A Brief Analysis of Guitar String Harmonics, Partials, and Timbre
Using Matlab FEA, FFT & ME345

Aleksey Ryzhakov
4/16/10

For this brief analysis, a Fall09 ME370 Guitar String project was used. The premise of the project was to use rudimentary Finite Element Analysis to simulate the response of a guitar string after it has been plucked. The string response was then scanned at a specific point on the string—presumably, the location would approximate the location of an electric guitar pickup that scans the string and produces sound. With ME345 skills, the string vibration can be more closely observed.

The Matlab code is attached at the end of this paper. Note: some of the code was provided by Dr. Eric Marsh. His portion of the code has been identified in the comments. Change the number of elements, n, for faster code processing (at the expense of accuracy).

Results

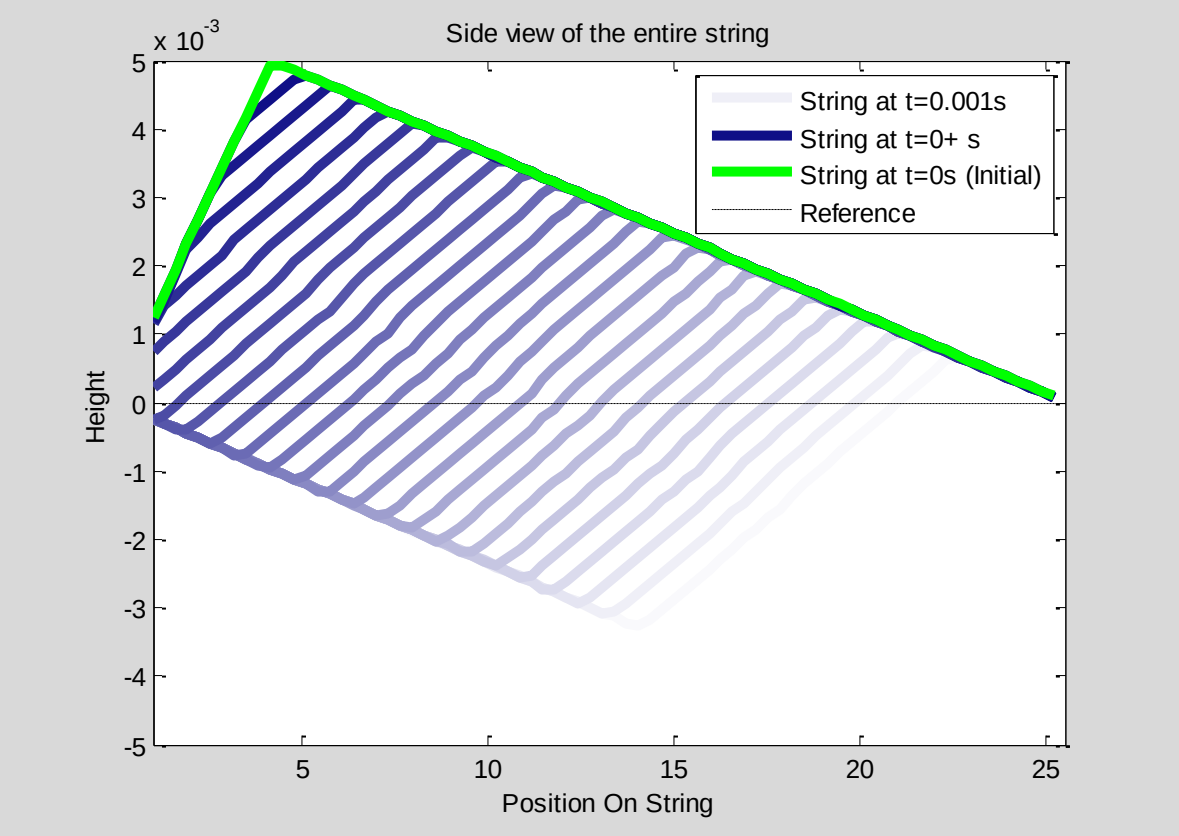

Fig 1.0

Figure 1.0: A visual representation of how the string changes with time. At t=0, it is being plucked, at t=0+ it is starting to oscillate.

