Submillimeter Array Technical Memorandum

Number: 106

Date: Jan. 15, 1997

From: Jun-Hui Zhao

Subject: SMA Servo Model in MATLAB

---

1. Abstract

Using MATLAB, a commercial analysis software tool developed by MathWorks in Natick, MA, we have built a SMA servo model. This servo model is now running in CANISMINOR, a PC in room M226. One can use it to assist in the prediction of the SMA Servo system behavior under various conditions. In this memorandum, We shall give a brief introduction to this useful tool.

2. Introduction

2.1. Dynamics of the SMA Servo Model

The dynamics of the SMA servo can be described by a set of second order differential equations (given by Eric Keto 1996):

$$I_1\ddot{\theta}_1 + P_1\dot{\theta}_1 - P_2(\dot{\theta}_2 - \dot{\theta}_1) - k(\theta_2 - \theta_1) = T_1, \quad (1a)$$

$$I_2\ddot{\theta}_2 + P_2(\dot{\theta}_2 - \dot{\theta}_1) + k(\theta_2 - \theta_1) = T_2, \quad (1b)$$

where I_1 and I_2 is the inertias of the motor and dish, respectively; P_1 and P_2 are the damping parameters for both motor and structure; k is the spring constant; and T_1 and T_2 are the torques applied on the motor and dish, respectively. The gear ratio between the dish and motor has been coupled into the system. The resonant frequency for the dish/spring system is given by

$$\nu_2 = \frac{1}{2\pi}\sqrt{k/I_2}. \quad (2)$$

For the entire motor/spring/dish system, the resonant frequency is

$$\nu_{12} = \frac{1}{2\pi}\sqrt{k/I_1 + k/I_2}. \quad (3)$$

2.2. PID Control Loop

In this MATLAB servo model, we use Peter Chemiets's design for the proportion-integration-derivative (PID) control loop (Figure 1). Four control parameters are the velocity proportional gain (K_v), velocity integration gain (K_{vi}), position proportional gain (K_p) and position integration gain (K_{pi}).

