Make graphs

Trevor Burgoyne 13 Nov 2022

```
function [avg_hss, var_hss, avg_kt, var_kt, kphat, kdhat, kp, kd] = make_graphs(ROOT_DIR, PREFIX, DISP_NAME, LABEL_NAME, N_TESTS, N_RUNS, masses)
    COLORS = ["red", "blue", "green", "black"];
    T_START = 20; % s
    T_STEADY = 30; % s, Chosen as start of steady-state response from observation
    T_END = T_START+25; % s
    HREF = 1.25; % m
    g = 9.81; % m/s^2

    % Plotting options
    font_size = 12;
    line_size = 15;
    line_width = 1;
    avg_hss = zeros(1,N_TESTS);
    var_hss = zeros(1,N_TESTS);
    avg_kt  = zeros(1,N_TESTS);
    var_kt  = zeros(1,N_TESTS);
    kphat   = zeros(1,N_TESTS);
    kdhat   = zeros(1,N_TESTS);
    kp      = zeros(1,N_TESTS);
    kd      = zeros(1,N_TESTS);
    for test_n=1:N_TESTS
        ts = zeros(1,N_RUNS);
        kt = zeros(1,N_RUNS);
        hss = zeros(1,N_RUNS);
        zs = zeros(1,N_RUNS);
        figure(test_n)
        hold on
        for run_n=1:N_RUNS
            % Load data
            path = ROOT_DIR + PREFIX + test_n;
            if N_RUNS > 1 % Add runs suffix if more than one run was done
                path = path + "R" + run_n;
            end
            path = path + ".mat";
            load(path);

            if run_n == 1 % Only plot reference once
                plot(time,-z_ref,'Linewidth',line_width,'Color',COLORS(4),'DisplayName','z_{ref}');
            end
            plot(time,-z_est,'Linewidth',line_width,'Color',COLORS(run_n),'DisplayName',LABEL_NAME + " " + run_n);

            % Steady-state error
            idxs = find(time >= T_STEADY & time <=T_END); % Indices of steady state region
            z_arr = -z_est(idxs); % Z values being investigated
            zs(run_n) = double(mean(z_arr)); % m, Experimental settling value
            hss(run_n) = zs(run_n) - HREF; % m, Steady state error

            % Settling time
            idxs = find(time >= T_START & time <=T_END); % Idxs of investigated response
            start_idx = idxs(1);
            z_arr = -z_est(idxs); % Z values being investigated

            % Find last time z dipped below 95% of z_settle
            ts_idxs = find(z_arr <= 0.95*zs(run_n));
            if isempty(ts_idxs)
                ts1 = 0;
            else
                ts1 = time(ts_idxs(end) + start_idx);
            end

            % Find last time z rose above 105% of z_settle
            ts_idxs = find(z_arr >= 1.05*zs(run_n));
            if isempty(ts_idxs)
                ts2 = 0;
            else
                ts2 = time(ts_idxs(end) + start_idx);
            end

            ts(run_n) = max(ts1,ts2); % s, Settling time (use the later time)

            % Kt calculation
```

```
            motors = abs([motor1' motor2' motor3' motor4']); % Get all motor values
            u = mean(motors);
            kt(run_n) = (masses(test_n)*g) / (4*u) * 1000; % N

        end

        scale = [1.05 0.95]; % scalar values used to position text
        for run_n=1:N_RUNS
            % Steady-state error
            if zs(run_n) == max(zs) % Position labels above if the line is higher
                y = scale(1);
            else
                y = scale(2);
            end
            line_name = "h_{ss} = " + hss(run_n);
            text(T_STEADY,y*zs(run_n),line_name,'Color',COLORS(run_n))
            yline(zs(run_n),"--",'Linewidth',line_width,'Color',COLORS(run_n),'HandleVisibility','off')

            % Settling time
            if ts(run_n) == max(ts) % Position labels above if the line is higher
                y = scale(1);
            else
                y = scale(2);
            end
            line_name = "ts = " + (ts(run_n)-T_START);
            text(1.01*max(ts),y*.7,line_name,'Color',COLORS(run_n))
            xline(ts(run_n),"--",'Linewidth',line_width,'Color',COLORS(run_n),'HandleVisibility','off')
        end
        title(sprintf('%s: $\\hat{K_{p}} = %s, \\hat{K_{d}} = %s$',DISP_NAME,num2str(Kp),num2str(Kd)),'Interpreter','latex');
        xlabel('Time (s)','fontsize',font_size);
        ylabel('Altitude (m)','fontsize',font_size);
        legend('show','Location','best');
        set(gca,'XMinorGrid','off','GridLineStyle','-','FontSize',line_size)
        xlim([T_START-1 T_END+1]);
        ylim([0.4 1.5]);
        grid on

        avg_hss(test_n) = mean(hss);
        var_hss(test_n) = var(hss);
        avg_kt(test_n)  = mean(kt);
        var_kt(test_n)  = var(kt);
        kphat(test_n)   = Kp;
        kdhat(test_n)   = Kd;
        kp(test_n)      = Kp/(4*avg_kt(test_n));
        kd(test_n)      = Kd/(4*avg_kt(test_n));
    end
end
```

```
Not enough input arguments.

Error in make_graphs (line 16)
    avg_hss = zeros(1,N_TESTS);
```

Published with MATLAB® R2020b

Contents

PID Control of Quadcopter Altitude Near Hover

Acknowledgement: Prof. Peter Seiler

```
%-------------------------------------------------
function [wn, zeta] = Low_Fidelity_Model(DISP_NAME, m, kT, Kp, Kd)
```

Vehicle Parameters

close all; clear all; m = 65e-3; % Mass, kg

```
    T_START = 19; % s
    T_STEADY = 30; % s, Chosen as start of steady-state response from observation
    T_END = T_START+25; % s
    g = 9.81; % m/s^2
    % kT = ; % Thrust coefficient, N
    % kT = input('Enter the estimated thrust coefficient, N, kT: ');
    umax = 500; % Maximum motor input command, unitless
    umin = 0; % Minimum motor input command, unitless
```

