Table of Contents

Optimized ODE45 Motor Model 1
Receive User Inputs 1
Calculate Flywheel Parameters 1
Open Collecteded Data 1
Calculate Velocity 2
Calculate Filtered Velocity 2
Plot Experimental Filtered Velocity 2
Motor Simulation and Error Metrics 3

Optimized ODE45 Motor Model

```
%MAE 156A Motor Modeling

clear; clc; close all
```

Receive User Inputs

```
    config.nut_ar = [2 2 2 2 2 2 2 2]; % Receive nut configuration
    % This is a user input in code, publishing feature does not allow
    config.pwm = 100;
    % This is a user input in code, publishing feature does not allow
    param_var.jm = 1.1e-6; % Motor shaft inertia [kg*m^2]
    param_var.mu = 0.254; % Coefficient of Friction
    param_var.cd = 0.67; % Coefficient of Drag
    param_var.tau_f = 0; % Motor friction [Nm]

    param_fixed.ngear = 4.4;                    % Gear ratio
    param_fixed.wn = 8200*2*pi/60;              % No load motor speed at 100%
 PWM in [rad/s]
    param_fixed.trq_stall = 0.17*0.0980665;     % Motor stall torque at 100%
 PWM [Nm]
    param_fixed.nut_mass = 3.20e-3;             % Mass of a single nut [kg]
    param_fixed.bolt_mass = 7.74e-3;            % Mass of a single bolt [kg]
    param_fixed.fw_mass = 5.53e-2;              % Flywheel mass [kg]
    param_fixed.hub_mass = 3.84e-3;             % Hub mass [kg]
    param_fixed.r_disk = 0.057;                 % Radius of disk [m]
    param_fixed.r_dis = 0.0508;                 % Distance of nut from center
 of flywheel [m]
```

Calculate Flywheel Parameters

```
    [param_var.j_eff , param_var.mfw, param_var.Tau_f] =
 flywheel_mass_prop(config, param_var, param_fixed);
```

Open Collecteded Data

```
    collectedData = readmatrix('EncoderData_Config33.txt');
    time_exp = collectedData(1:end,1)./1e6;    % [s]
    position_exp = collectedData(1:end,2);     % [counts]
    npt = length(time_exp);
```

Calculate Velocity

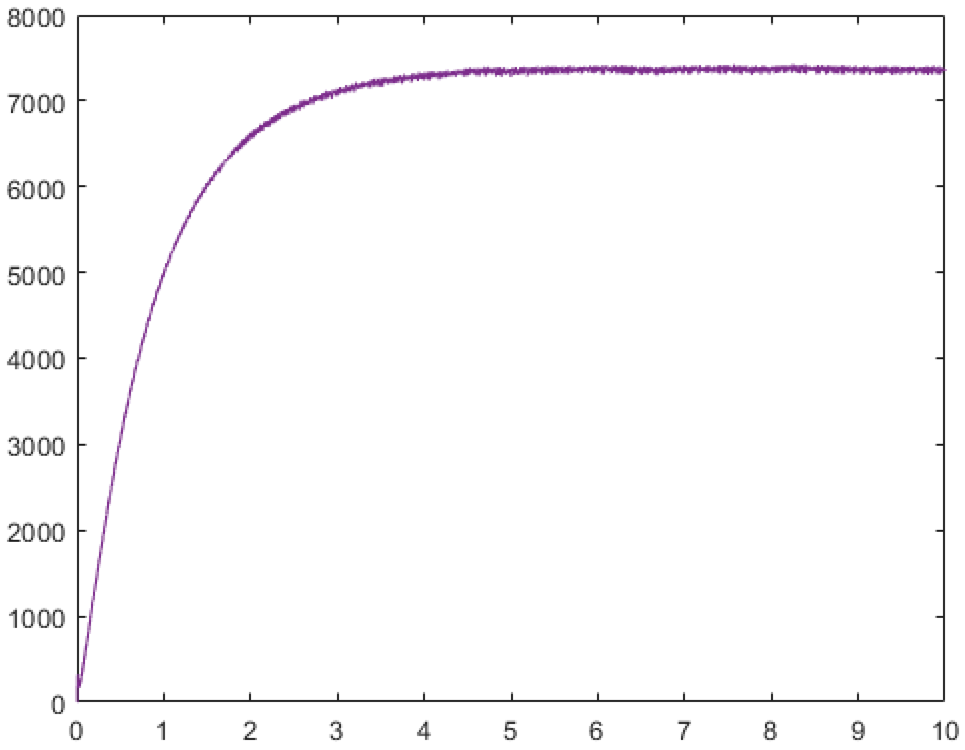
```
    for ipt = 1:npt-1
        vel(ipt) = (position_exp(ipt+1) - position_exp(ipt)) / ...
            (time_exp(ipt+1) - time_exp(ipt));
    end
```

