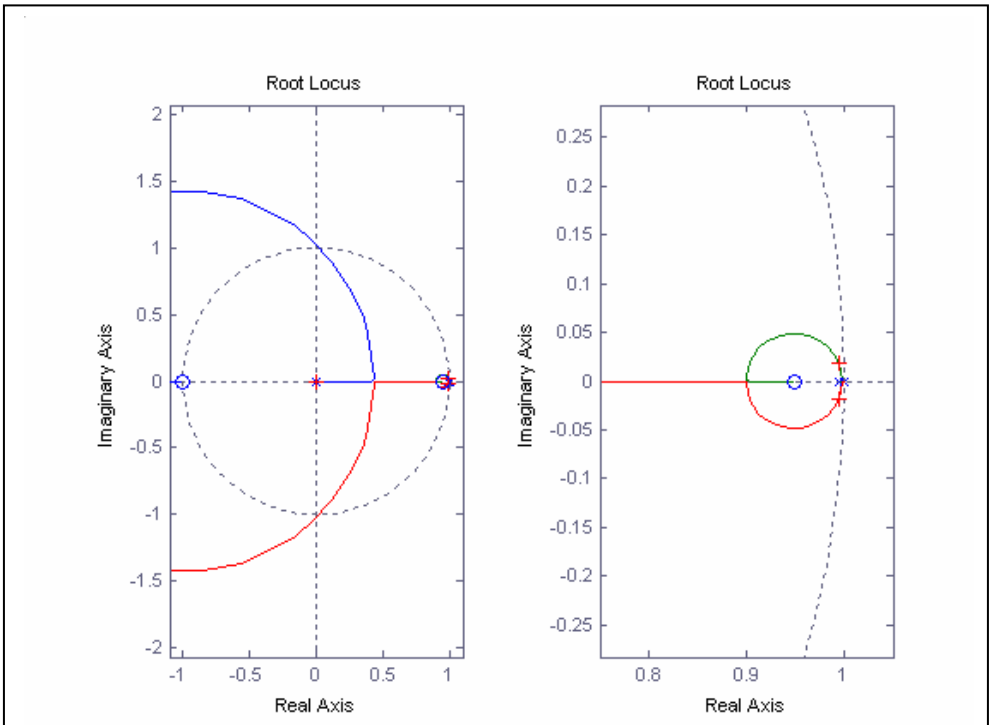


Figure (5): Actual data and model fit data.

Final System Identification

The next step involved creating a more complete system model to include the flexible connection between the two masses. The complete system is illustrated in figure (6). The state-space dynamic model of the open-loop plant is given in equation (3).

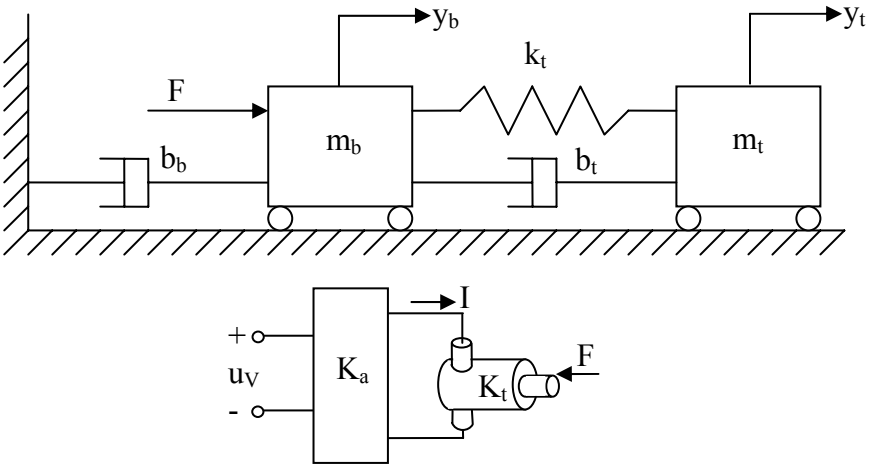


Figure (6): Linear Track Model

$$\begin{bmatrix} \dot{y}_b \\ \ddot{y}_b \\ \dot{y}_t \\ \ddot{y}_t \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{-k_t}{m_b} & \frac{-(b_b+b_t)}{m_b} & \frac{k_t}{m_b} & \frac{b_t}{m_b} \\ 0 & 0 & 0 & 1 \\ \frac{k_t}{m_t} & \frac{b_t}{m_t} & \frac{-k_t}{m_t} & \frac{-b_t}{m_t} \end{bmatrix} \begin{bmatrix} y_b \\ \dot{y}_b \\ y_t \\ \dot{y}_t \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_a K_t}{m_b} \\ 0 \\ 0 \end{bmatrix} u_V \qquad (3)$$

$$\begin{bmatrix} y_b \\ y_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} y_b \\ \dot{y}_b \\ y_t \\ \dot{y}_t \end{bmatrix}$$

The flexible system model in equation (3) was identified by sending a ramp input (desired position) to the real system and recording the base position from the encoder. The parameters of the flexible system shown in figure (6) were then identified using Matlab's fmincon routine to find the optimal values of m_b, m_t, b_b, k_t, b_t, and $K_m=K_aK_t$ by comparing the actual data with the model output and minimizing the error between the two. The estimated plant parameters are given in Table (2).