Step change in altitude reference, m

```
    Tf    = 25;           % Final simulation time, sec
    hdes0 = 0.7;          % Initial altitude, m
    hdesf = 1.25;         % Final altitude, m
```

Initial Conditions

```
    h0 = hdes0;        % Initial altitude, m
    hdot0 = 0;         % Initial altitude velocity, m/s
    hddmax = (4*kT*umax-m*g)/m;    % Maximum upward acceleration, m/s^2
```

```
Not enough input arguments.

Error in Low_Fidelity_Model (line 26)
    hddmax = (4*kT*umax-m*g)/m;    % Maximum upward acceleration, m/s^2
```

PD Control With Gravity Feedforward

Consider the following PD control law with gravity feedforward u = Kp*(hdes-h) - Kd*hdot + (mhat*g)/(4*kThat) Here u is the ***individual motor command*** input (unitless). Also, (mhat,kThat) indicate that these are estimates used by the controller. The parrot drone Simulink diagram uses:5 uhat = Kphat*(hdes-h) - Kdhat*hdot + mhat*g where uhat is the ***total thrust*** command (N). These are related by uhat = (4*kT)*u Thus gains computed using the simplified second-order model must be scaled by (4*kT) before implementing in the parrot Simulink diagram.

NOTE - An integral gain is included below for testing but this can be set to zero to simply to the PD control given above.

```
% Select Gains -- Gains are specified using simplified second order model

GainCase = 2;
switch GainCase
    case 1
        % Select closed-loop natural frequency and damping ratio
        wn = input('Desired natural freq, rad/sec, wn: ');      % Desired natural freq, rad/sec
        zeta = input('Desired damping ratio, unitless, zeta: ' );  % Desired damping ratio, unitless

        % Closed-loop ODE with PD control + perfect gravity cancellation:
        %   m hdd = (4*kT)*( Kp*(hdes-h) - Kd*hdot )
        % --> hdd + (4*kT*Kd/m) hdot + (4*kT*Kp/m) h = (4*kT*Kp/m)*hdes
        Kp = wn^2*m/(4*kT);
        Kd = 2*zeta*wn*m/(4*kT);
        Ki = 0;

    case 2
        % Kp = input('Enter the proportional gain, Kp: ');
        % Kd = input('Enter the derivative gain, Kd: ');
        Ki = 0;

        wn = sqrt(Kp*4*kT/m);
        zeta = 4*kT*Kd/(2*wn*m);
end
```

Closed-Loop Transfer Function (hdes to h)

This constructs the closed-loop model from altitude reference to altitude. It assumes perfect cancellation of the gravitational force by the feedforward term. It also neglects the saturation that limits the motor command.

```
% Plant dynamcs (neglecting gravity term)
%   m d^2(h)/dt^2 = (4*kT)*u
% The state-space model below has input motor command and outputs [h;hdot]
Ag = [0 1; 0 0];
Bg = [0; 4*kT/m];
Cg = eye(2);
Dg = 0;
G = ss(Ag,Bg,Cg,Dg);
time = [];
h_arr = [];

tstep = 1;   % Step time, sec
% Controller: PI and D terms
if Ki==0
    Cpi = ss(Kp);
else
    Cpi = tf([Kp Ki],[1 0]);
end
Cd = ss(Kd);

% Form closed-loop from r to h, T
systemnames = 'G Cpi Cd';
inputvar = '[r]';
outputvar = '[G(1)]';
input_to_G = '[Cpi+Cd]';
input_to_Cpi = '[r-G(1)]';
input_to_Cd = '[-G(2)]';
%T = sysic;

if false
    % Code to debug:  Represent controller
    % (*)  u = Kp e + Kd edot + Ki integral(e)
    % The actual implementation only uses rate-feedback, i.e. Kd*edot is
    % replaced by -Kd*hdot.  However, the form in (*) above is much easier
    % to construct with standard functions and will yield the same poles
    % (only the closed-loop zeros will be different)
```

```
        Gb = tf(4*kT/m,[1 0 0]);
        if Ki==0
            Cpid = tf([Kd Kp],1);
        else
            Cpid = tf([Kd Kp Ki],[1 0]);
        end
        Tb = feedback(Gb*Cpid,1);
    end
```

Convert Gains

There are two sets of gains: 1) (Kd,Kp,Ki) for simple second order model with motor command input 2) (Kdhat,Kphat,Kihat) for parrot drone simulink model with total thrust command input. These are the gains that should be used in the parrot drone Simulink model.

```
Kdhat = (4*kT)*Kd;
Kphat = (4*kT)*Kp;
Kihat = (4*kT)*Ki;
```

Simulate Closed-Loop

```
% Parameter estimates
mhat = m;
kThat = kT;

% Disturbance Force, N
Fd = 0;

% Simulate system

% Allow sim to be called as a function
% and have proper variable scope
options = simset('SrcWorkspace','current');
sim('QuadPID',[0 Tf],options);
```

