LAB 1 - Solutions

ECE 238
Spring, 2010

1)
2)
3) } → See MATLAB files (available on website.).

4) There are $2^{12} = 4096$ counts per revolution (2π rad), and rotating the encoder gear 1 turn on the gear "rack" corresponds to a linear distance of 0.093 meters. So your answer should be

a) $2\pi / (2^{12}) =$ **0.001534 (radians/ct)**

b) $0.093(m)/(2^{12}) =$ **2.27×10^{-5} (m/ct)**

(Either answer is OK...)

5) (See "figure 1".) To solve this, I measured where two "peaks" in the waveform occur.

$P_1 \approx (4.0, 0.73)$ and $P_2 \approx (9.0, 0.67)$

- There are exactly 2 periods between these points, so:

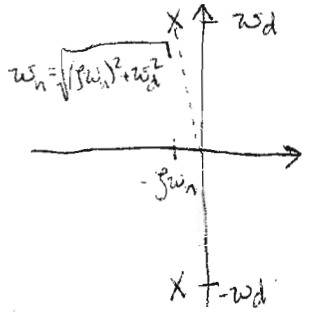
$$\omega_d \approx \frac{2\pi}{(9-4)} \approx 2.513\left(\frac{rad}{s}\right)$$

- The decay envelope will obey $x = Ae^{-\zeta\omega_n t}$, so

$$\frac{x_2}{x_1} = e^{-\zeta\omega_n(9-4)} = \frac{.67}{.73} = .92$$, taking log on each side:

$$-\zeta\omega_n \cdot 5 = \ln(.92) = -.083$$

$$\zeta\omega_n = 0.0167\ (rad/s)$$



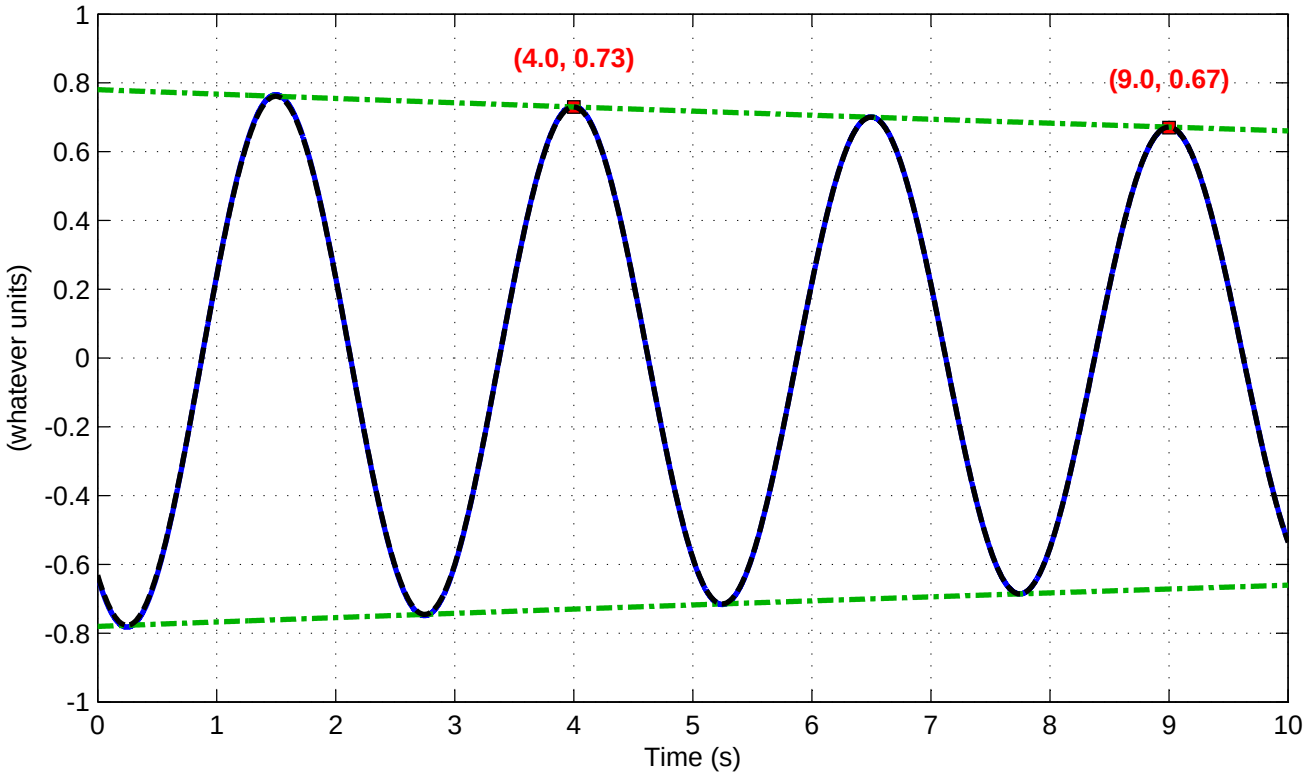
- From geometry at left, $\omega_n = \sqrt{(\zeta\omega_n)^2 + \omega_d^2}$