Table (2): Optimized model plant parameters

Parameter	Value	Unit
Base mass m_b	12.8	Kg
Tip mass m_t (unloaded)	1.5	Kg
Base damping b_b	45*	N·s/m
Spring rate k_t	333	N/m
Spring damping b_t	3.2	N·s/m
Amp/motor gain $K_m=K_aK_t$	13	N/V

* *approximate: depends on base velocity*

The resulting model predicted the actual response quite well, as illustrated in figure (7). However, the friction b_b between the base and the track was found to be nonlinear due to stiction and other unmodeled effects, and therefore the accuracy of the model was not perfect. Since the accelerometer signal was used only for vibration measurement and not for system ID or feedback control due to noise-induced integration error, measured tip position was not available.

Note that the closed loop system was 5th order with two oscillatory modes, since the phase-lead compensator drove two of the plant poles off the real axis and added another pole near the origin in the z-plane as shown in figure (5).

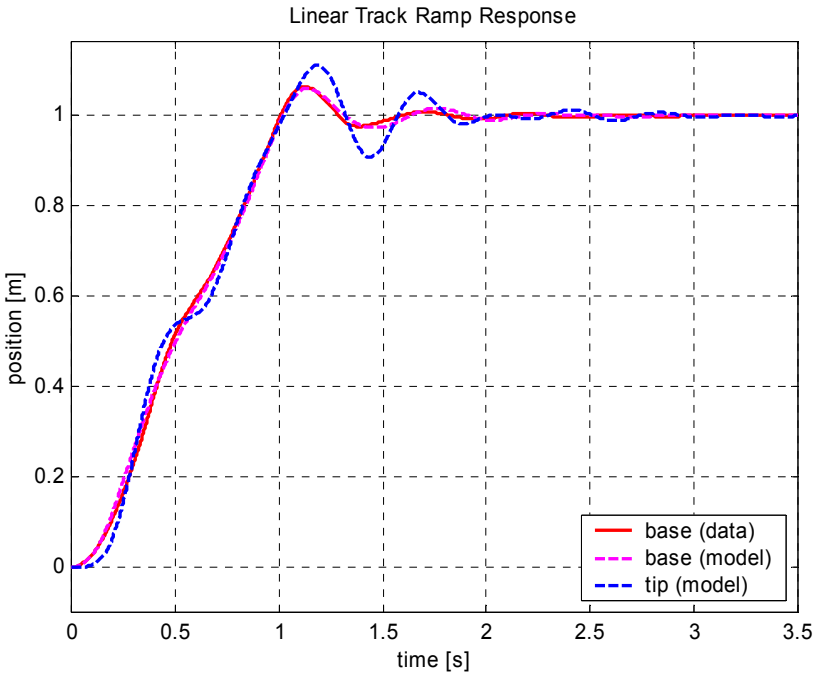


Figure (7): Comparison of data with model predicted response

Closed-Loop Natural Frequency and Damping Ratio

For the first attempt at vibration reduction, a ZVD command shaper was designed based on the eigenvalues of the closed-loop system model. However, due to modeling errors stemming from the nonlinear friction in the track, this shaper did not completely eliminate vibration when tested on the real system.

A more direct approach was then implemented to measure the natural period and damping ratio by using the accelerometer, by recording the free response to an initial tip displacement with the compensator running. Natural periods and log decrements were then obtained directly from the accelerometer time traces. Data was collected for both the unloaded tip and nominally loaded tip cases, resulting in natural frequencies at 2.50 and 1.83 Hz, respectively. The time trace from the accelerometer, filtered at 20Hz, is shown in figure (8).

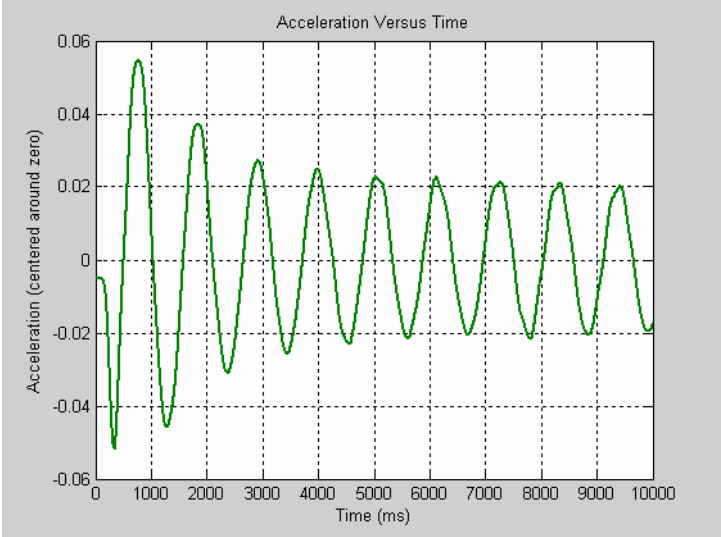


Figure (8): Free response acceleration versus time data used to determine damped frequency and damping ratio for the two-mass system.