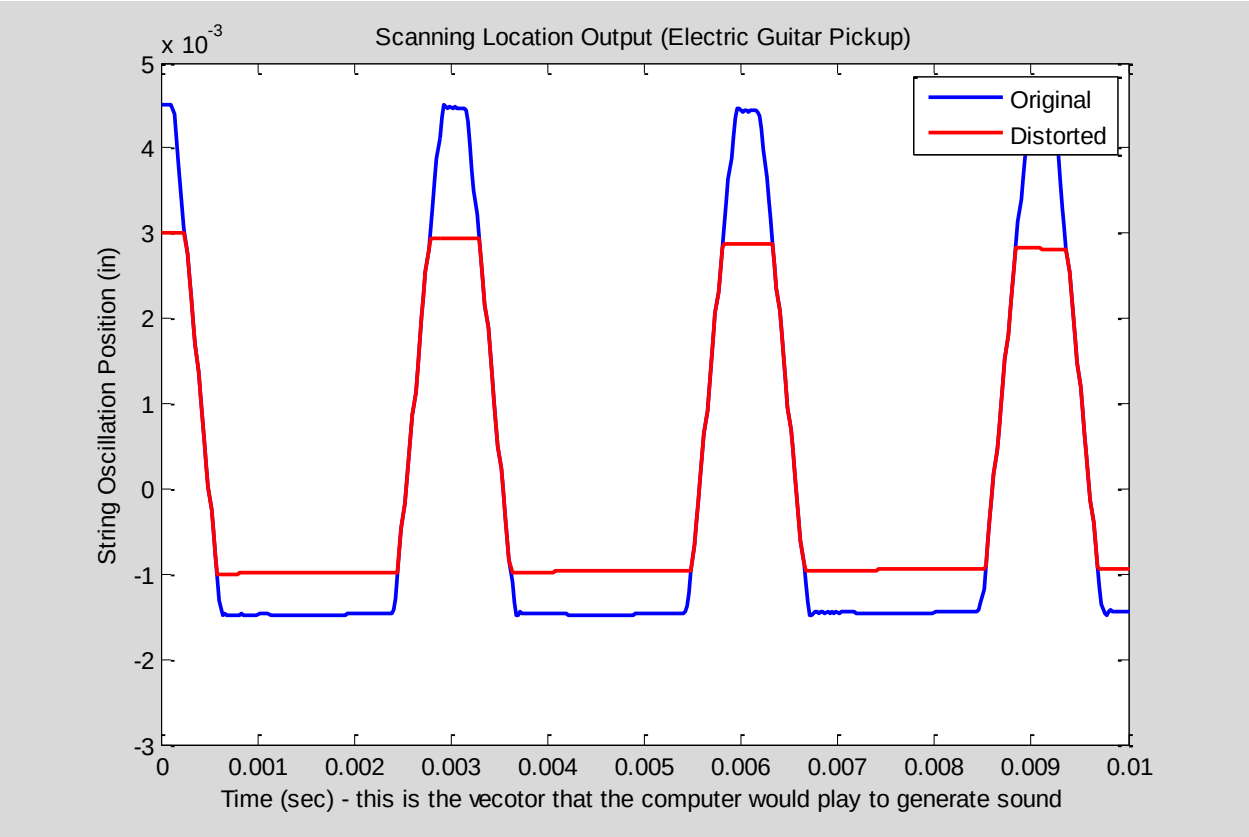

Fig 1.1

Figure 1.1: String output as measured by observing the deflection of the string at 6.6 in from the left. This is the focus of the FFT analysis.

Blue – Natural string output
Red – Simple guitar distortion effect added (exponentially decaying clipping, decays with the string oscillation amplitude decay). This produces a very rough approximation of how a distorted guitar sounds.

It is critical to note that while an electric guitar pickup observes only one portion of the string, an acoustic guitar vibrates the air with its entire length, and it's time plot would thus be different. More importantly, an acoustic guitar also contains a resonating chamber that further affects the time plot.

FFT Analysis: is the simulation correct?

The string FEA initial conditions are made specifically such that the output note is E4 – 329.6 Hz
Tension force T = 72.5239 lbs
Length of the String L = 0.6477 in
Linear Density rho = 0.0004 lbs/in^2

Using the general equation for frequency of a string

F_theoretical = (1/2L)sqrt(T/rho) = 329.6 Hz = E4

This confirms that the string being simulated should be playing the E4 note. We expect the Finite Element Analysis to return a string that oscillates at 329.6Hz, and an FFT output to contain the biggest response at around that frequency.