Plot results

```
font_size = 12;
line_size = 15;
line_width = 1;
color = 'b';

figure();
hold on
text(T_STEADY,1,"w_{n} = " + wn + newline + "zeta = " + zeta);
yline(hdesf,'Linewidth',line_width,'Color','black','DisplayName','z_{ref}')
plot(tsim+T_START,h,'Linewidth',line_width,'Color',color,'DisplayName','Low Sim');
title(sprintf('%s: $\\hat{K_{p}} = %s, \\hat{K_{d}} = %s$',DISP_NAME,num2str(Kphat),num2str(Kdhat)),'Interpreter','latex');
xlabel('Time (s)','fontsize',font_size);
ylabel('Altitude (m)','fontsize',font_size);
legend('show','Location','best');
set(gca,'XMinorGrid','off','GridLineStyle','-','FontSize',line_size)
xlim([T_START T_END+1]);
ylim([0.4 1.5]);
grid on;




time  = [time tsim];
h_arr = [h_arr h];

% Steady-state error
idxs = find(time >= T_STEADY-T_START); % Indices of steady state region
z_arr = h_arr(idxs); % Z values being investigated
```

```
    zs = double(mean(z_arr)); % m, Experimental settling value
    hss = zs - hdesf; % m, Steady state error

    line_name = "h_{ss} = " + hss;
    text(T_STEADY,1.05*zs,line_name,'Color',color)
    yline(zs,"--",'Linewidth',line_width,'Color',color,'HandleVisibility','off')

    % Settling Time
    if zeta < 0.5 % Underdamped case
        ts = 3/(zeta*wn);
        line_name = "ts = " + (ts-1) + " (theory)";
        text(1.01*(ts+T_START),.6,line_name,'Color','r')
        xline(ts+T_START,"--",'Linewidth',line_width,'Color','r','HandleVisibility','off')
    end

    % Find last time z dipped below 95% of z_settle
    ts_idxs = find(h_arr <= 0.95*zs);
    if isempty(ts_idxs)
        ts1 = 0;
    else
        ts1 = time(ts_idxs(end));
    end

    % Find last time z rose above 105% of z_settle
    ts_idxs = find(h_arr >= 1.05*zs);
    if isempty(ts_idxs)
        ts2 = 0;
    else
        ts2 = time(ts_idxs(end));
    end

    ts = max(ts1,ts2); % s, Settling time (use the later time)

    line_name = "ts = " + (ts-1);
    text(1.01*(ts+T_START),.6,line_name,'Color',color)
    xline(ts+T_START,"--",'Linewidth',line_width,'Color',color,'HandleVisibility','off')
```

```
end
```

