Last time...
(SHV 6.3)
record loops
Desired Position

\[ \vec{x}_{h,\text{des}} = \Lambda(\theta_1,\text{des}, \theta_2,\text{des}, \theta_3,\text{des}) \]

\[ x_{h,\text{des}}, y_{h,\text{des}}, z_{h,\text{des}} \]

Actual Position

\[ \vec{x}_h = \Lambda(\theta_1, \theta_2, \theta_3) \]

\[ x_h, y_h, z_h \]

Proportional Feedback Controller

\[ \vec{F} = k(\vec{x}_{h,\text{des}} - \vec{x}_h) \]

\[ F_x = k(x_{h,\text{des}} - x_h) \]

\[ F_y = k(y_{h,\text{des}} - y_h) \]

\[ F_z = k(z_{h,\text{des}} - z_h) \]
replay loops:
spring force,
fixed kinematics
Mass on a spring: simple harmonic oscillator

Image from http://lifshitz.ucdavis.edu/~dmartin/phy7/7C.html
%% Simulate a mass bouncing on a spring using ode45
% Class example for MEAM 520 on November 29, 2012 by KJK.

clear;

%% Parameters
% Set parameters of the system we want to simulate, noting units.
% Make them global so that the compute derivatives function can see them.
global m k b g
m = 3.5; % kg
k = 400; % N/m
b = 10; % Ns/m
g = 9.81; % m/s^2

%% Time Vector
% Create a time vector. The ' makes it a column vector.
tstart = 0;
tfinal = 3;
tstepmax = 0.01; % Maximum time step, in seconds.

% Time step for graphical output
graphical_tstep = 0.01; % s

%% Initial Conditions
% Define the initial conditions for the mass.
y0 = .2; % m
v0 = 0; % m/s

% Put initial conditions into vector.
X0 = [y0; v0];
%% Initial Conditions
% Define the initial conditions for the mass.
y0 = .2; % m
v0 = 0; % m/s

% Put initial conditions into vector.
X0 = [y0; v0];

%% Simulation
% Show a message to explain how to cancel the graph.
disp('Click in this window and press control-c to cancel simulation of time-domain graph.

% Run the simulation using ode45.
% The state equation function must be in the same directory as this file for Matlab to find it. The @ makes the name a function handle, so it can call it over and over. The other two inputs are the time span and the initial conditions. The outputs are the resulting time vector and the resulting state vector (nx4).
% Here, we set the maximum time step to be tstepmax, to lower the likelihood that the solver will accidentally miss interactions with the interest zones in the world.
options = odeset('MaxStep',tstepmax);
[t, Xhistory] = ode45(@compute_mass_derivatives, [tstart tfinal], X0, options);

%% Plot set up
% The simulation results are not evenly spaced in time, so graphing them directly does not let you see the speed of the puck. Thus, we re-interpolate the data to see where the puck is at evenly spaced times.
\[\Sigma F_y = m\ddot{y}\]

\[-mg - ky - b\dot{y} = m\ddot{y}\]

\[-mg = m\ddot{y} + b\dot{y} + ky\]

\[-g = \ddot{y} + \frac{b}{m}\dot{y} + \frac{k}{m}y\]

**Second-order system**

\[\frac{k}{m} = \omega_n^2\]

\[-g = \ddot{y} + 2\zeta\omega_n\dot{y} + \omega_n^2y\]

\[\frac{b}{m} = 2\zeta\omega_n\]

\[k_{\text{controller}} = m\omega_n^2, \text{desired}\]

\[b_{\text{controller}} = 2m\zeta_{\text{desired}}\omega_n - b_{\text{robot}}\]

\[\zeta_{\text{desired}} = 1\]
\begin{equation*}
\vec{F} = k(\vec{x}_{h,\text{des}} - \vec{x}_h) + b(\vec{v}_{h,\text{des}} - \vec{v}_h)
\end{equation*}

\begin{equation*}
F_x = k(x_{h,\text{des}} - x_h) + b(x_{h,\text{des}} - \dot{x}_h)
\end{equation*}

