Table of Contents

```
% compare_sim_exp.m
% Compare simulation and compare data
```

This section plots and filters collected data

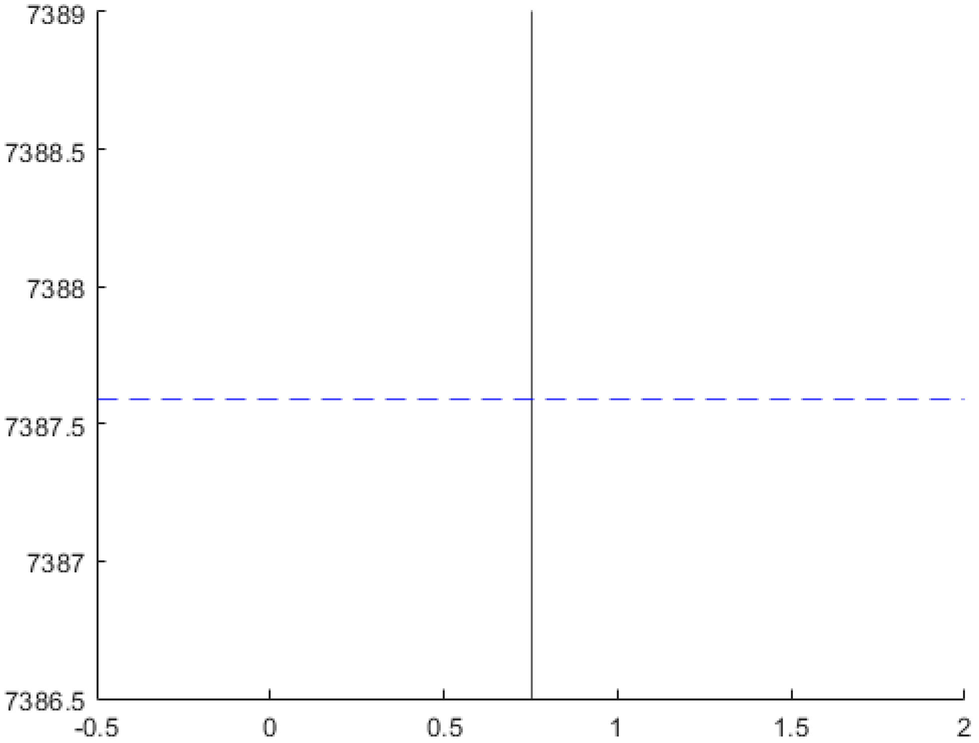
```
pl_flag = false; % true (1) or false (0) to use to turn plotting on or off

% Clear Variables
clear vel time_exp velFilteredRealTime

% calculate flywheel mass properties
[param_var.j_eff , param_var.mfw, param_var.Tau_f] =
 flywheel_mass_prop(config, param_var, param_fixed);

filename = config_ar(iconfig).exp_data_filename;  % 'testA100.txt';
fileID = fopen(filename);
collectedData = readmatrix(filename);
time_exp = collectedData(1:end,1)./1e6;    % [s]
position_exp = collectedData(1:end,2);    % [counts]
npt = length(time_exp);

% Velocity calculation
for ipt = 1:npt-1

    vel(ipt) = (position_exp(ipt+1) - position_exp(ipt)) / ...
        (time_exp(ipt+1) - time_exp(ipt));

end  % end velocity calculation

if pl_flag == true

    figure(2)
    hold on
    plot(time_exp,position_exp)
    xlabel('Time [s]')
    ylabel('Angular Position [counts]')
    title('Experimental Motor Position vs. Time')

    figure(3)
    plot(time_exp(2:end),vel*60/48, 'Color',[0.9290 0.6940 0.1250])  % Note,
 60/48 converts counts/rev to rpm
    xlabel('Time [s]')
    ylabel('Angular Velocity RPM)')
    title('Experimental Motor Velocity vs. Time')
    ylim([0,inf])

end % end plotting of position and unfiltered velocity data

velFilteredRealTime(1) = vel(1);
alpha = 0.04;  % ranges form 0 to 1

for ipt = 2:length(vel)  % start at 2

    velFilteredRealTime(ipt) = alpha*vel(ipt) + (1-
alpha)*velFilteredRealTime(ipt-1);

end % end filtering velocity loop

% Plotting filtered experimental velocity
if pl_flag == true

    figure(1)
    plot(time_exp(2:end),velFilteredRealTime*60/48, 'Color',[0.4940 0.1840
 0.5560])
    hold on

end % end plotting filtered velocity
```

Motor simulation

```
% test motor_sim function
[w_sim_ar, t_sim_ar, tr_sim, wterm_sim] = motor_sim_ODE45(config, param_var,
 param_fixed);
```

Use find_metrics to determine rise time and term velocity

```
[tr_exp, wterm_exp] = find_metrics(velFilteredRealTime*60/48,time_exp(2:end));
xline(tr_exp);
yline(wterm_exp, 'b--');
```



Error metric calculation

```
% FIXME - Calculate the percent error and error metric
w_term_err = abs(wterm_exp-wterm_sim)/wterm_exp; % percent error in terminal velocity
tr_err = abs(tr_exp-tr_sim)/tr_exp; % percent error in rise time
err_metric = tr_err + 4*w_term_err; % error metric
```

Title plot and label axis

```
if pl_flag == true

    figure(1)
    title_line1 = ['Experimental Velocity (Filtered) vs Simulated Velocity (ODE45) [RPM]'];
    title_line2 = ['Wterm error = ' num2str(w_term_err) '%, tr error = ' num2str(tr_err) '%, Error metric = ' num2str(err_metric)]; % display terminal velocoty with 4 significant figures
    title_line3 = [config(1).name ', Jm = ' num2str(param_var.jm) ' (kgm^2), Cd = ' num2str(param_var.cd) ', mu = ' num2str(param_var.mu)];
    title({title_line1;title_line2;title_line3})

    xlabel('Time [s]');
    ylabel('Angular Velocity [rpm]');
    legend('Experimental','Simulation','Location','best')
```

```
end % end labeling axis and titles
```

Store data in array to compare between configurations

```
tr_exp_ar(iconfig) = tr_exp;
wterm_exp_ar(iconfig) = wterm_exp;
tr_sim_ar(iconfig) = tr_sim;
wterm_sim_ar(iconfig) = wterm_sim;
tr_err_ar(iconfig) = abs(tr_exp-tr_sim)/tr_exp;
wterm_err_ar(iconfig) = abs(wterm_exp-wterm_sim)/wterm_exp;
err_metric_ar(iconfig) = err_metric;
```

Published with MATLAB® R2021b