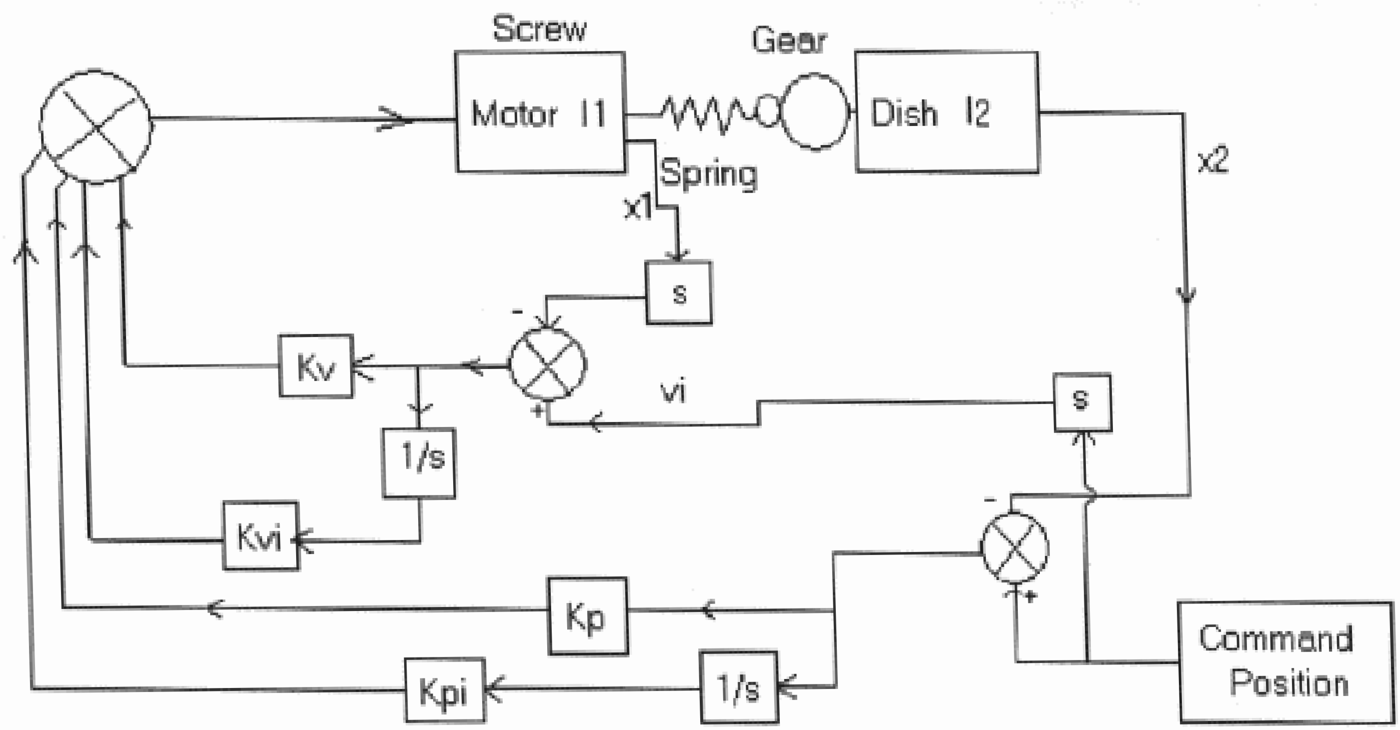

Figure 1: Sketch PID Control Architecture provided by P. Chemiets.

2.3. Friction and Stiction Model

A friction and stiction (F-S) model has been built using the SIMULINK, a toolbox within MATLAB. Four control parameters are used to adjust the F-S model. A range of velocity can be set by poa; within the velocity range (–poa, poa), the stiction torque is

$$T_s = -sign(\dot{\theta}) \cdot sticf \cdot poa \cdot Kfa - afa \cdot \dot{\theta}; \tag{4}$$

Outside this velocity range, the following friction torque is applied:

$$T_f = -sign(\dot{\theta}) \cdot poa \cdot Kfa. \tag{5}$$

Figure 2 illustrates this F-S model.