\begin{equation*}
F_y = k(y_{h,\text{des}} - y_h) + b(y_{h,\text{des}} - \dot{y}_h)
\end{equation*}

\begin{equation*}
F_z = k(z_{h,\text{des}} - z_h) + b(z_{h,\text{des}} - \dot{z}_h)
\end{equation*}

\begin{equation*}
\vec{e}_h = \vec{x}_{h,\text{des}} - \vec{x}_h
\end{equation*}

\begin{equation*}
\dot{\vec{e}}_h = \vec{v}_{h,\text{des}} - \vec{v}_h
\end{equation*}

\begin{equation*}
\vec{F} = k \vec{e}_h + b \dot{\vec{e}}_h
\end{equation*}
Questions ?
Very nice!

Is it perfect?
How can we improve the controller’s tracking?
Add gravity compensation!
How should I compensate for gravity?

mechanically (add weight near the tip)
mechanically (springs)
in software (use the motors)

Which option is better?
Both approaches to gravity compensation are useful.

For haptics, a small amount of software gravity compensation is useful to avoid having to increase the inertia of the system.

For Phantom, first focus on joint 3, because its inherent gravity balance is worse than joint 2.

Gravity compensation is a form of feedforward control (SHV 6.4)
How do we calibrate gravity compensation?
Move the robot slowly through a trajectory and record the torque needed to hold up the weight of the robot.
What motion should we use?
\[ \theta_3 = 0 \]

\[ \theta_3 = -\frac{\pi}{2} = -1.57 \]

\[ \theta_3 = -2.6 \]
% Make t.
   t = linspace(0,10,9000)';

% Make all thetas.
   thetas = zeros(9000,3);
   thetas(:,1) = 1.46;
   thetas(:,2) = -2;
   thetas(:,3) = -2.6;
   thetas(1001:2000,3) = linspace(-2.6,-1.3,1000)';
   thetas(2001:3000,3) = linspace(-1.3,-2.6,1000)';
   thetas(3001:4000,3) = linspace(-2.6,-1.3,1000)';
   thetas(4001:5000,3) = linspace(-1.3,-2.6,1000)';
   thetas(5001:6000,3) = linspace(-2.6,-1.3,1000)';
   thetas(6001:7000,3) = linspace(-1.3,-2.6,1000)';
   thetas(7001:8000,3) = linspace(-2.6,-1.3,1000)';
   thetas(8001:9000,3) = linspace(-1.3,-2.6,1000)';

% Plot thetas.
   figure(6);
   plot(t,thetas)
Figure 1

Link lengths in millimeters:

11 = 168;
12 = 140;
13 = 140;
The graph shows the recorded data (blue line) and the approximated line $-0.032 \sin(\theta_3)$ (red line) against $\theta_3$ (rad) and $\gamma_3$ (Nm). The data points fluctuate around the approximated line, indicating a trigonometric relationship between $\gamma_3$ and $\theta_3$. The graph is used to validate the accuracy of the approximated formula against experimental data.
Can we make this better?
% Make t.
t = linspace(0,10,9000)';

% Make all thetas.
thetas = zeros(9000,3);
thetas(:,1) = 1.46;
thetas(:,2) = -2;
thetas(:,3) = -2.6;
thetas(1001:2000,3) = linspace(-2.6,-1.3,1000)';
thetas(2001:3000,3) = -1.3;
thetas(3001:4000,3) = linspace(-1.3,-2.6,1000)';
thetas(4001:5000,3) = -2.6;
thetas(5001:6000,3) = linspace(-2.6,-1.3,1000)';
thetas(6001:7000,3) = -1.3;
thetas(7001:8000,3) = linspace(-1.3,-2.6,1000)';
thetas(8001:9000,3) = -2.6;