The formulae used to obtain the damping ratio are given in equations (4) and (5), where x_n is the wave amplitude at cycle n and δ is the log decrement. The measured natural frequencies and damping ratios for the unloaded and loaded tip cases, obtained from the accelerometer data, are given in table (3).

$$\delta = \ln\left(\frac{x_n}{x_{n+1}}\right) \tag{4}$$

$$\zeta = \frac{\delta}{\sqrt{(2\pi)^2 + \delta^2}} \tag{5}$$

Table (3): Natural Frequency and Damping Ratio

Tip loading	**Natural Frequency f_d [Hz]**	**Damping Ratio ζ**
Unloaded	2.500	0.0178
Fully Loaded	1.829	0.0582

It can be seen in figure (8) that the magnitude of the free response acceleration decreases initially, but then flattens out after about three cycles. It was observed that this change in the exponential decay envelope occurred when the base mass stopped oscillating, but while the tip was still oscillating. This can be explained by noticing that while the base is still moving, vibrations are damped by the friction in the track, but when the base stops, the only damping left is the structural damping in the spring which is very small.

Input Shaping

Once the natural frequencies and damping ratios had been measured, a ZV/ZVD convolved shaper was designed for the two measured natural periods corresponding to the unloaded and loaded tip. First, the number of time steps k_1, k_2, and k_3 corresponding to T_d and $T_d/2$ were calculated based on the sampling frequency and data in table (3) for both cases, and the pulse amplitudes were adjusted based on the damping ratios. A ZV shaper was designed at the unloaded-tip natural frequency because this frequency was well known and not expected to change. A ZVD shaper was designed at the nominally-loaded-tip natural frequency, because it was desired that the system be robust to variable tip loads. Equations (6-8) illustrate the multimode ZV/ZVD shaper design in discrete transfer function format.

$$\begin{aligned} ZV(z) &= \frac{\begin{bmatrix} A_{11}z^{-k_{11}} & 0 & \cdots & 0 & A_{12}z^{-k_{12}} \end{bmatrix}}{1} \cdot \frac{z^{k_{12}}}{z^{k_{12}}} \\ &= \frac{\begin{bmatrix} A_{11}z^{k_{12}-k_{11}} & 0 & \cdots & 0 & A_{12} \end{bmatrix}}{\begin{bmatrix} z^{k_{12}} & 0 & \cdots & \cdots & 0 \end{bmatrix}} = \frac{ZV_{num}}{ZV_{den}} \end{aligned} \tag{6}$$

$$\begin{aligned} ZVD(z) &= \frac{\begin{bmatrix} A_{21}z^{-k_{21}} & 0 & \cdots & 0 & A_{22}z^{-k_{22}} & 0 & \cdots & 0 & A_{23}z^{-k_{23}} \end{bmatrix}}{1} \cdot \frac{z^{k_{23}}}{z^{k_{23}}} \\ &= \frac{\begin{bmatrix} A_{21}z^{k_{23}-k_{21}} & 0 & \cdots & 0 & A_{22}z^{k_{23}-k_{22}} & 0 & \cdots & 0 & A_{23} \end{bmatrix}}{\begin{bmatrix} z^{k_{23}} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \end{bmatrix}} = \frac{ZVD_{num}}{ZVD_{den}} \end{aligned} \tag{7}$$

$$ZVZVD(z) = \frac{R(z)}{Y_d(z)} = \frac{conv(ZV_{num}, ZVD_{num})}{\begin{bmatrix} z^{k_{12}+k_{23}} & zeros(1, k_{12}+k_{23}-1) \end{bmatrix}} \tag{8}$$

The numerators of the ZV and ZVD shapers in equations (6) and (7) were convolved and the resulting input shaper in equation (8) was implemented in real-time using the Simulink discrete transfer function block. Using the sampling rate of 1 kHz, the resulting multi-mode ZV/ZVD shaper contained 6 pulses over 747 ms, which means that the numerator and denominator vectors in the shaper each contained 747 elements, with six nonzero terms in the numerator and one nonzero term in the denominator. This approach was easy to implement and worked well in real-time.

The equivalent continuous-time block diagram of the final control system is illustrated in figure (9).

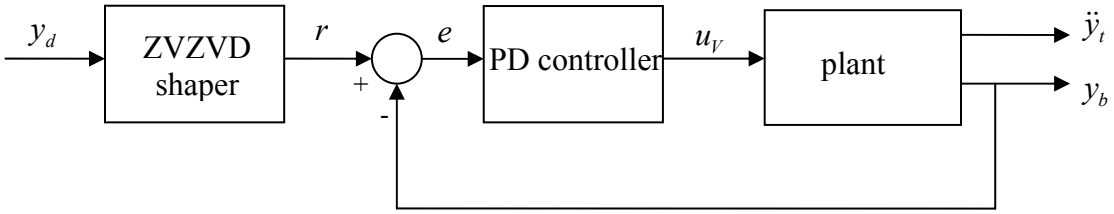


Figure (9): Control system block diagram

Input Shaping Results

The effectiveness of the input shaper was evaluated by comparing the system response to a ramp input for ***unloaded*** tip conditions with and without input shaping. Figure (10) is a plot of the collected position data for both cases, and figure (11) compares the tip vibration measured with the accelerometer. Note that the accelerometer data was digitally filtered at 20Hz to get rid of sensor noise and it was multiplied by a constant to obtain position. Figures (12) and (13) illustrate the equivalent situations with a ***loaded*** tip. As can be seen from figures (10-13), input shaping significantly reduced tip deflections and vibrations for both unloaded and loaded tip conditions. Video footage was also recorded to document the effectiveness of input shaping.