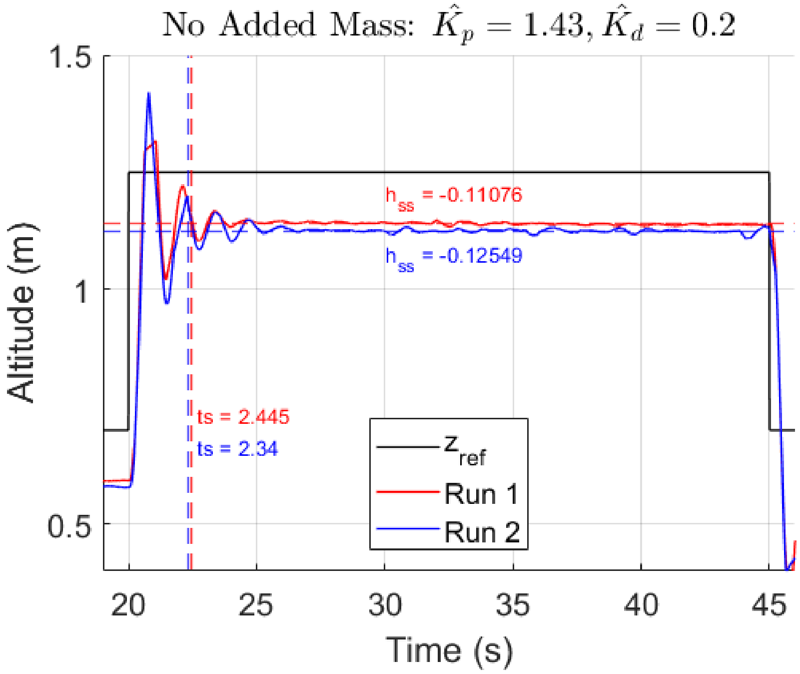
Published with MATLAB® R2020b

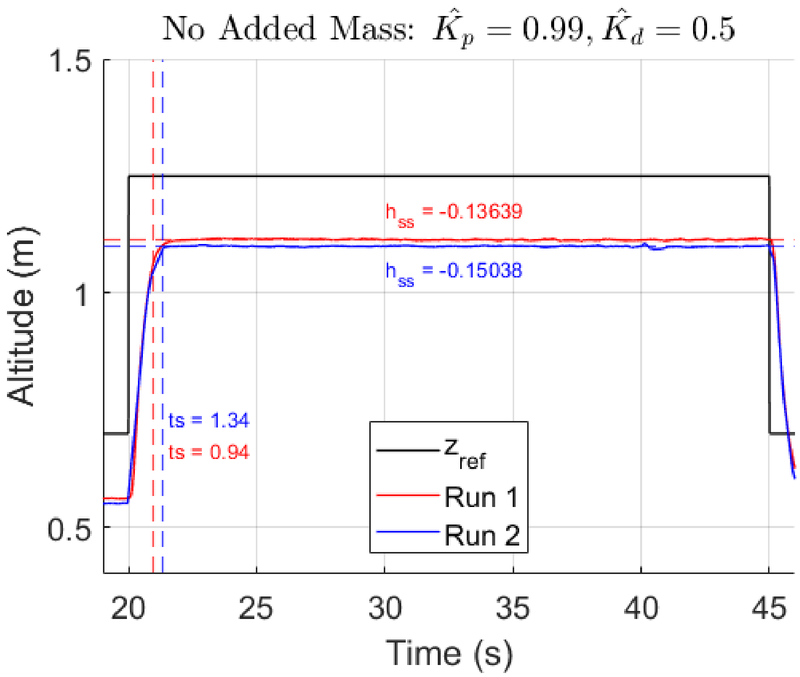
Flight Experiments

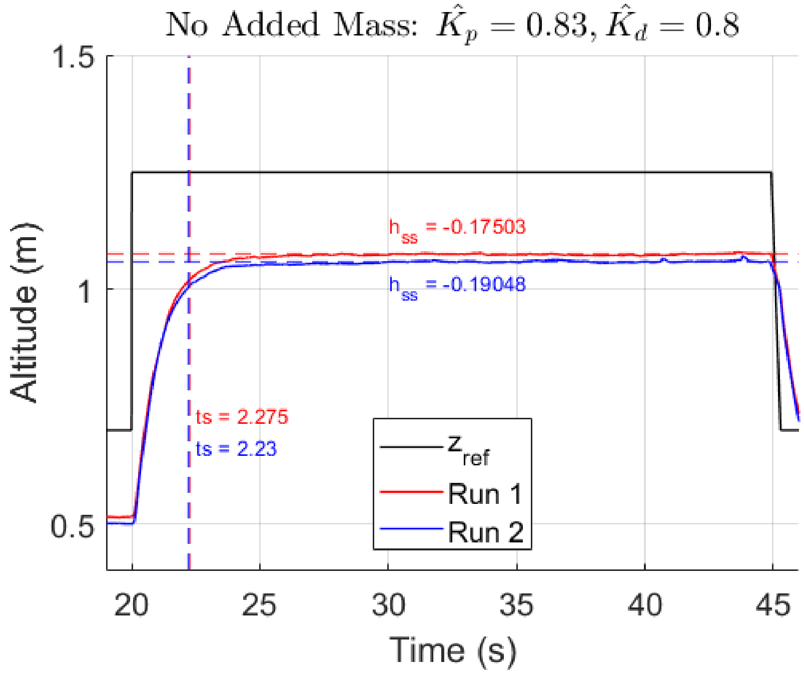
Trevor Burgoyne 13 Nov 2022

```
% Paths for data loading
ROOT_DIR = "C:/Users/Trevor/Desktop/AEM 4602W/Controls-Lab/Data/Part 1/";
PREFIX = "pjsdata_T";
DISP_NAME = "No Added Mass";
LABEL_NAME = "Run";
N_TESTS = 3;
N_RUNS = 2;
masses = [68.1, 68, 69.5] / 1000; % kg
[avg_hss, var_hss, avg_kt, var_kt, kphat, kdhat, kp, kd] = make_graphs(ROOT_DIR, PREFIX, DISP_NAME, LABEL_NAME, N_TESTS, N_RUNS, masses)
% Generate Low Fidelity Model Plots
for test_n=1:N_TESTS
    [wn, zeta] = Low_Fidelity_Model(DISP_NAME, masses(test_n), avg_kt(test_n), kp(test_n), kd(test_n));
end
```

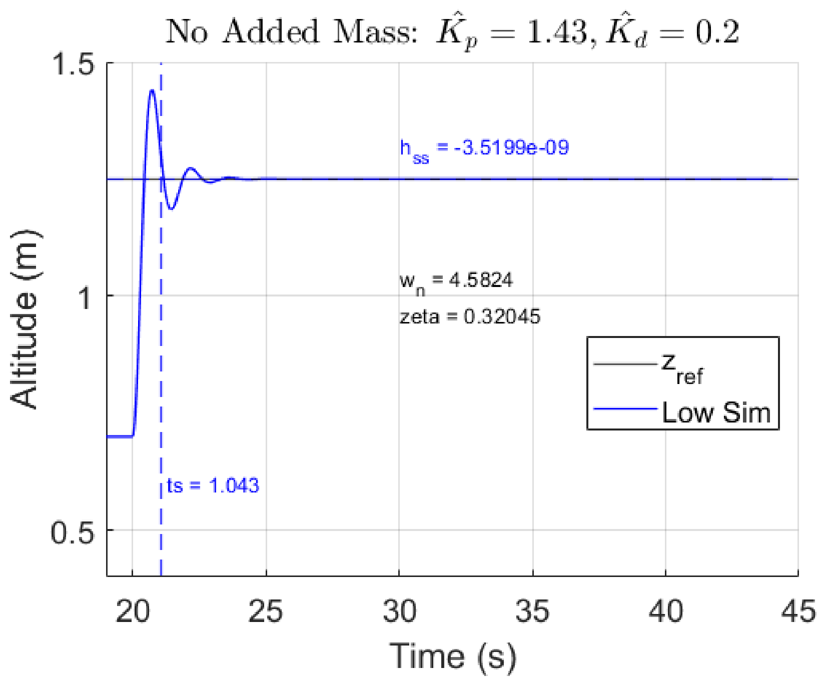
```
avg_hss =

   -0.1181   -0.1434   -0.1828


var_hss =

   1.0e-03 *

    0.1084    0.0977    0.1194


avg_kt =

    0.5240    0.5358    0.5237


var_kt =

   1.0e-04 *

    0.6498    0.4380    0.5865


kphat =

    1.4300    0.9900    0.8300


kdhat =

    0.2000    0.5000    0.8000


kp =

    0.6823    0.4619    0.3962


kd =

    0.0954    0.2333    0.3819
```


No Added Mass: $\hat{K}_p = 1.43, \hat{K}_d = 0.2$
Altitude (m)
Time (s)
h_{ss} = -0.11076
h_{ss} = -0.12549
ts = 2.445
ts = 2.34
z_{ref}
Run 1
Run 2

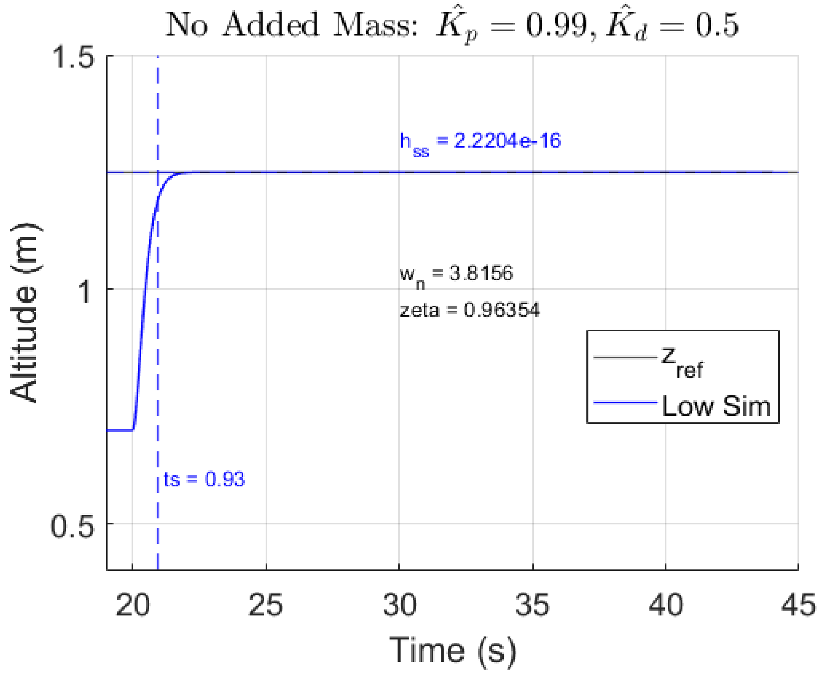

No Added Mass: $\hat{K}_p = 0.99, \hat{K}_d = 0.5$
Altitude (m)
Time (s)
h_{ss} = -0.13639
h_{ss} = -0.15038
ts = 1.34
ts = 0.94
z_{ref}
Run 1
Run 2