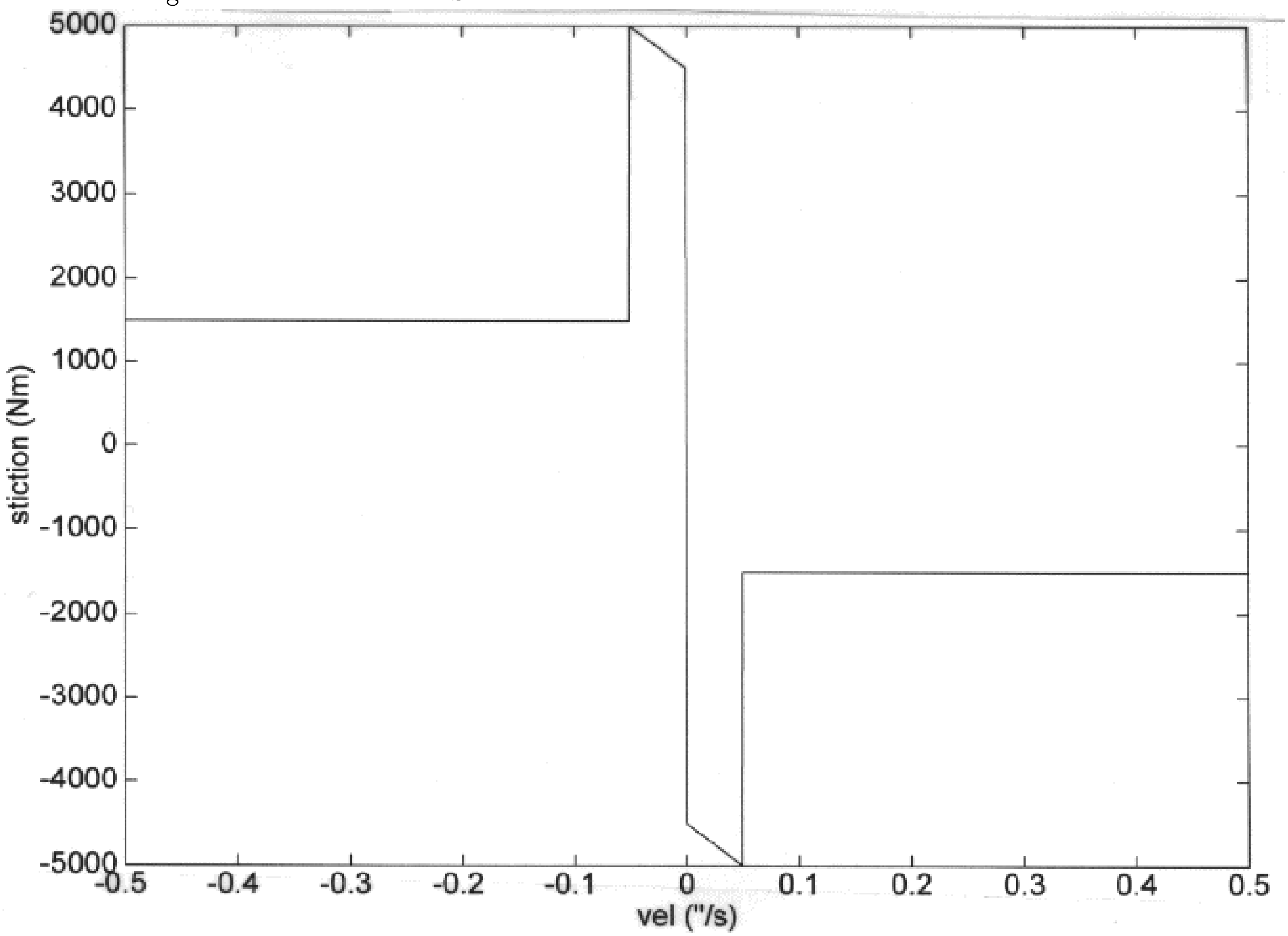

Figure 2: Illustration of F-S model.

2.4. Gust Wind Model

A Gust wind model is created using a band limited white noise generator in SIMULINK. A wind model is plotted in Figure 3 (upper panel). Assuming that the torque applied by the wind is proportional to the square of wind speed, the wind torque can be calculated (Figure 3, lower panel).