% Plot thetas.
figure(6);
plot(t,thetas)
The graph shows a comparison between the "Recorded Data" (blue line) and the line defined by the equation $-0.032 \sin(\theta_3)$ (red line). The x-axis represents $\theta_3$ in radians, ranging from $-2.6$ to $-1.2$, while the y-axis shows $\gamma_3$ (Nm), ranging from 0 to 0.06. The recorded data fluctuates significantly, while the sinusoidal line provides a smooth trend for comparison.
Can we make this better?
Trajectory Smoothing

\[ q(t_0) = q_0 \quad \rightarrow \quad q(t_f) = q_f \]
\[ \dot{q}(t_0) = v_0 \quad \rightarrow \quad \dot{q}(t_f) = v_f \]

cubic polynomial

\[ q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \]

\[
\begin{bmatrix}
q_0 \\
v_0 \\
q_f \\
v_f 
\end{bmatrix} =
\begin{bmatrix}
1 & t_0 & t_0^2 & t_0^3 \\
0 & 1 & 2t_0 & 3t_0^2 \\
1 & t_f & t_f^2 & t_f^3 \\
0 & 1 & 2t_f & 3t_f^2 
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3 
\end{bmatrix}
\]
% Make an abbreviated time vector with 1000 elements.
t = 1:1000;

% Define initial conditions.
q0 = -2.6;
v0 = 0;

% Define final conditions.
qf = -1.3;
vf = 0;

% Put initial and final conditions into a vector.
conditions = [q0; v0; qf; vf];

% Put time elements into matrix.
mat = [1  t(1)  t(1)^2  t(1)^3;
       0  1    2*t(1)  3*t(1)^2;
       1 t(end) t(end)^2  t(end)^3;
       0  1    2*t(end) 3*t(end)^2];

% Solve for coefficients.
as = mat \ conditions;

% Pull individual coefficients out.
a0 = as(1);
a1 = as(2);
a2 = as(3);
a3 = as(4);
vf = 0;

% Put initial and final conditions into a vector.
conditions = [q0; v0; qf; vf];

% Put time elements into matrix.
mat = [1  t(1)  t(1)^2  t(1)^3;
      0  1    2*t(1)  3*t(1)^2;
      1  t(end) t(end)^2  t(end)^3;
      0  1    2*t(end) 3*t(end)^2];

% Solve for coefficients.
as = mat \ conditions;

% Pull individual coefficients out.
a0 = as(1);
a1 = as(2);
a2 = as(3);
a3 = as(4);

% Calculate cubic trajectory with coefficients.
q = a0 + a1*t + a2*t.^2 + a3 * t.^3;

% Plot cubic trajectory.
figure(2)
plot(t,q)
xlabel('Time (index)')
ylabel('theta3')
34    figure(2)
35    plot(t,q)
36    xlabel('Time (index)')
37    ylabel('theta3')
38
39
40    % Make t.
41    t = linspace(0,10,9000)';
42
43    % Make all thetas.
44    thetas = zeros(9000,3);
45    thetas(:,1) = 1.46;
46    thetas(:,2) = -2;
47    thetas(:,3) = -2.6;
48    thetas(1001:2000,3) = q';
49    thetas(2001:3000,3) = -1.3;
50    thetas(3001:4000,3) = flipud(q');
51    thetas(4001:5000,3) = -2.6;
52    thetas(5001:6000,3) = q';
53    thetas(6001:7000,3) = -1.3;
54    thetas(7001:8000,3) = flipud(q');
55    thetas(8001:9000,3) = -2.6;
56
57    % Plot thetas.
58    figure(6);
59    plot(t,thetas)
60    xlabel('Time (s)')
61    ylabel('Joint Angle (rad)')
62    legend('theta1', 'theta2', 'theta3')
Questions ?
Can we make this better?
What causes these squiggles?
cal3new

The graph illustrates the relationship between $\theta_3$ (rad) and $\gamma_3$ (Nm). The blue line represents the recorded data, while the red line shows $-0.032 \sin(\theta_3)$. The graph shows a clear trend of the recorded data following the model closely.
Recorded Data
- 0.032 \sin(\theta_3)
What is going on?