FFT analysis was conducted on the string scanning location output vector (fig 1.1 – note, the graph has axis limits, the actual response is measured for one second)

Sample Frequency was 44100 Hz, and the sampling time was 1 s.

The FFT output is located on the next page (figure 2.0 and figure 2.1).

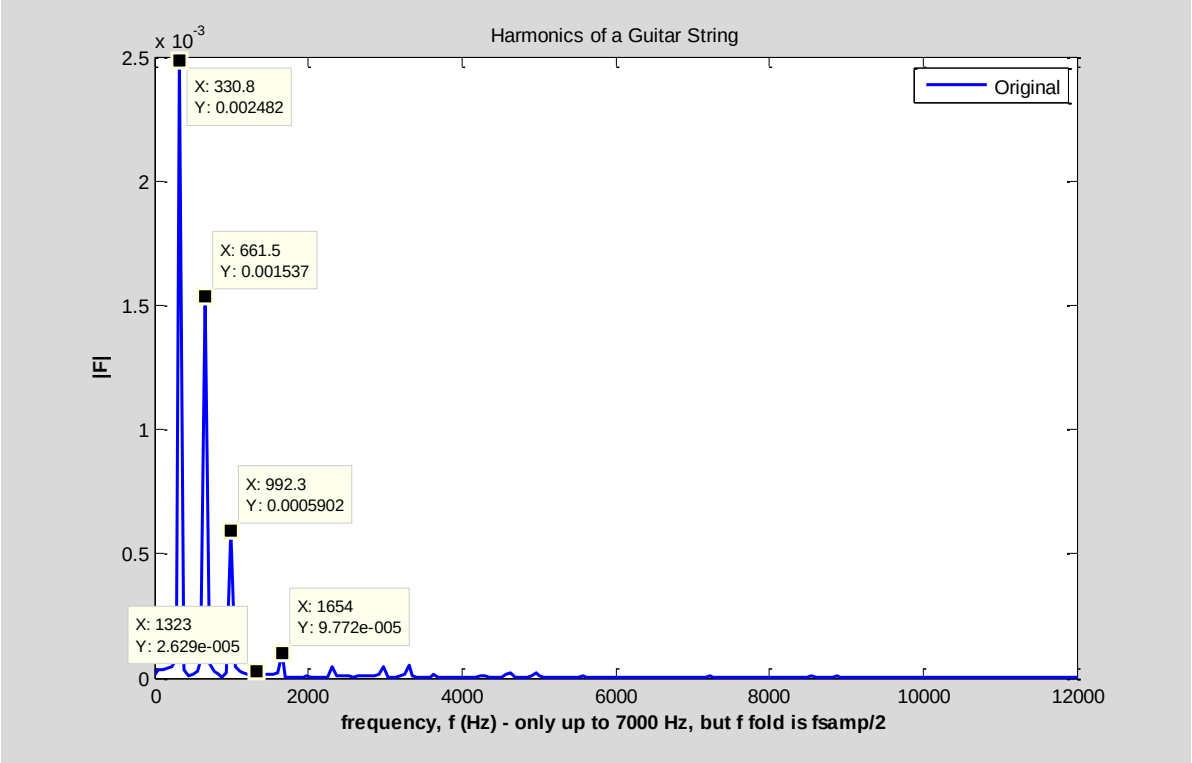

Figure 2.0

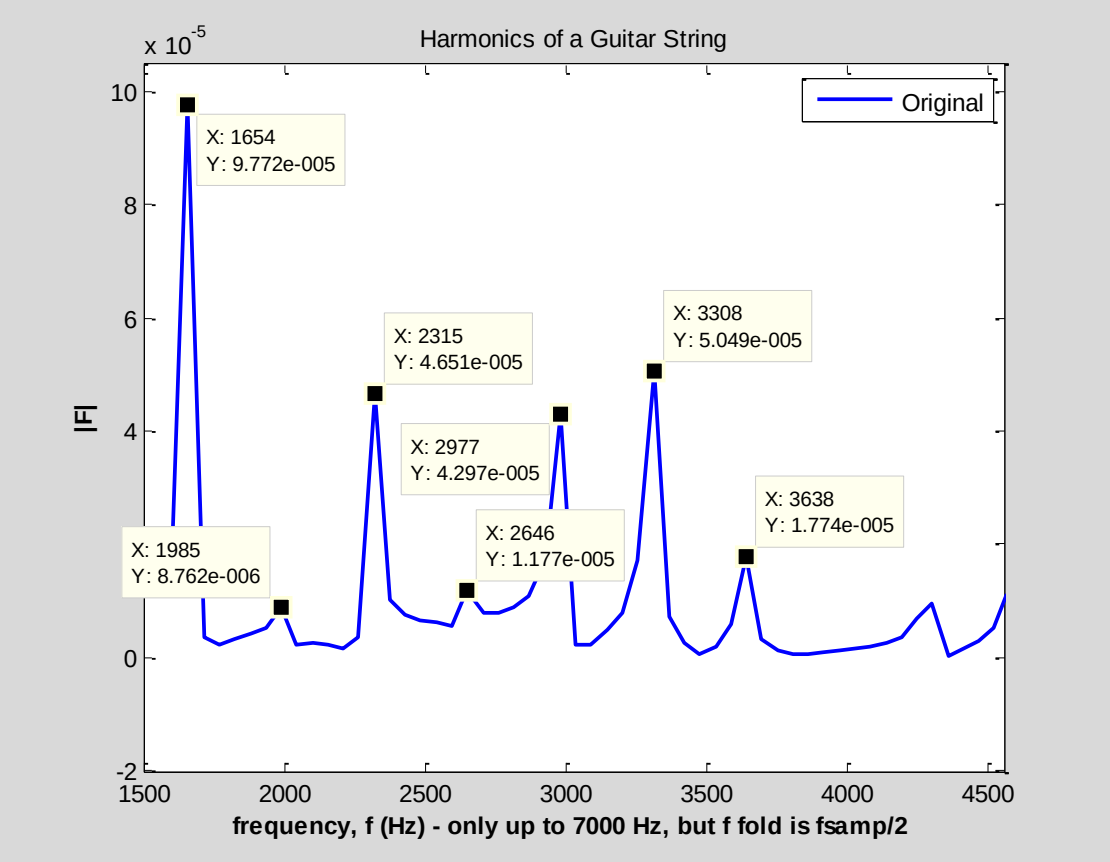

Figure 2.1 – zooming in on higher harmonics

The largest frequency is 330.8 Hz, which is in line with our expectation of 329.6 Hz. The simulation is out of tune only by 1.2 hz. According to the calculator found on the website http://www.sengpielaudio.com/calculator-centsratio.htm, this corresponds to being out of tune by about 6 cents – a small amount.

The main concern, however, is how accurate is this simulation at higher harmonics?

Expected harmonic frequency = Harmonic # * fundamental frequency

Harmonic	Cent Offset	Expected Frequency (Hz)	FFT Output (Hz)	% Error
2	1200	659.2	661.5	0.348908
3	1902	988.8	992.3	0.353964
4	**2400**	**1318.4**	**1323**	**0.348908**
5	2786	1648	1654	0.364078
6	3368	1977.6	1985	0.374191
7	3600	2307.2	2315	0.338072
8	3803.9	2636.8	2646	0.348908
9	3986.3	2966.4	2977	0.357335
10	4151.3	3296	3308	0.364078
11	4302	3625.6	3638	0.342012

Figure 3.0

The table concludes that the model is very accurate. Curiously, the 3rd harmonic had an extremely low amplitude (but it is there)—that may have been due to the limited leakage.

As seen from our FFT, the simulated guitar string had all the harmonics (multipliers of fundamental frequency), but no partials (peaks in between the harmonics). This may be due to the fact that our simulation is perfect – in other words, the original output has no electronic post processing or, in the case of an acoustic guitar, vibration effects that would be found between the acoustic guitar and its wooden elements.

Let's analyze the FFT with a simple post-processing effect: simple distortion (red line in Fig 1.1). According to Wikipedia, the definition of an instrument timbre is its unique harmonic amplitude spectrum, and possibly the addition of partials. The timbre is what makes a guitar sound different than, for instance, a piano. This will test that definition.