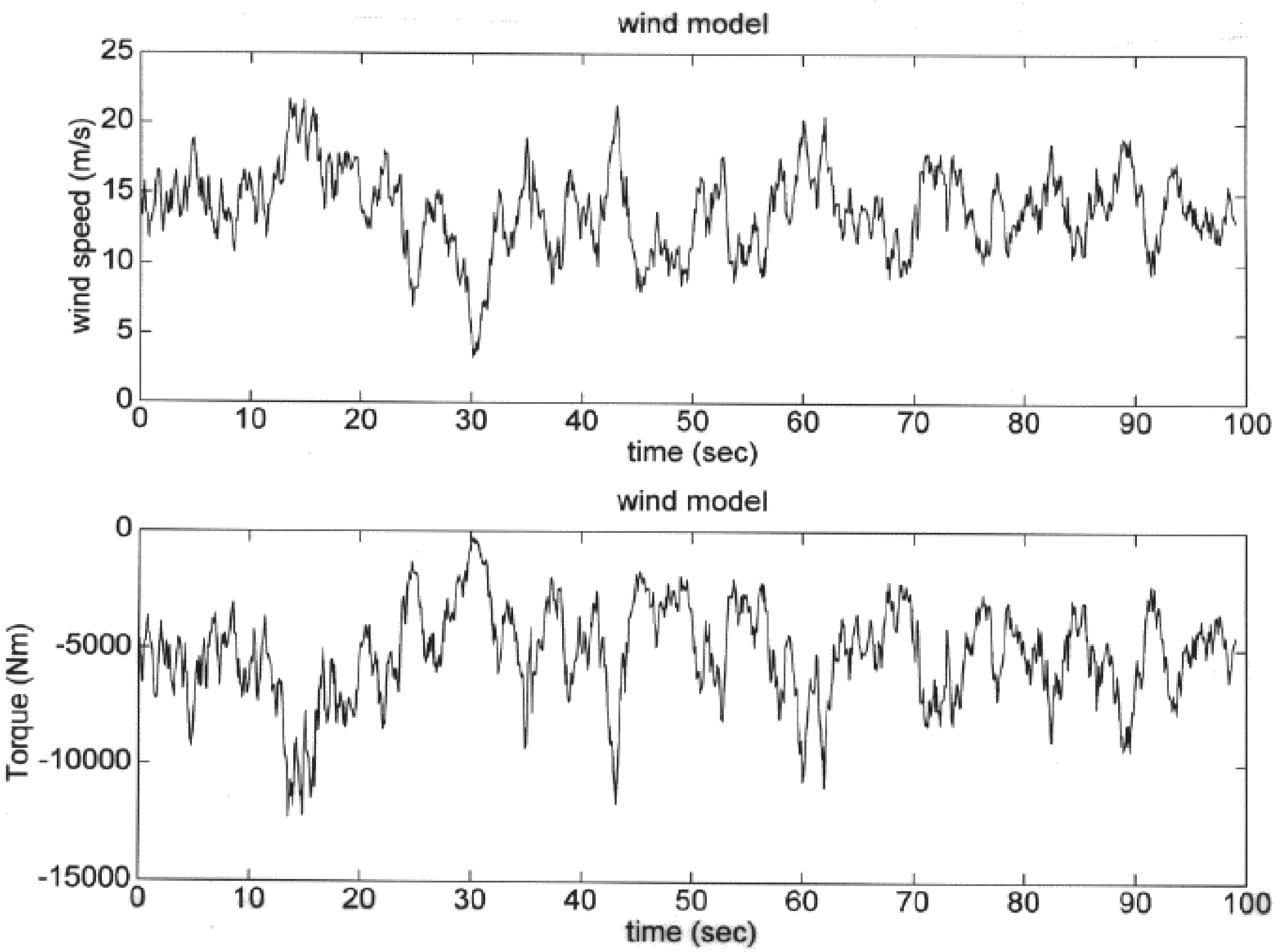

Figure 3: Illustration of gust wind model. Upper panel: wind speed vs. time; lower panel: wind torque vs. time.

2.5. Unit System

The unit systems used in astronomy and mechanical engineer are sometimes confused. For convenience for both astronomers and engineers, a special unit system is used in this MATLAB servo model. The units for the parameters used in this model are tabulated in Table 1:

Table 1: Parameters and Units

Parameter	Symbol	Unit
Inertia	I	kg m^2
Torque	T	Nm
Angular Position	θ	"
Angular Velocity	$\dot{\theta}$	"/s
Damping Constant	P	Nm/"/s
Spring Constant	k	Nm/"
Velocity Prop. Gain	K_v	Nm/"/s
Velocity Int. Gain	K_{vi}	Nm/"
Position Prop. Gain	K_p	Nm/"
Position Int. Gain	K_{pi}	Nm/"s

Note: " stands for arcsecond.

3. Running Servo Model

3.1 Starting MATLAB and Making Servo SETUP File

This section covers running servo model using MATLAB in the Windows environment. The servo model will be available in the local network as soon as we get the MATLAB and other Toolbox licenses for our alpha stations.

After login, users must ensure that Windows itself is running before MATLAB starts from within Microsoft Windows. MATLAB for Windows can be started by clicking on the MATLAB icon in the Windows Program Manager. When users invoke MATLAB, the Command Window is created and made the active window. The Command Window is the mechanism through which users communicate with the MATLAB interpreter. The interpreter displays its prompt (>>) indicating that it is ready to accept instructions from users. It is convenient for users to combine a number of instructions into an M-file. Users can type the M-file name under the prompt (>>) to execute a model or several models involved.

A setup file (RSERVO.M, see Appendix A) for the SMA servo model has been compiled including the following blocks:

1. Inertia: to give the inertia values for both the dish (Id) and motor (Im);
2. Stiction and Friction: to define the F-S model by setting the parameters (poa, Kfa, afa and sticf);
3. Gust Wind: to apply the wind torque;
4. PID Control: to tune the PID control parameters (Kv, Kvi, Kp and Kpi);
5. Telescope Tacking: to give the tracking speed and initial offset between dish and command position;
6. MATLAB Control Parameter: to set setup size for the simulation algorithms;
7. Initial Value: to set the initial values of the integrators involved;
8. Plotting: there are several sub-blocks to plot the F-S model, Gust Wind model, and SMA Servo results.