Calculate Filtered Velocity

```
    velFilteredRealTime(1) = vel(1);
    alpha = 0.04;  % ranges from 0 to 1
    for ipt = 2:length(vel)
        velFilteredRealTime(ipt) = alpha*vel(ipt) + (1-
alpha)*velFilteredRealTime(ipt-1);
    end
```

Plot Experimental Filtered Velocity

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



Motor Simulation and Error Metrics

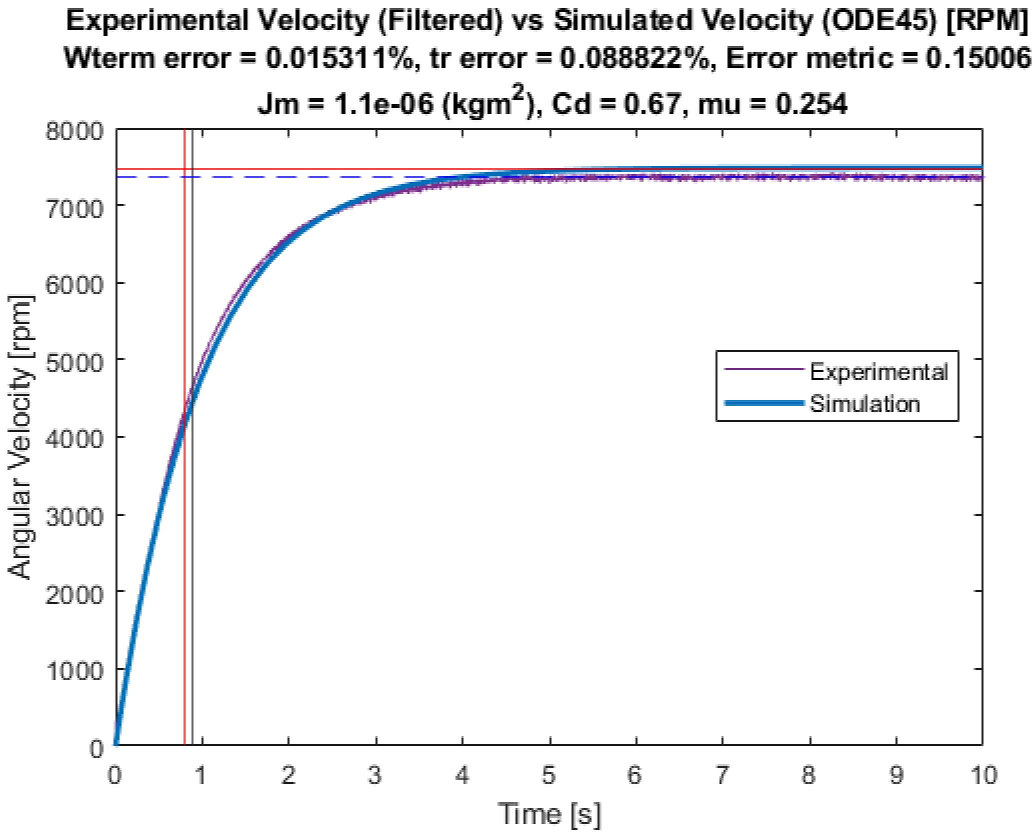
```
   [w_sim_ar, t_sim_ar, tr_sim, wterm_sim] = motor_sim_ODE45(config,
param_var, param_fixed);

   [tr_exp, wterm_exp] =
find_metrics(velFilteredRealTime*60/48,time_exp(2:end));
   figure(1)
   xline(tr_exp);
   yline(wterm_exp, 'b--');

   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

   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 = [' 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')
```



Published with MATLAB® R2021b