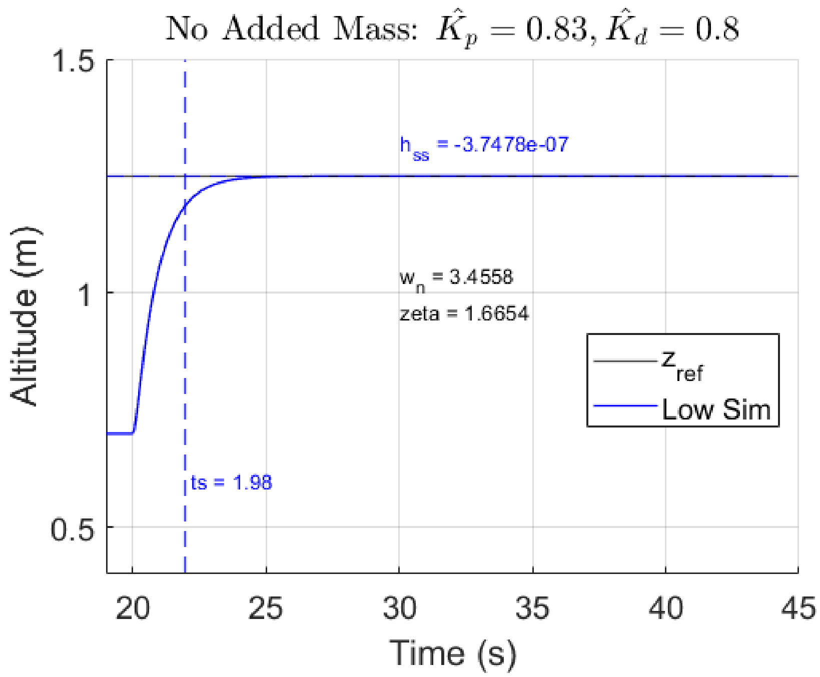


No Added Mass: $\hat{K}_p = 0.83, \hat{K}_d = 0.8$
1.5
1
0.5
Altitude (m)
h_{ss} = -0.17503
h_{ss} = -0.19048
ts = 2.275
ts = 2.23
z_{ref}
Run 1
Run 2
20
25
30
35
40
45
Time (s)



No Added Mass: $\hat{K}_p = 1.43, \hat{K}_d = 0.2$
1.5
1
0.5
Altitude (m)
h_{ss} = -3.5199e-09
w_n = 4.5824
zeta = 0.32045
z_{ref}
Low Sim
ts = 1.043
20
25
30
35
40
45
Time (s)