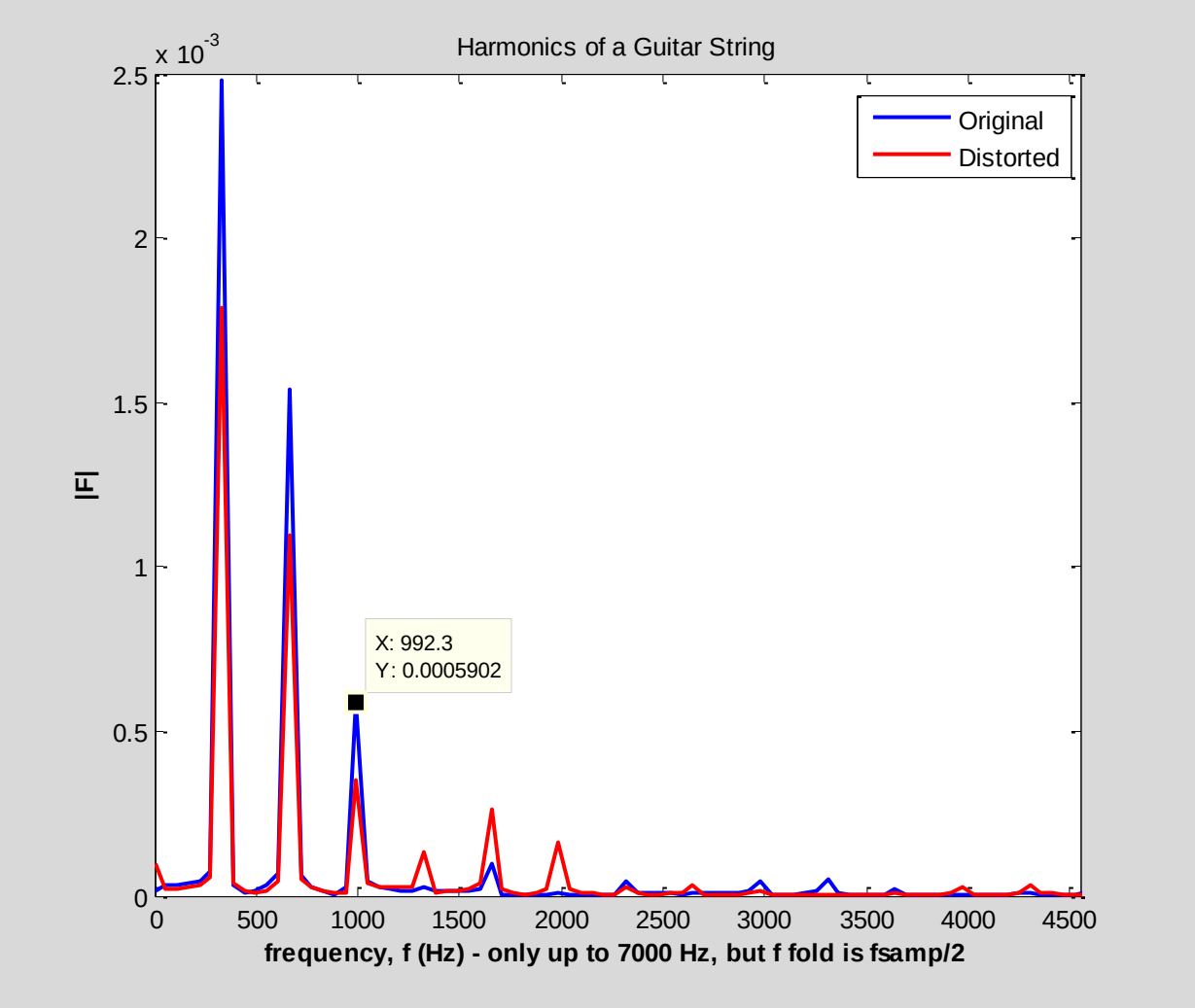

Figure 4.0

The distortion effect dampens the first three harmonics, but drastically amplifies harmonics 4, 5, 6. No partials are added. If played through MatLab, the distortion can be clearly heard, and thus the FFT plot confirms the definition of timbre—even with this same harmonic layout, the change in relative amplitudes affects the sound that is produced by the guitar.

```
%% Guitar String Analysis
% Guitar String Appearance and Output

clc
clear all

% Number of elements in the FEA analysis of the string
n = 400;

% Define time vector (lasts Tmax seconds)
fsamp = 44100;
Tmax = 1;
t = (0: 1/fsamp: Tmax);


% Define string properties to be used in the FEA analysis.

T = 16.2975 * 4.45;                 % tension
L = 25.5 * 0.0254;                  % length of standard 25.5" ax
rho = 7850*pi*(0.010*0.0254)^2/4;   % density per length of 0.010" string

% To prove the FFT correct, use the approximation formula to calculate the
% theoretical frequency output.  Based on the above properties, if
% unchanged, f_theory will be 329.6 Hz, or E4
f_theory = sqrt(T/rho)*(1/(2*L));
disp(f_theory)


% ***** Finite Element Analysis of the Guitar String!*****
% ***** Program for Initial values created by Dr. Marsh***
%*********************************************************
% Element mass and stiffness matrices
k = T*n/L*[1 -1; -1 1];
m = rho*L/n*[3 0; 0 3]/6;

% Assemble global mass and stiffness matrices
K = zeros(n+1);
M = zeros(n+1);
for inc = 1: n,
    K(inc:inc+1,inc:inc+1) = K(inc:inc+1,inc:inc+1) + k;
    M(inc:inc+1,inc:inc+1) = M(inc:inc+1,inc:inc+1) + m;
end
clear T rho k m inc


% Rayleigh damping model (alpha = 5, beta = 0.00000002)
% String Damping
C = 5*M + 2e-8*K;

% Apply boundary conditions (ME 461 material)
K(1,1) = 50*K(1,1);
K(end,end) = 50*K(end,end);

% Solve eigenvalue problem
```

```
[X, omega_squared] = eig(K, M);
omega_n = sqrt(diag(omega_squared));

% Reorder from lowest to highest
[omega_n, i] = sort(omega_n);
X = X(:,i);

% Decouple matrices
mi = diag(X'*M*X);
ci = diag(X'*C*X);
ki = diag(X'*K*X);

clear M C K

% Underdamped system properties
zeta = ci./sqrt(ki.*mi)/2;
omega_d = omega_n .* sqrt(1 - zeta.^2);

%%
%***END OF Dr. Marsh material;**
%******************************

% 4. Initial conditions from spatial to modal coords
x0 = zeros(length(X), 1);
v0 = zeros(length(X), 1);
m = .005; % pluck amount


f = 6;
input = L/f; %Location from the left where the string will be plucked (1/6 is