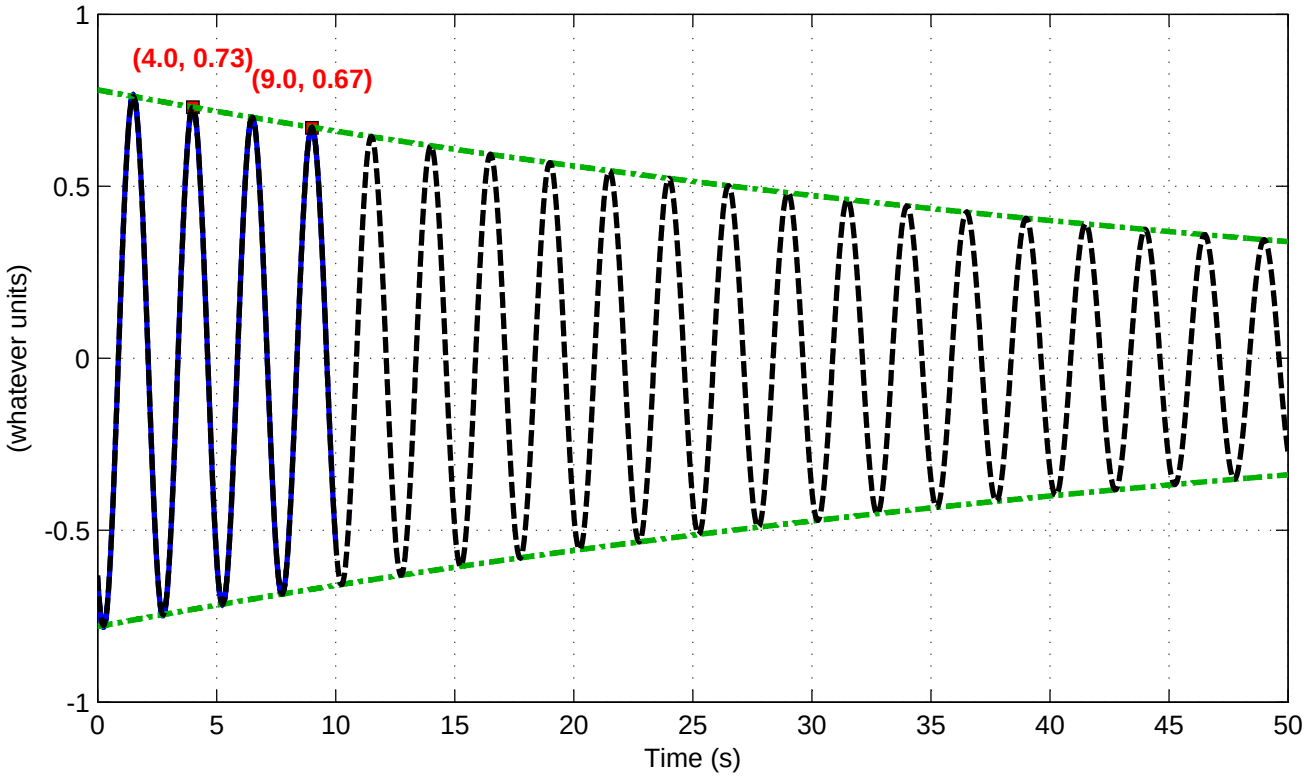
$$\omega_n \approx \sqrt{(2.51)^2 + (.0167)^2} = 2.51\left(\frac{rad}{s}\right)$$

$$\zeta = \frac{\zeta\omega_n}{\omega_n} \approx 0.007 \qquad s = -\zeta\omega_n \pm \omega_d j$$

$$s_1 = -.0167 + 2.51\left(\frac{rad}{s}\right) \qquad s_2 = -.0167 - 2.51\left(\frac{rad}{s}\right)$$