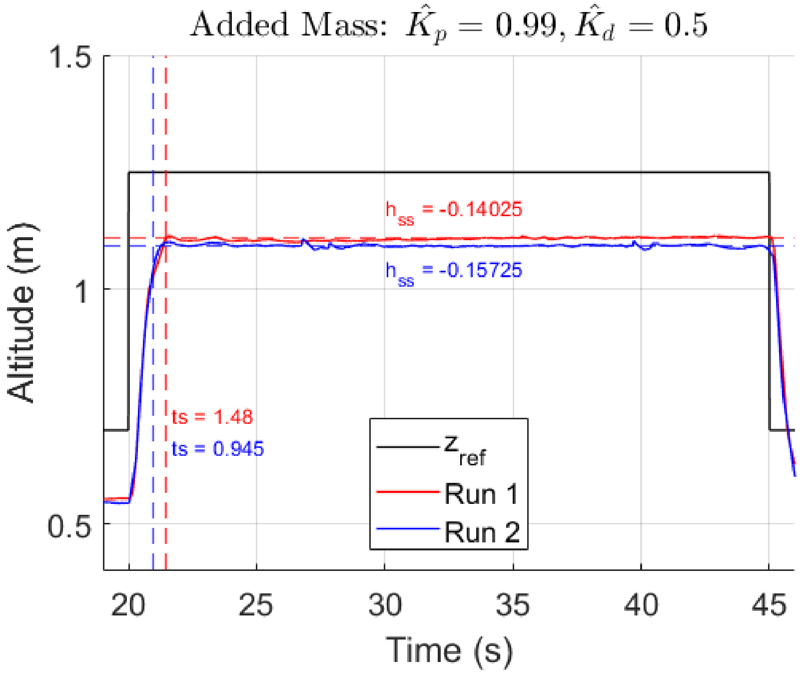
Published with MATLAB® R2020b

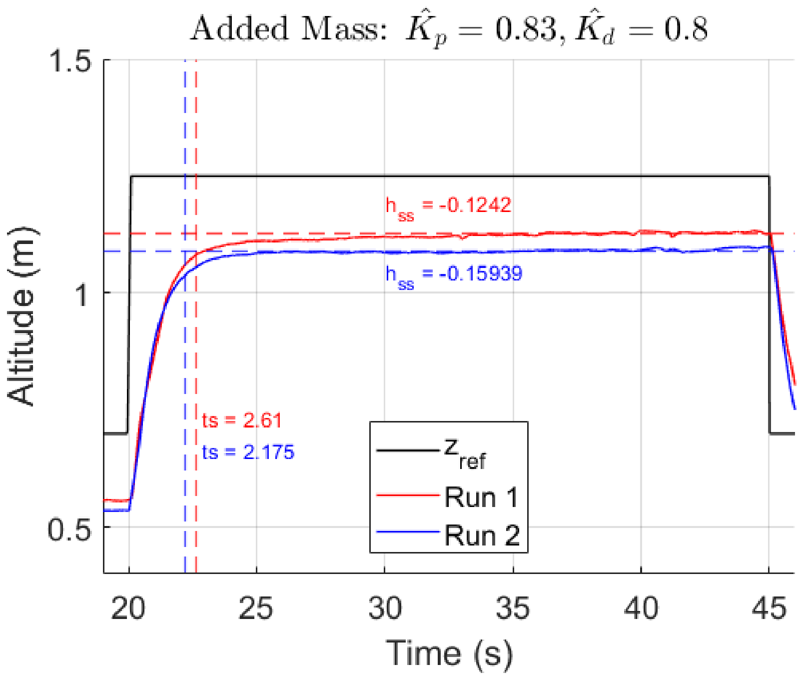
Added Mass Experiments

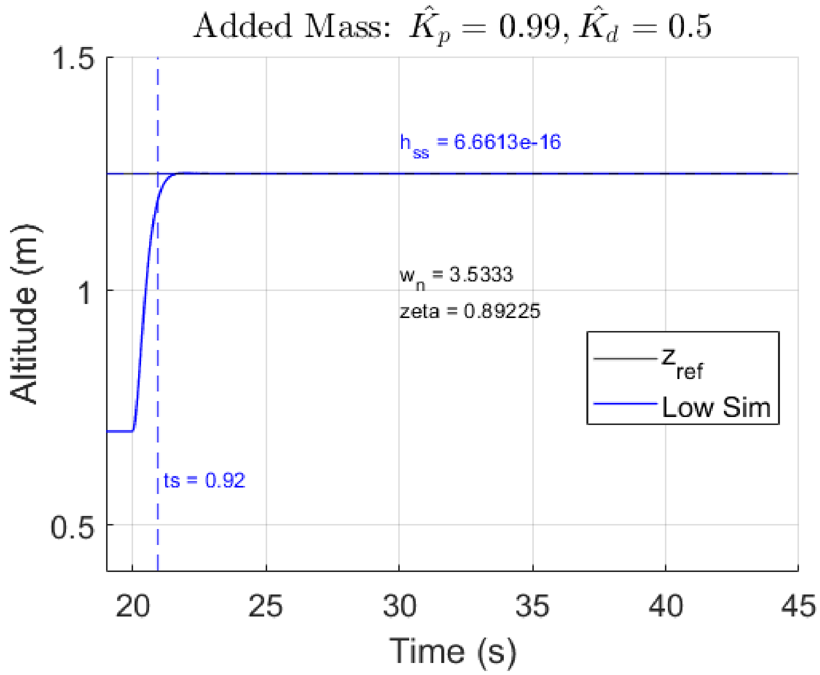
Trevor Burgoyne 13 Nov 2022

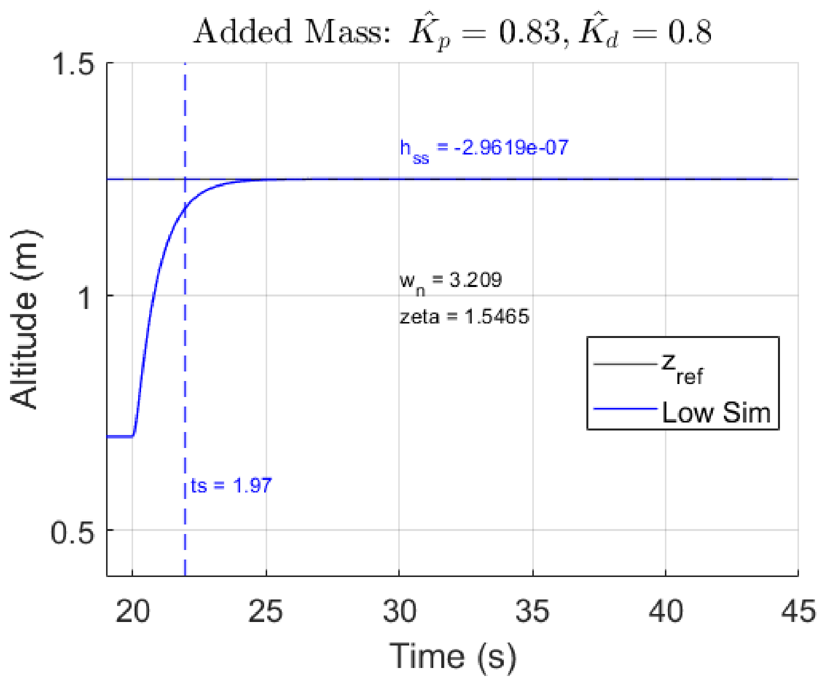
```
% Paths for data loading
ROOT_DIR = "C:/Users/Trevor/Desktop/AEM 4602W/Controls-Lab/Data/Part 3/";
PREFIX = "pjsdata_P3T";
DISP_NAME = "Added Mass";
LABEL_NAME = "Run";
N_TESTS = 2;
N_RUNS = 2;
masses = [79.3, 80.6] / 1000; % kg
[avg_hss, var_hss, avg_kt, var_kt, kphat, kdhat, kp, kd] = make_graphs(ROOT_DIR, PREFIX, DISP_NAME, LABEL_NAME, N_TESTS, N_RUNS, masses)

for test_n=1:N_TESTS
    [wn, zeta] = Low_Fidelity_Model(DISP_NAME, masses(test_n), avg_kt(test_n), kp(test_n), kd(test_n));
end
```

```
avg_hss =

   -0.1487   -0.1418


var_hss =

   1.0e-03 *

    0.1446    0.6192


avg_kt =

    0.5450    0.5636


var_kt =

   1.0e-03 *

    0.0114    0.1419


kphat =

    0.9900    0.8300


kdhat =

    0.5000    0.8000


kp =

    0.4541    0.3682


kd =

    0.2294    0.3549
```