The reason that vibration was not completely eliminated was most likely due to nonlinearities such as the damping in the base, or other unmodeled factors, such as the natural mode of the support structure, which was observed to during testing to occur around 9-10 Hz.

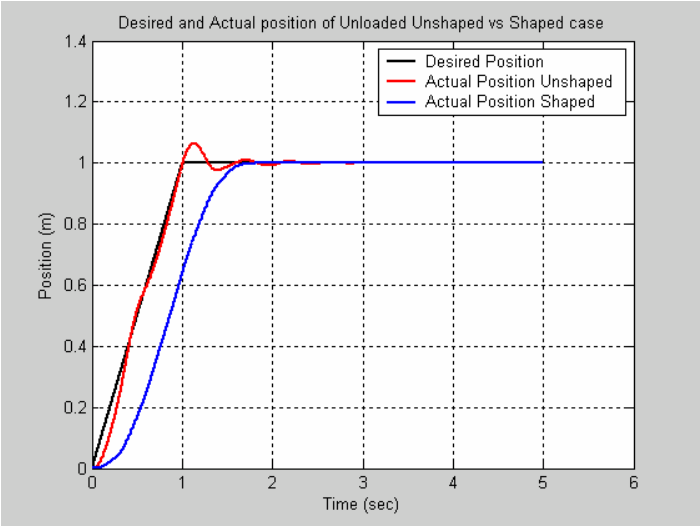


Figure (10): Desired and measured position of unloaded unshaped and shaped cases.

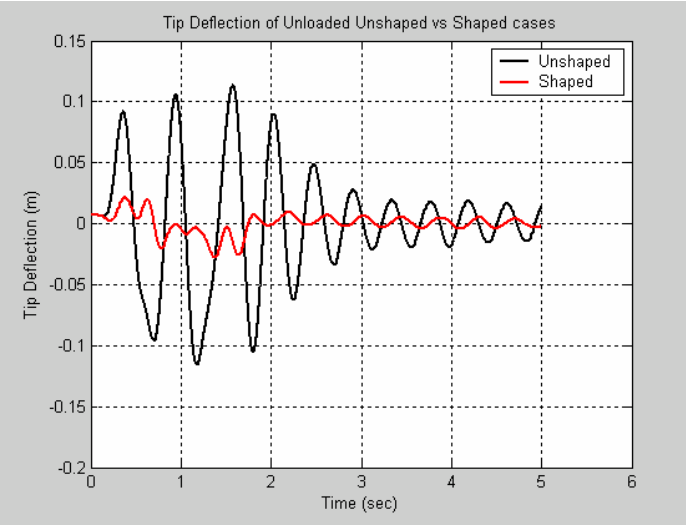


Figure (11): Tip deflection for unloaded unshaped and shaped cases.

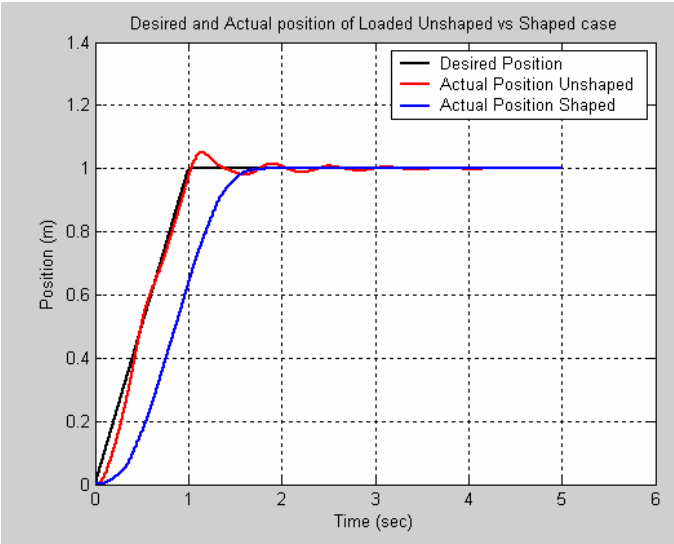


Figure (12): Desired and measured position of loaded unshaped and shaped cases

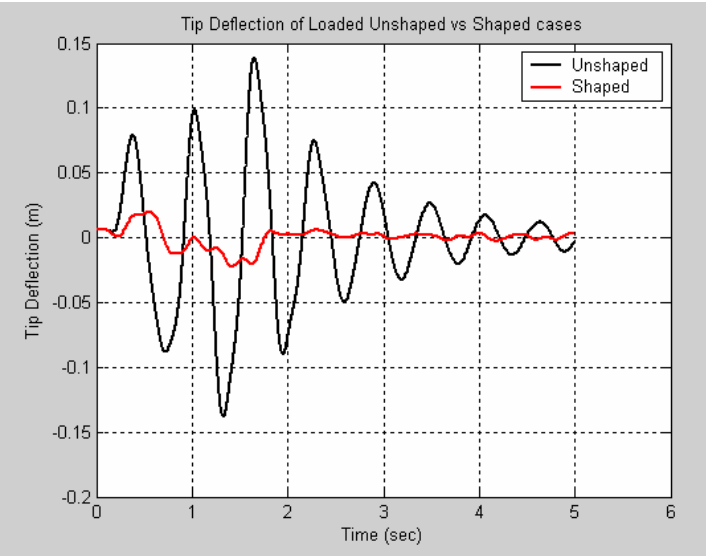


Figure (13): Tip deflection for loaded unshaped and shaped cases.