3.2. Simulation of SMA Servo

Using a setup file given in Appendix A, we simulate the SMA servo with Runge-Kutta (rk45) algorithm within MATLAB. The method rk45 performs well for mixed continuous and discrete time system.

Figure 4 shows the some testing results obtained from this SMA servo model in MATLAB. Discussion of the problems in the current SMA servo is given in a separate SMA memo (Eric Keto 1997).

3.3. Improvement

We are improving the servo model including digitalization.

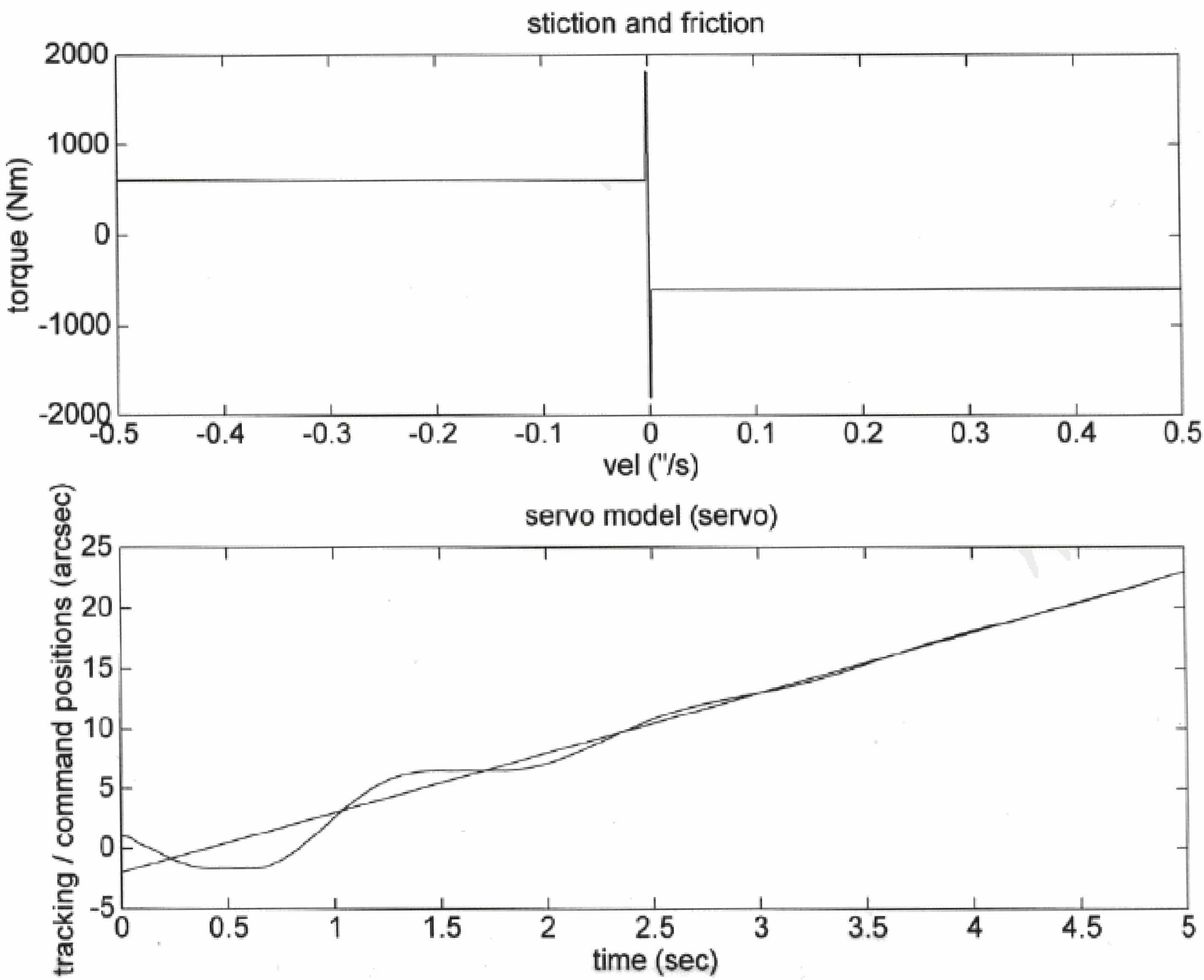

Figure 4a: Results from SMA servo simulation using MATLAB. Upper panel: F-S model used in the simulation; lower panel: one of the simulation outputs; the steady line represents the command position and the curve is the dish position.

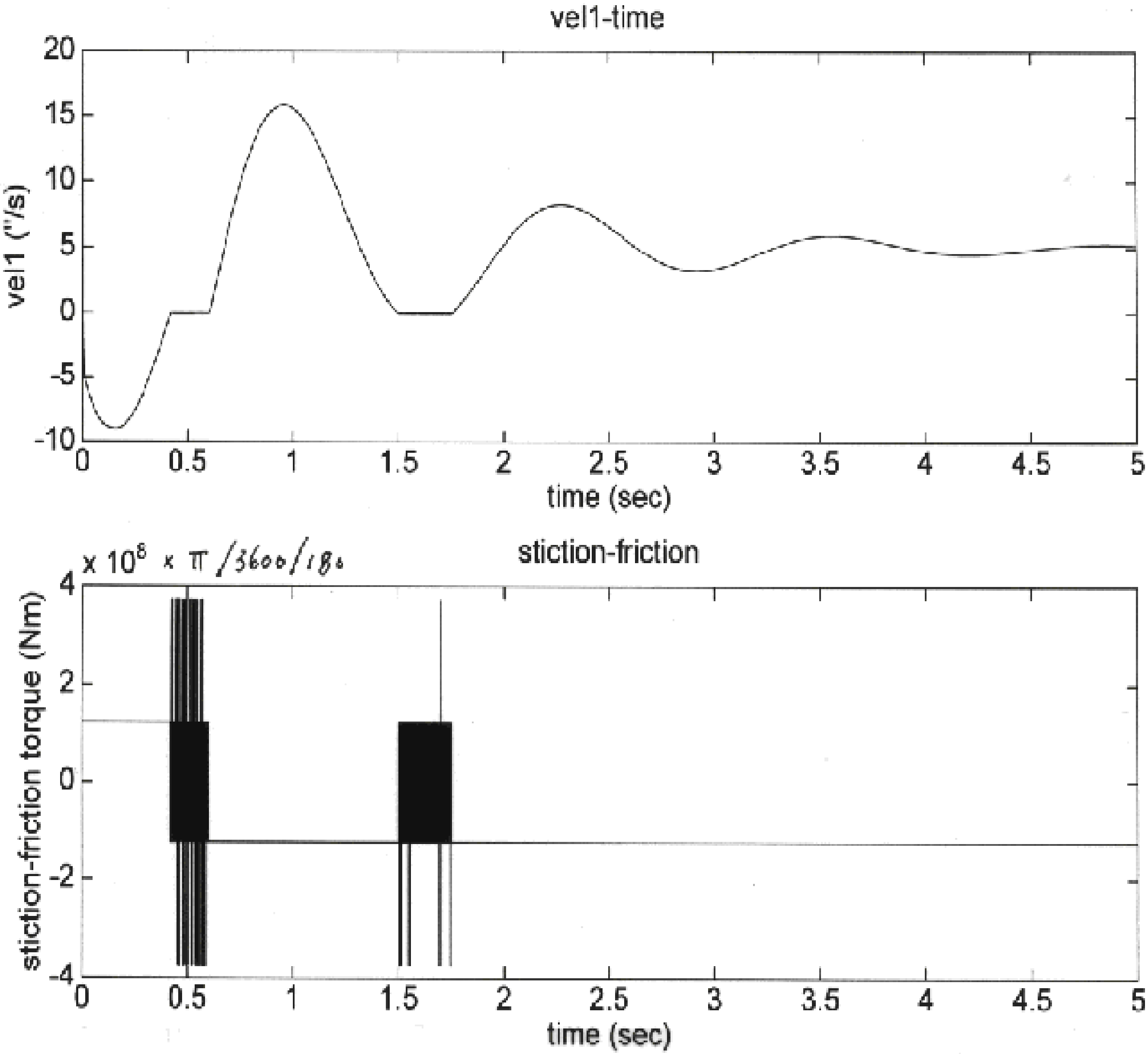

Figure 4b: Results from SMA servo simulation using MATLAB. Upper panel: the motor velocity vs time; lower panel: the simulated F-S behavior when the system starts running.