Figure 1 - Problems 5-7

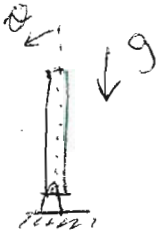




6) From 5), we found the characteristic eqn for the stable pendulum to be:

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0$$

$$s^2 + 0.033s + 6.32 = 0$$



The equation of motion for the upright pendulum should be just the same as the downward-pointing pendulum, except for the direction of gravity:

$$J\ddot{\theta} + B\dot{\theta} - gm\frac{L}{2}\sin\theta = 0$$

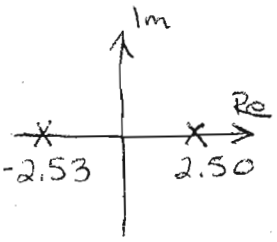
Linearizing:

$$J\ddot{\theta} + B\dot{\theta} - gm\frac{L}{2}\theta = 0$$

So, if we divide through by "J", only the last term in the characteristic equation should change:

$$s^2 + 0.033s - 6.32 = 0$$

$$s_1 \approx -2.53 \qquad s_2 \approx +2.50$$



7) (Extra Credit...)

For a solid rod, $J_{end} = \frac{1}{3}mL^2$.

$$\omega_n^2 = \frac{gm\frac{L}{2}}{J_{end}} = \frac{gm\frac{L}{2}}{\frac{1}{3}mL^2} = \frac{3}{2}\cdot\frac{g}{L}$$

Let's assume $g \approx 9.8\left(\frac{m}{s^2}\right)$

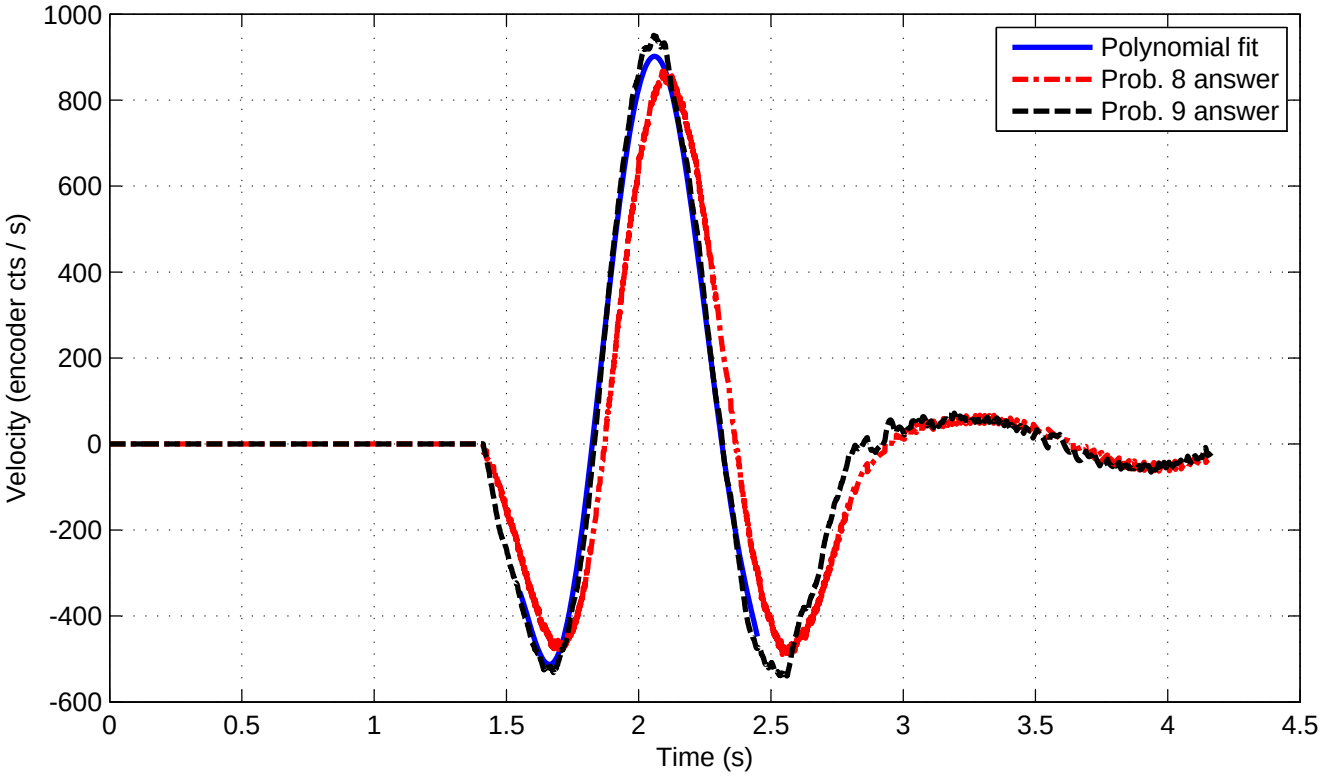
$$\therefore \quad L \approx \frac{3}{2}\cdot\frac{g}{\omega_n^2} = \frac{3}{2}\cdot\frac{9.8\left(\frac{m}{s^2}\right)}{6.3\left(\frac{rad}{s}\right)^2} = 2.3 \text{ meters}$$