Robustness to tip load variation

Tests were conducted to evaluate the shaper's ability to remove vibration with an unloaded tip, nominally loaded tip, and fractions of the nominal load. The tip mass position time traces for variable tip loading conditions are shown in figure (14). The vibration sensitivity to tip load variation is shown in figure (15) for both the data given in figure (14) and the vibration predicted by the model. In all cases, vibration was defined as the largest tip deflection relative to the final desired position just after the end of the ramp. The baseline unshaped vibration reference was computed to be 10.1cm from the accelerometer data after calibration. As can be seen from the plot, the shaper removed at least 94% of vibration for all loading conditions.

Conclusion

In this project a feedback control system and input shaping scheme was designed and implemented. The goal was to achieve point-to-point motion similar to a manufacturing setup were an object would be picked up and moved to a different location. Tests were conducted to show that tip deflection and vibration was reduced over a range of loads using the input shaping scheme. The cost of the input shaping was an increase in rise time; however, the settling time is actually faster for the shaped case.

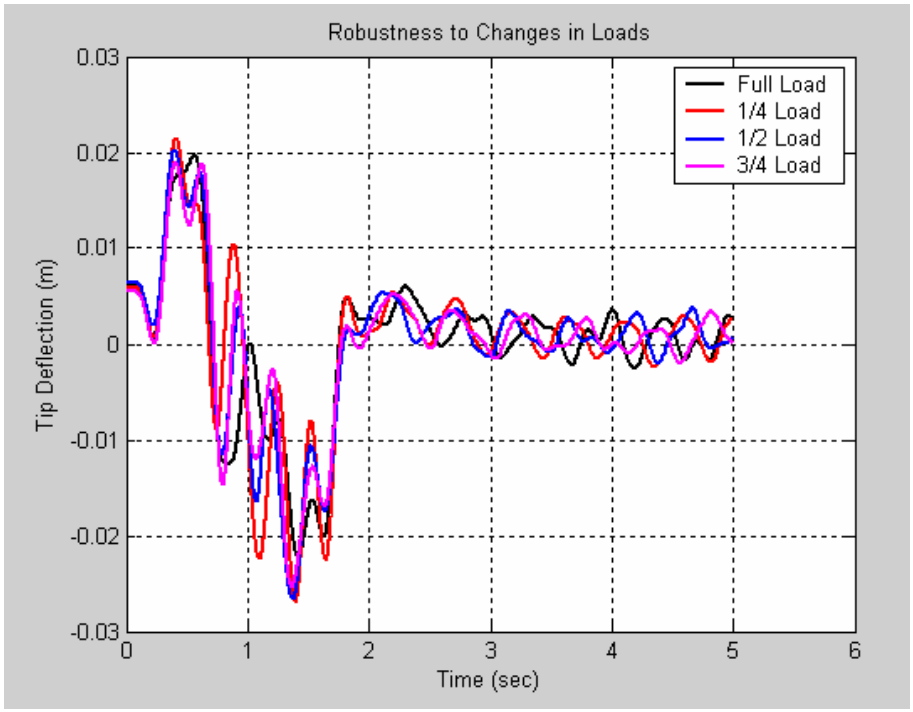


Figure (14): Tip deflection comparison for different loads. Robustness to changes in load can be noticed.