%a good approximation for where a guitar is plucked)

loc = round(input/(L/n)); %identifies the mass element that is being plucked
z=0; %distance vector

% Populate the initial condition matricies with the initial deflection
for inc = 1:loc
     z = z+L/((loc)*f);
     x0(inc, 1) = m*f/L*z;
end
intval = x0(loc, 1);
clear inc
for inc = (loc+1):(n)
     z = z+L/((loc)*f);
     x0(inc, 1) =intval -m*f/(L*f-L)*(z-L/f);
end

eta0  = X\x0;
etad0 = X\v0;

scan_loc = round(n/(f-2)); %the location where you wish to scan the


rest_pos = [ 0 0]; %ignore - this is to plot the string rest position
```

```
values = [0 25.5]; %


% Conduct a differential equations approach to find a solution to the
% matrix of simultanous equations

W = eta0;    %Constants, Solution
V = (etad0 + zeta.*omega_n.*W)/omega_d;    %Constants, Solution

 %Homogenious Solution

 eta_h = zeros((length(X)),(length(t)));
 for inc = 1: length(W),
    eta_h(inc,:) = W(inc)*(exp(-
zeta(inc)*omega_n(inc)*t).*cos(omega_d(inc)*t)) + ...
    V(inc)*(exp(-zeta(inc)*omega_n(inc)*t).*sin(omega_d(inc)*t));
end

x = X*(eta_h);
clear eta X M C K mi ci ki

% Plot the response
figure(1)
for z = 1:2:49
    inc=50-z;
    color=[4*inc/200 4*inc/200 (0.5+2*inc/200)];
    if inc==1
        plot(25.5*[0:5:(n-1)]/n, x(1:5:n, 1),'g','LineWidth',4)
    else
    plot(25.5*[0:5:(n-1)]/n, x(1:5:n, inc),'Color',color,'LineWidth',4)
    end
    axis([1 25.5 -.005 .005])
    hold on
end
plot(values, rest_pos, 'k--')

xlabel('Position On String')
ylabel('Height')
title('Side view of the entire string')
clear x0 v0
hold off

figure(2)
plot(t,x(scan_loc,:),'b','LineWidth',2)
hold on
axis([0 .01 -.003 .005])
xlabel('Time (sec) - this is the vecotor that the computer would play to
generate sound')
ylabel('String Oscillation Position (in)')
title('Scanning Location Output (Electric Guitar Pickup)')

%% ADAPTED FFT ANALYSIS CODE
% Adapted from S10 ME345 course

N = 800;
```

```
f_s = fsamp; % 44100 Hz

f = x(scan_loc,1:length(t)); % signal data

%Calculated values:

T = N/f_s; % Total sample time (s)
del_t = 1/f_s;
del_f = 1/T;
f_fold = f_s/2; % Folding frequency = max frequency of FFT
N_freq = N/2; % Number of discrete frequencies

% FFT of the time signal

for k = 0:N/2
    frequency(k+1) = k*del_f;
end

%NFFT = 2^nextpow2(N); % Use power of 2 for FFT (NOT necessary in Matlab, but
faster)
%F = fft(f,NFFT);    % Compute FFT for case with integer multiple of 2 data