8 & 9) Some flavor of "derivative & a low-pass filter" will probably be the most common solution.

e.g. $F(s) = s\cdot\left(\frac{1}{\tau s + 1}\right)$ → converted to z-domain.

(See "figure 2" & MATLAB code...)

Figure 2 - Problems 3 and 8 (velocity estimates)



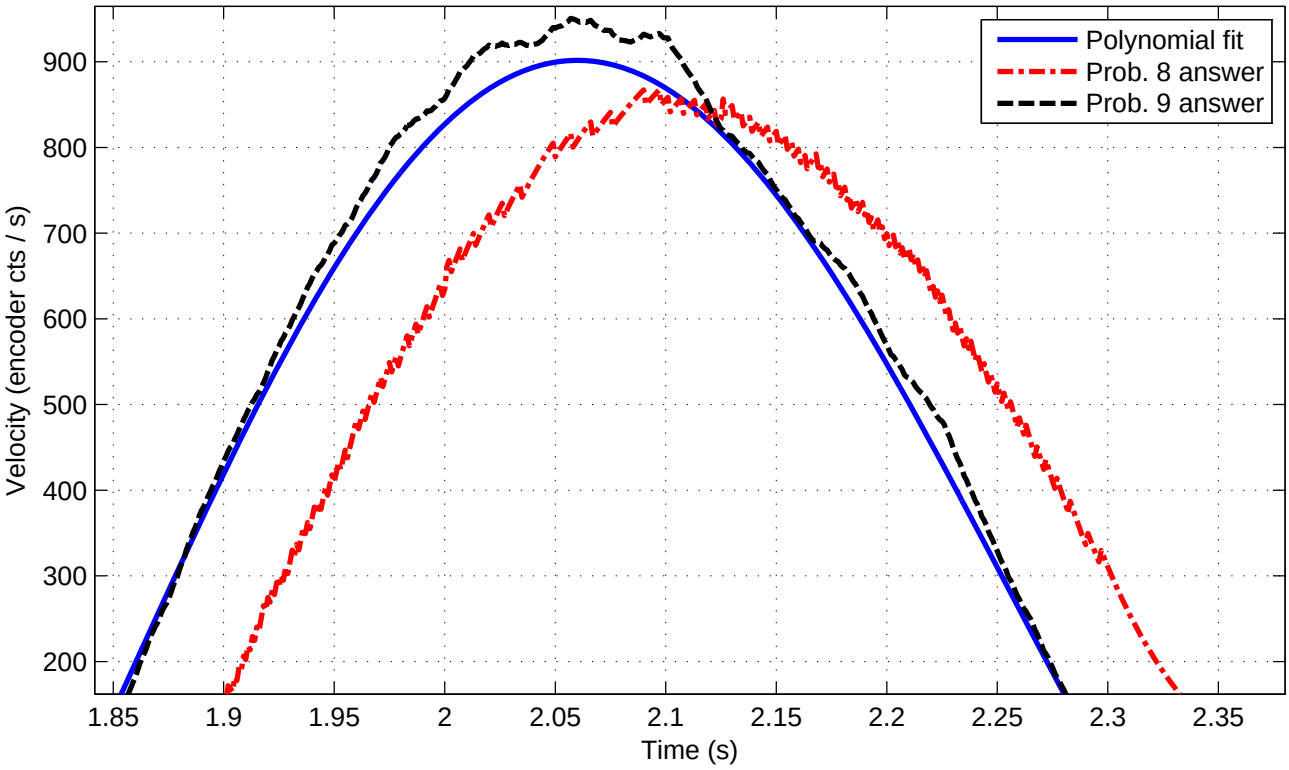
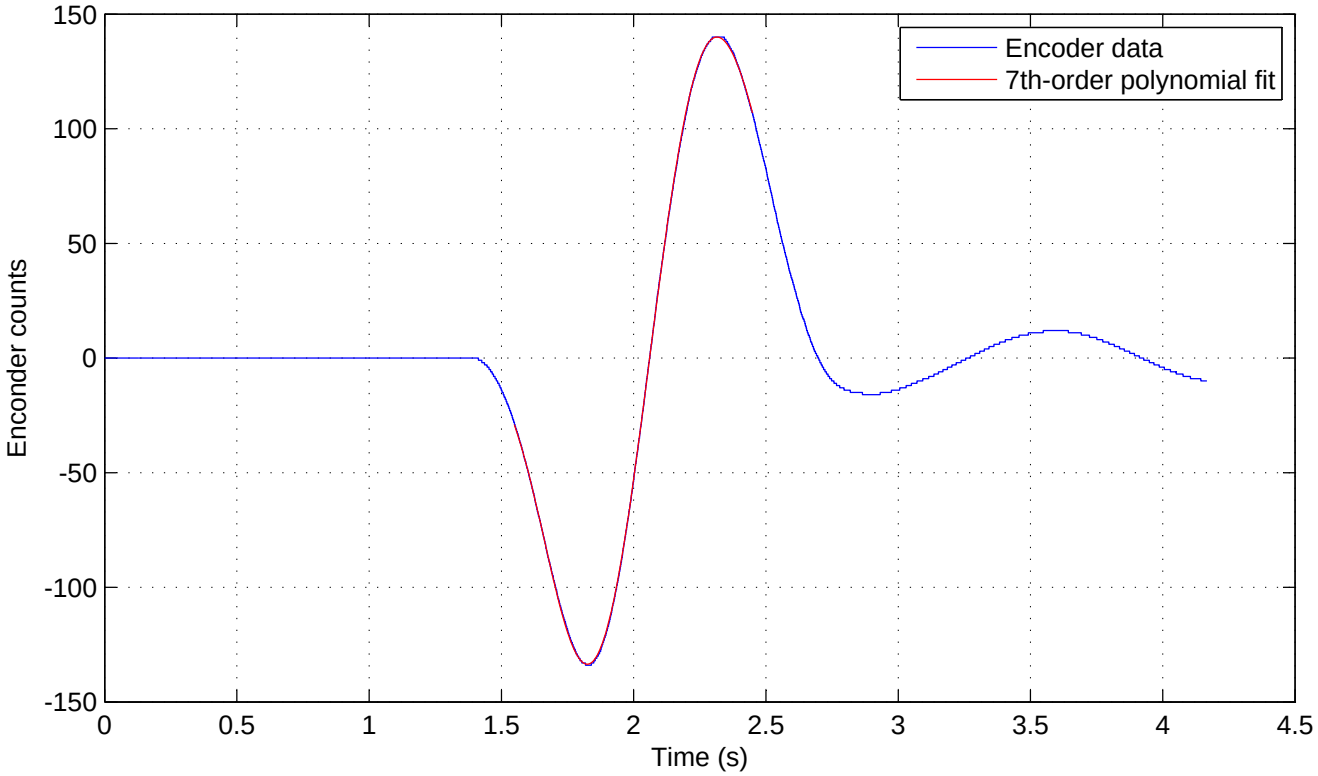
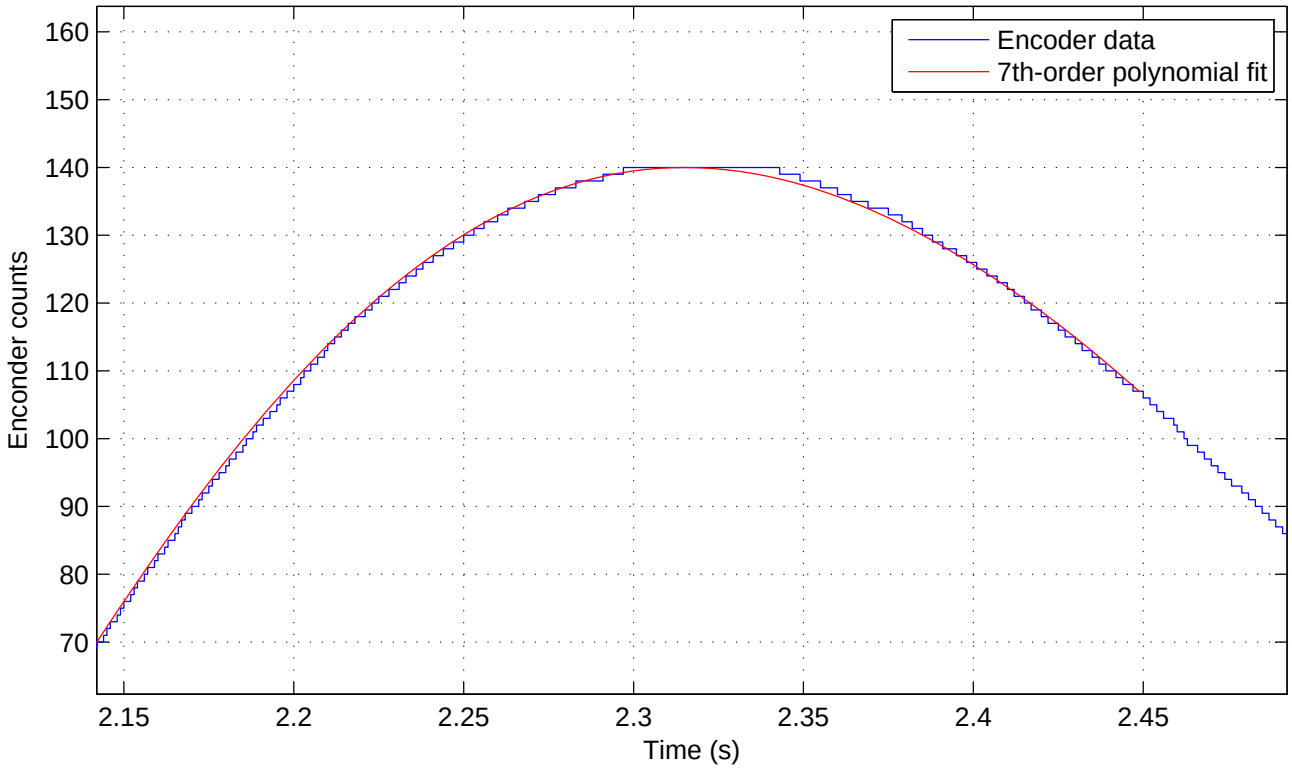