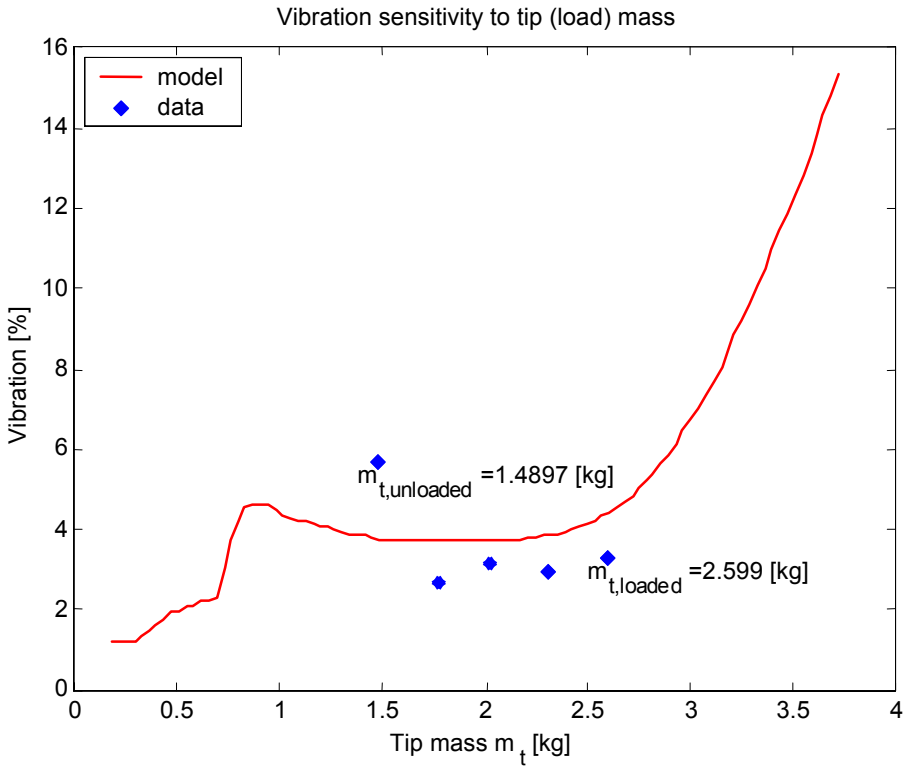


Figure (15): Tip vibration sensitivity to load variation

APPENDIX: Matlab/Simulink Code

System ID Routine

```
% ME 6404
% Lab 6 - Final Project
% Amir Shenouda
% Matt Kontz
% Joe Frankel

% This file computes the 4th order model from a minimization between
% the data and theoretical model

clear all ; close all ; clc ;

global t y yd N Ts Cz

load finaldata

% Time vector
Tf=3.5;
Ts=0.001;
N=floor(Tf/Ts)+1;
t=[0:Ts:Tf]';

% Estimated system parameters
Kt=9.9;               % motor gain [N/A]
Ka=1.35;                 % amp gain [A/V] (could be 0-5)
Km=Kt*Ka;             % overall amp/motor gain [N/V]
mb=8;              % base mass [kg]
mt=2;                 % tip mass [kg]
bb=50;                % track-base damping [N*s/m]
bt=1;                 % base-tip damping [N*s/m]
kt=3400;              % spring rate [N/m]

% compensator
Cz=tf(2750*[1 -0.95],[1 0],Ts);

% Use estimated parameters for initial guesses and set limits
X0=[mb   mt    bb    bt    kt    Km ];
LB=[7.0  1.48  0     0     300   1  ];
UB=[30.0 1.49  45    10    390   100];

% Experimental data
yd=unluns(1:N,1);
y=unluns(1:N,2);
uV=unluns(1:N,3);
auf=unluns(1:N,4);

% Optimization parameters
options.MaxIter=500;
options.MaxFunEvals=1000;
options.Display='iter';
options.TolFun=1e-8;
options.TolX=1e-8;

% Estimate model parameters based using numerical optimization
```

```
%X=fminsearch('pos_error',X0,options);
X=fmincon('pos_error',X0,[],[],[],[],LB,UB,'nonlcon',options);

% Return system parameters from optimization results
mb=X(1)
mt=X(2)
bb=X(3)
bt=X(4)
kt=X(5)
Km=X(6);

Ka=Km/Kt
kt_eng=kt*0.2248089*0.0254    % spring rate in english units [lbf/in]

% Set up system
Gz=linear_track(X,Ts);
SYScl=feedback(Cz*Gz,1,1,1);

% Dynamic parameters
e=eig(d2c(SYScl));
sig=real(e(2));
wd=imag(e(2));
z=(-sig^2+sqrt(sig^4+4*wd^2*sig^2))/(2*wd^2)
wn=wd/sqrt(1-z^2);
Td=2*pi/wd;
fd=1/Td

% Simulate system
out=lsim(SYScl,yd,t);
ybase=out(:,1);
ytip=out(:,2);

% Plot results and compare with experimental data

figure
plot(t,uV,'r','linewidth',2)
xlabel('time [s]')
ylabel('input voltage [V]')
legend('input voltage [V]')
grid on

figure
plot(t,y,'r','linewidth',2)
hold on
plot(t,ybase,'m--','linewidth',2)
plot(t,ytip,'b--','linewidth',2)
legend('base (data)','base (model)','tip (model)',4)
grid on
xlabel('time [s]')
ylabel('position [m]')
title('Linear Track Ramp Response')
ylim([-0.1 max(y)+0.1])

figure
plot(t,(ytip-ybase)*1000,'r','linewidth',2)
xlabel('time [s]')
ylabel('relative position [mm]')
```

```
title('Tip Vibration')
grid on

%SYStf=tf(SYS);
%BaseTF=tf(SYStf.num(1),SYStf.den(1))
%TipTF=tf(SYStf.num(2),SYStf.den(2))

save Trackmodel SYScl wd z Ts Gz Cz X
```

```
function Gz=linear_track(X,Ts)

% this function returns a discrete time LTI model of the linear track
% based on input parameters

mb=X(1);
mt=X(2);
bb=X(3);
bt=X(4);
kt=X(5);
Km=X(6);

% Set up system
A=[0 1 0 0;
   -kt/mb -(bt+bb)/mb kt/mb bt/mb;
   0 0 0 1;
   kt/mt bt/mt -kt/mt -bt/mt];
B=[0 Km/mb 0 0]';
C=[1 0 0 0;
   0 0 1 0];
D=[0;0];
Gs=ss(A,B,C,D);
Gz=c2d(Gs,Ts);
```

```
function F=pos_error(X)

% This function computes the error between the data and the theoretical model
% over the course of the simulation

global t y yd N Ts Cz

% compute system model for these parameters
Gz=linear_track(X,Ts);
SYScl=feedback(Cz*Gz,1,1,1);

% simulate system using input data
out=lsim(SYScl,yd,t);
yth=out(:,1);

% Sum the total error between data and simulation output vectors
% This is the quantity to minimize in the optimization
F=0;
for i=1:N
   F=F+(y(i)-yth(i))^2;
end
```

```
function [c,ceq]=nonlcon(X)
```