Appendix: A Setup file for SMA Servo in MATLAB

```
                                RSERVO.M
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dec 18, 1996,  Jun-Hui Zhao                              %
% This is a setup for running servop.m built in Matlab     %
% using Simulink with frinction and stiction in it         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% INERTIAs For Dish and Motor    %
% (kg m**2)                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Id=23200;
Im=80000;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      STICTION & FRICTION       %
%      Torques (Nm)              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%friction stiction on if fon>0;fon=0 for off
% friction=poa*Kfa    (Nm)
% stiction=sticf*poa*Kfa+afa*velocity (Nm)
% poa ("/s): the maximum velocity
%            to overcome the stiction torque;
% Kfa (Nm/"/s): a friction coeficient;
% afa (Nm/"/s): a maximum stiction coeficient;
% sticf:  a ratio of sticiton to friction.
fon=3600*180/3.1415926*1;
%fon=0;
poa=2e-3;
Kfa=3e5;
afa=1e4;
sticf=3;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            WIND               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%wind torque on if wind=1;wind=0 for off
wind=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      PID CONTROL                         %
%  Parameters:                             %
%  Kv (Nm/"/s) velocity proportional gain; %
%  Kvi(Nm/")   velocity integrated gain;   %
%  Kp (Nm/")   position proportional Gain; %
%  Kpi(Nm/"s)  position integrated gain;   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%pid control loop on if pid=1; 0 is off
pid=3600*180/3.1415926;
```

Page 1

```
%pid=0;
%
%PID loop parameters:
%The following pid parameters are good for stable servop
%Kv=-2e5;
%Kvi=5e4;
%Kp=1e7;
%Kpi=8e5;
%1/G=1
%testing
Kp=200;
Kpi=2424;
Kvi=0;
Kv=-10e1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       TRACKING SPEED ("/s)  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tspeed=5;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%position offset=tspeed*offset (")  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
offset=2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   DAMPING Constant  p2 (Nm/"/s)   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%dish damp on if damp2on=1; 0 is off
damp2on=1;
p2=2.8e5;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SPRING CONSTANT  K2 (Nm/")  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
k2=6.3e7;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MATLAB CONTROL Parameters   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tol=2e-5;
minstep=2e-5;
maxstep=0.001;
length=5e5;
options=[tol,minstep,maxstep];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   PLOT THE STICTION & FRICTION %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x0=0;
tend=1;
```

```
[tt,x]=linsim('stictst',tend,[],options);
subplot(2,1,1);
plot(vel,stic);
title('stiction and friction');
xlabel('vel ("/s)');
ylabel('torque (Nm)');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Plot the wind-gust a as function of     %
%   time by remove % sign in front of next %
%   line.                                   %
%[tw,x]=linsim('windtst',99,[],[1e-3,1e-5,.5]);
%*********************************%
%   Set up the initial value        %
%   x1: intial motor position;      %
%   x2: intial dish position;       %
%   v1: intial motor velocity;      %
%   v2: intial dish velocity;       %
%   pi1: intial value for the       %
%        position gain integrator;%
%   vi1: intial value for the       %
%        velocity gain integrator.%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x1=1;
x2=1;
v1=0;
v2=0;
pi1=0;
vi1=0;
[t,x]=rk45('servo',5,[],options);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Plot the servo results       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(2,1,2);
plot(t,servo2);
title('servo model (servo)');
xlabel('time (sec)');
ylabel('tracking / command positions (arcsec)');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   plot stiction vs time
%        vel      vs time
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plotsti
% end
```