Added Mass: $\hat{K_p} = 0.99, \hat{K_d} = 0.5$
Altitude (m)
Time (s)
h_ss = -0.14025
h_ss = -0.15725
ts = 1.48
ts = 0.945
z_ref
Run 1
Run 2


Added Mass: $\hat{K_p} = 0.83, \hat{K_d} = 0.8$
Altitude (m)
Time (s)
h_ss = -0.1242
h_ss = -0.15939
ts = 2.61
ts = 2.175
z_ref
Run 1
Run 2





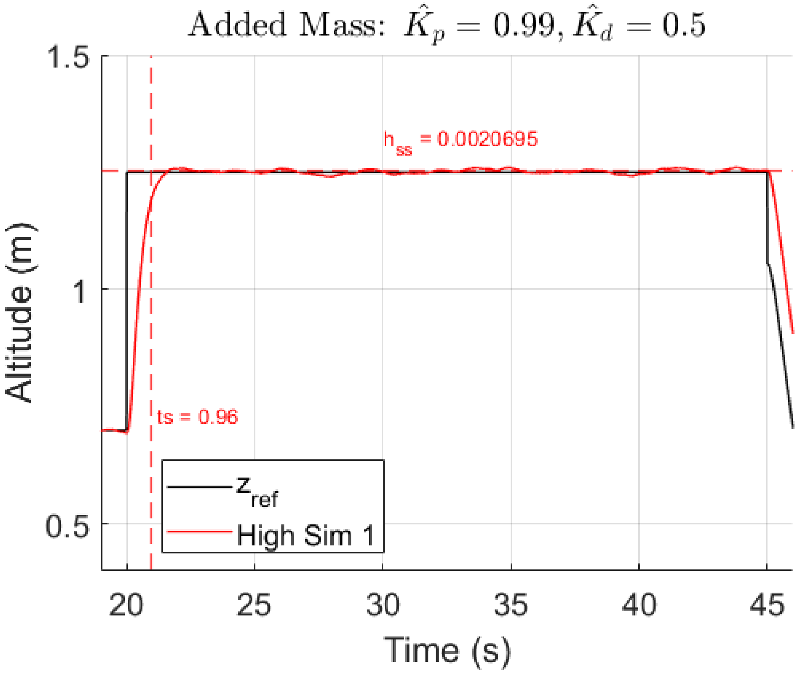
Published with MATLAB® R2020b

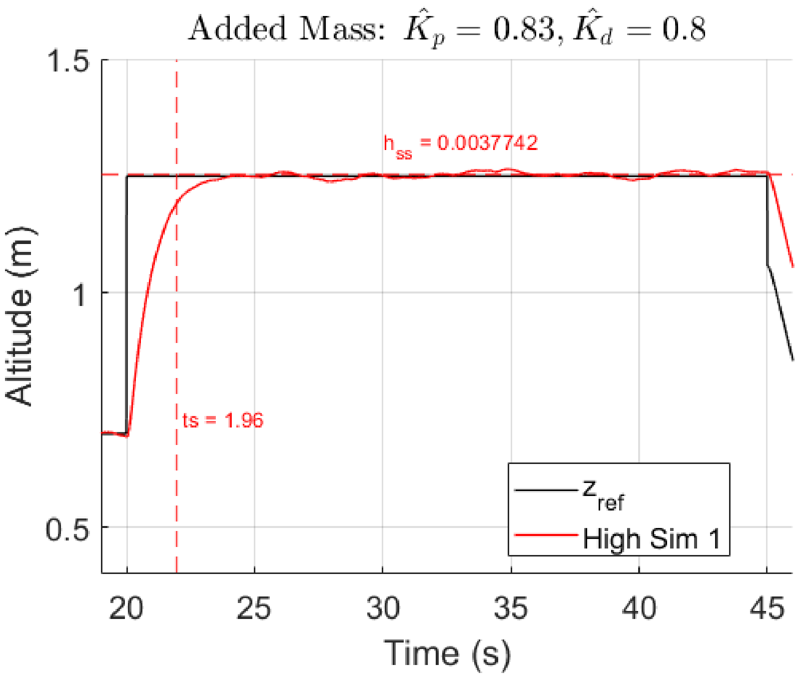
High-Fidelity, Added Mass Sims

Trevor Burgoyne 13 Nov 2022

```
% Paths for data loading
ROOT_DIR = "C:/Users/Trevor/Desktop/AEM 4602W/Controls-Lab/Data/Part 3/";
PREFIX = "part3_t";
DISP_NAME = "Added Mass";
LABEL_NAME = "High Sim";
N_TESTS = 2;
N_RUNS = 1;
masses = [79.3, 80.6] / 1000; % kg
[avg_hss, var_hss, avg_kt, var_kt, kphat, kdhat, kp, kd] = make_graphs(ROOT_DIR, PREFIX, DISP_NAME, LABEL_NAME, N_TESTS, N_RUNS, masses)
```

```
avg_hss =

    0.0021    0.0038


var_hss =

     0     0


avg_kt =

    0.6585    0.6660


var_kt =

     0     0


kphat =

    0.9900    0.8300


kdhat =

    0.5000    0.8000


kp =

    0.3758    0.3116


kd =

    0.1898    0.3003
```