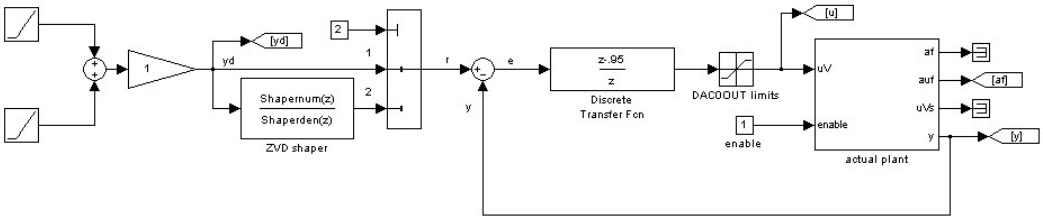
```
% This function defines nonlinear constraints for the optimization routine

global t y yd N Ts Cz

% compute system model from input parameters
Gz=linear_track(X,Ts);
SYScl=feedback(Cz*Gz,1,1,1);
%SYScl=tf(SYScl.num(1),SYScl.den(1))

% Compute damped natural frequency from system eigenvalues
e=eig(d2c(Gz));
sig=real(e(1));
wd=imag(e(1));
z=(-sig^2+sqrt(sig^4+4*wd^2*sig^2))/(2*wd^2);
wn=wd/sqrt(1-z^2);
Td=2*pi/wd;
fd=1/Td;

% Constrain natural frequency of model to what was observed
ceq=fd-2.5;
c=0;
```

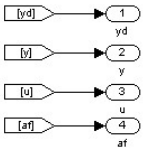
Shaper Design & Control System Setup Routine

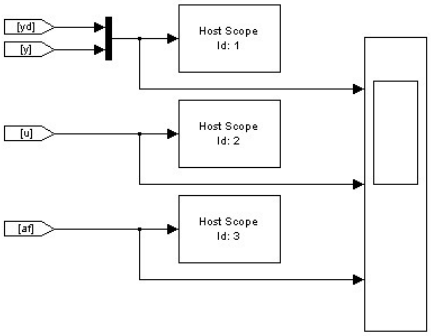
```
clear all ; close all ; clc ;

% This routine sets up the ZVZVD shaper and other variables necessary
% for running the control system

% plant model
load Trackmodel

% closed-loop plant dynamic parameters
% 1st element: unloaded tip
% 2nd element: nominally loaded tip
Td=[0.4 0.5467];
z=[0.0178 0.0582];

funload=1/Td(1)
fload=1/Td(2)

% ZV shaper for unload frequency
K=exp(-z(1)*pi/sqrt(1-z(1)^2));
k1=1
k2=floor((1/2)*(Td(1)/Ts))+1
A1=1/(1+K);
A2=K/(1+K);

ZVnum=zeros(1,k2);
ZVnum(k1)=A1;
ZVnum(k2)=A2;

ZVden=zeros(1,k2);
ZVden(1)=1;

% ZVD shaper for load frequency
K=exp(-z(2)*pi/sqrt(1-z(2)^2));
```

```
k1=1
k2=floor((1/2)*(Td(2)/Ts))+1
k3=2*k2-1
A1=1/(1+K)^2;
A2=2*K/(1+K)^2;
A3=K^2/(1+K)^2;

ZVDnum=zeros(1,k3);
ZVDnum(k1)=A1;
ZVDnum(k2)=A2;
ZVDnum(k3)=A3;

ZVDden=zeros(1,k3);
ZVDden(1)=1;

Shapernum=conv(ZVnum,ZVDnum);
Shaperden=conv(ZVden,ZVDden);

% Accelerometer Filter
fc=100;         % cutoff frequency
fN=2/Ts;
X=fc/fN;
[b,a]=butter(8,X);

w1=2*pi*funload;
w2=2*pi*fload;

save ZVD Shapernum Shaperden w1 w2
```