points
F = fft(f,N);        % Compute FFT for general case: N not necessarily a
multiple of 2

for k =0:N/2
    Magnitude(k+1) = abs(F(k+1))/(N/2);
end
Magnitude(1) = Magnitude(1)/2; % Divide first term by a factor of 2
% Plot the frequency spectrum

figure(5)
% plot(frequency,Magnitude,'-
bo','MarkerFaceColor','r','MarkerEdgeColor','r','LineWidth',2)
plot(frequency,Magnitude,'LineWidth',2)
% title('FFT Frequency Spectrum','FontWeight','Bold','FontSize',16)
xlabel('frequency, f (Hz) - only up to 7000 Hz, but f fold is
fsamp/2','FontWeight','Bold')
ylabel('|F|','FontWeight','Bold')
title('Harmonics of a Guitar String')
xmin = 0;
xmax = 12000;
xlim([xmin xmax])
grid
hold on;
%% Distort

    dist = .003*exp(-zeta(10)*omega_n(10)*t).*(ones(1, length(x)));
    dist2 = -.001*exp(-zeta(10)*omega_n(10)*t).*(ones(1, length(x)));
    for con = 1:length(x)
        if x(scan_loc,con) > dist(1,con)
        x(scan_loc,con) = dist(1,con) ;

        elseif x(scan_loc,con) < dist2(1,con)
            x(scan_loc,con) = dist2(1,con);
```

```
        else
           x(scan_loc,con) = x(scan_loc,con);
        end
    end

N = 800;
f_s = fsamp; % 44100 Hz

f = x(scan_loc,1:length(t)); % signal data

%Calculated values:

T = N/f_s; % Total sample time (s)
del_t = 1/f_s;
del_f = 1/T;
f_fold = f_s/2; % Folding frequency = max frequency of FFT
N_freq = N/2; % Number of discrete frequencies

% FFT of the time signal

for k = 0:N/2
    frequency2(k+1) = k*del_f;
end

%NFFT = 2^nextpow2(N); % Use power of 2 for FFT (NOT necessary in Matlab, but
faster)
%F = fft(f,NFFT);    % Compute FFT for case with integer multiple of 2 data
points
F = fft(f,N);        % Compute FFT for general case: N not necessarily a
multiple of 2

for k =0:N/2
    Magnitude2(k+1) = abs(F(k+1))/(N/2);
end
Magnitude2(1) = Magnitude2(1)/2; % Divide first term by a factor of 2
% Plot the frequency spectrum

figure(5)
% plot(frequency,Magnitude,'-
bo','MarkerFaceColor','r','MarkerEdgeColor','r','LineWidth',2)
plot(frequency2,Magnitude2,'r','LineWidth',2)
% title('FFT Frequency Spectrum','FontWeight','Bold','FontSize',16)
xlabel('frequency, f (Hz) - only up to 7000 Hz, but f fold is
fsamp/2','FontWeight','Bold')
ylabel('|F|','FontWeight','Bold')
title('Harmonics of a Guitar String')
xmin = 0;
xmax = 12000;
xlim([xmin xmax])
grid
hold off;
legend('Original', 'Distorted')

figure(2)
plot(t,x(scan_loc,:),'r','LineWidth',2)
```

```
legend('Original','Distorted')
hold off
```