Figure 3 - Polynomial fit to raw encoder data





```
% lab1_mfile.m  - Partial solution to Lab 1
% ECE 238, Spring 2010 (UCSB)
% (Solution code below by Katie Byl, 4/11/2001)
%-----------------------------------------------

load lab1_matfile tdata ydata % raw data for encoder
T = tdata(2)-tdata(1); % sampling time, T=.001 (seconds)

% ydata : raw encoder data, "wiggling" pendulum
% tdata : corresponding time vector

%------- Problem 2) POLYNOMIAL FIT:
% select indices for a range of interest in the data:
i1 = 1500; i2 = 2500;
% find a polynomial fit. (Here, 7th order):
pfit = polyfit(tdata(i1:i2),ydata(i1:i2),7);
% Make sure it matches the real data!
% Here, we will plot encoder data with stair steps,
% to show the discretization better:
figure(1); clf
tt = [tdata, tdata]*0;
tt(1:2:end) = tdata; tt(2:2:end) = tdata+T;
yy = 0*tt;
yy(1:2:end) = ydata; yy(2:2:end) = ydata;
plot(tt,yy,'b-'); hold on
i1a = i1+50; i2a = i2-50;
plot(tdata(i1a:i2a),polyval(pfit,tdata(i1a:i2a)),'r-')
grid on
xlabel('Time (s)')
ylabel('Enconder counts')
legend('Encoder data','7th-order polynomial fit')

% Take the derivative of the polynomial, to get
% VELOCITY. MATLAB can do this via "polyder":
pfitvel = polyder(pfit);
figure(2); clf
plot(tdata(i1a:i2a),polyval(pfitvel,tdata(i1a:i2a)),'b-',...
    'LineWidth',2)
hold on

%------- Problems 3) and 8) Velocity estimates

% To get a smoothed estimate of velocity, let's
% try using a filter "(s/(tau*s + 1))".  After
% experimenting with several values for tau,
% let's go with tau = 0.5 (sec):
tf_s_lowpass = tf([1 0],[.05 1]);

% Now, create a DISCRETE version of this transfer
% function, either by hand or via MATLAB:
tf_filter_z = c2d(tf_s_lowpass,.001,'tustin');
```

```
% Extract numerator and denominator polynomials:
Alowpassvel = tf_filter_z.num{:};
Blowpassvel = tf_filter_z.den{:};
% Now, filter the encoder data to get a low-passed
% estimate of velocity:
dy_lowpass = filter(Alowpassvel,Blowpassvel,ydata);
plot(tdata,dy_lowpass,'r-.','LineWidth',2)

%------- 9) Creative/Improved filters
% NOTE: I was just hoping to get people to be CREATIVE about the notion
% of what a "digital filter" is actually DOING!  Lot's of answers would
% be acceptible here.
% * For instance: a HIGHER-ORDER (low-pass) filter on the velocity...
% * My solution: I tried to be as creative as possible and implemented
% a filter that does a polynomial fit to the last 101 data points and
% uses this to estimate velocity at the CURRENT point.  The advantage
% is there is little or no time delay. The disadvantage is that it is
% such a high-order filter!!  However, it is still "causal" and could be
% implemented on a digital controller.
%
% In practice, if a little time delay doesn't hurt too much, a little
% low-pass filtering wouldn't prevent you from designing a nice controller
% for the overall system...

xp = [-100:0]*.001; % take the last 101 points
yp = 0*xp;
clear p2der
for n=1:length(yp);
    yp=0*xp;  yp(n) =1; % make only THIS data point "non-zero".

    % Below, it turns out each past data point has a LINEAR
    % effect on any order polynomial fit's estimate of VELOCITY
    % at the CURRENT time. This loop finds this linear relationship
    % for each of the previous data points. (Note, you could try
    % higher-order polynomials (above "2"), too... Higher-
    % order polynomials will look less smooth.)

    % Find effect of ONLY THIS POINT being non-zero:
    p2der(n,:) = polyder(polyfit(xp,yp,2));
end
% Since we take derivative at x=0, only the LAST column matters...
% y_est = a*x^2 + b*x + c,   so:
% dy_est = 2*a*x + b, where x=0.
% dy_est(x=0) = b

Apoly = fliplr((p2der(:,end))'); % last col gives "b" vals
Bpoly = 0*Apoly; Bpoly(1) = 1; % use for the CURRENT vel est!
dy_filter_poly = filter(Apoly,Bpoly,ydata);

plot(tdata,dy_filter_poly,'k--','LineWidth',2); grid on
xlabel('Time (s)'); ylabel('Velocity (encoder cts / s)')
legend('Polynomial fit','Prob. 8 answer','Prob. 9 answer')
```