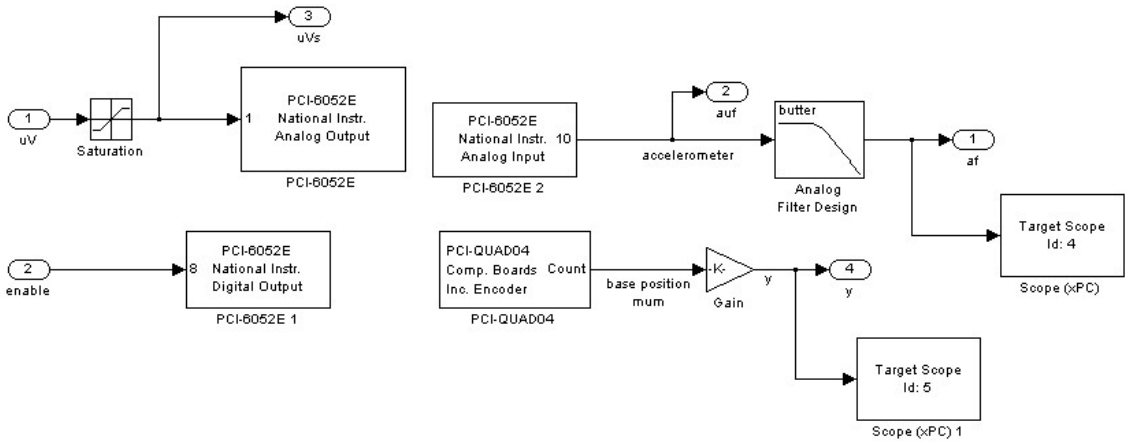
Simulink Control Program







Plant Subsystem



Accelerometer Calibration Routine

```
clear all ; close all ; clc ;

% calibrate accelerometer
load accel_data
t=data(:,1);
t=t-t(1);
V=fdata(:,1);
subplot(2,1,1);
plot(t,V)
xlabel('time [s]')
ylabel('voltage [V]')
title('Accelerometer Calibration')

x0=0.05; % initial displacement [in]
Td=0.43; % eyeballed from plot [s]
fd=1/Td
wd=2*pi/Td
V0=max(V);
Ka=x0*wd^2/V0

x=Ka*V/wd^2;

subplot(2,1,2)
plot(t,x,'r')
xlabel('time [s]')
ylabel('tip position [m]')
text(1.75,-0.025,strcat('Calib. constant: K_a=',num2str(Ka),' [m/s^2/V]'))
text(1.75,-0.040,strcat('Damped natural freq: f_d=',num2str(fd),' [Hz]'))

save accel Ka fd wd
```

Vibration Sensitivity Routine

```
% This file plots vibration sensitivity as a function of
% tip load mass, both experimental data and model predictions

clear all ; close all ; clc ;

% load model data
load Trackmodel
```

```
% load input data
load finaldata
yd=unluns(:,1);
kf=length(yd);
Ts=0.001;
t=[0:Ts:Ts*(kf-1)]';

% shape input data & create reference signal
load ZVD
Shaper=tf(Shapernum,Shaperden,Ts);
r=lsim(Shaper,yd,t);

% simulate unshaped response to compute baseline vibration
y=lsim(SYScl,yd,t);
yt=y(:,2);
Vib0=max(yt(1001:end)-1)

% ========== SENSITIVITY ANALYSIS ==========
n=100;
mt0=X(2);
mt=linspace(0.1*mt0,2.5*mt0,n);
for i=2:n

   % Set up system for this tip mass
   X(2)=mt(i);
   Gz=linear_track(X,Ts);
   SYScl=feedback(Cz*Gz,1,1,1);

   % simulate response for this tip mass
   y=lsim(SYScl,r,t);
   yt=y(:,2);

   % locate time when tip first passes reference
   flag=0;
   for k=1:length(r)-1
      if r(k)<1 & r(k+1)==1 & flag==0
         kc=k;
         flag=1;
      end
   end

   % vibration amplitude for this tip mass
   Vib(i)=max(yt(kc:end)-1);

end

% load accelerometer parameters
load accel

% digital filter design
fc=20;          % cutoff frequency
fN=2/Ts;
X=fc/fN;
[b,a]=butter(8,X);

% read in accelerometer data
```

```
V0=unlsh(:,4);
V1=lsh1(:,4);
V2=lsh2(:,4);
V3=lsh3(:,4);
V4=lsh4(:,4);

% filter accelerometer data
V0f=filter(b,a,V0);
V1f=filter(b,a,V1);
V2f=filter(b,a,V2);
V3f=filter(b,a,V3);
V4f=filter(b,a,V4);

% compute relative tip position
x0=detrend(Ka*V0f(2000:end))/wd^2;
x1=detrend(Ka*V1f(2000:end))/wd^2;
x2=detrend(Ka*V2f(2000:end))/wd^2;
x3=detrend(Ka*V3f(2000:end))/wd^2;
x4=detrend(Ka*V4f(2000:end))/wd^2;
t=t(2000:end);
figure
plot(t,x0,t,x1,t,x2,t,x3,t,x4)

% compute vibration amplitude
Vibm(1)=max(x0);
Vibm(2)=max(x1);
Vibm(3)=max(x2);
Vibm(4)=max(x3);
Vibm(5)=max(x4);

% tip loads [kg] (measured w/scale in machine shop)
m0=1.481;
mL=[0.289 0.291 0.538];
mtL=[m0 m0+mL(1) m0+mL(3) m0+mL(1)+mL(3) m0+mL(1)+mL(2)+mL(3)];

% plot results
figure
plot(mt(2:end),Vib(2:end)/Vib0*100,'r','linewidth',2)
hold on
plot(mtL,Vibm/Vib0*100,'bx','linewidth',5)
xlabel('Tip mass m_t [kg]')
ylabel('Vibration [%]')
title('Vibration sensitivity to tip (load) mass')
text(mtL(1)-0.1,Vibm(1)/Vib0*100-0.5,strcat('m_t_,_u_n_l_o_a_d_e_d ='...
   ,num2str(mt0),' [kg]'))
text(mtL(5)-0.1,Vibm(5)/Vib0*100-0.5,strcat('m_t_,_l_o_a_d_e_d ='...
   ,num2str(mtL(5)),' [kg]'))
legend('model','data',2)
```