Published with MATLAB® R2020b

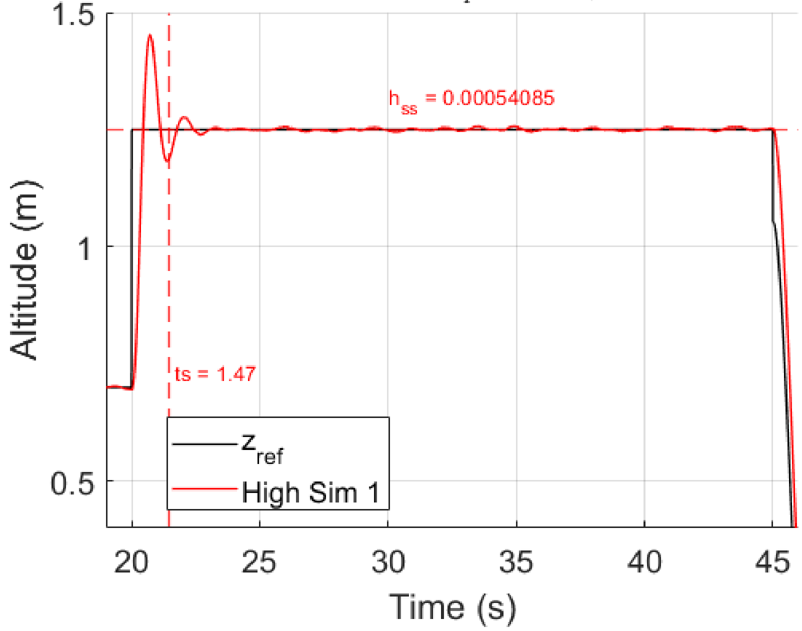
High-Fidelity, No Mass Sims

Trevor Burgoyne 13 Nov 2022

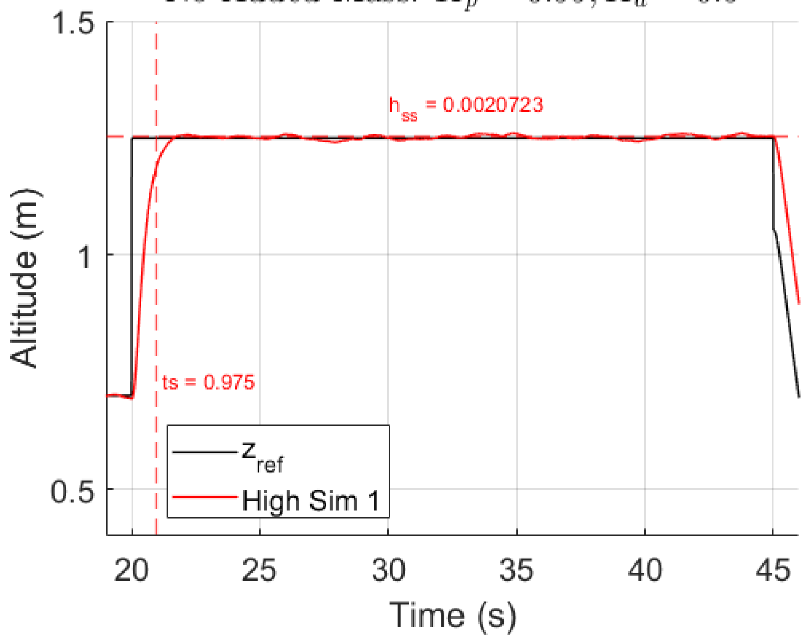
```
% Paths for data loading
ROOT_DIR = "C:/Users/Trevor/Desktop/AEM 4602W/Controls-Lab/Data/Part 2/";
PREFIX = "sim_t";
DISP_NAME = "No Added Mass";
LABEL_NAME = "High Sim";
N_TESTS = 6;
N_RUNS = 1;
masses = 69*ones(1,N_TESTS)/1000; % kg
[avg_hss, var_hss, avg_kt, var_kt, kphat, kdhat, kp, kd] = make_graphs(ROOT_DIR, PREFIX, DISP_NAME, LABEL_NAME, N_TESTS, N_RUNS, masses)
```

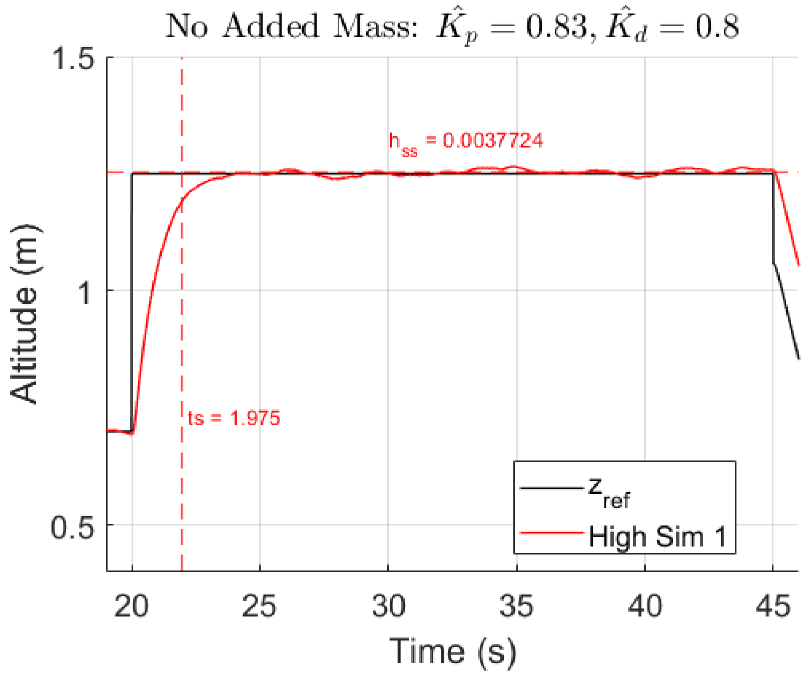
```
avg_hss =

    0.0005    0.0021    0.0038    0.0017    0.0062    0.0042


var_hss =

     0     0     0     0     0     0


avg_kt =

    0.6815    0.6669    0.6489    0.6615    0.6534    0.6536


var_kt =

     0     0     0     0     0     0


kphat =

    1.4300    0.9900    0.8300    1.6900    0.5000    0.1000


kdhat =

    0.2000    0.5000    0.8000    0.6900    0.9900    0.1000


kp =

    0.5245    0.3711    0.3198    0.6387    0.1913    0.0383


kd =

    0.0734    0.1874    0.3082    0.2608    0.3788    0.0383
```

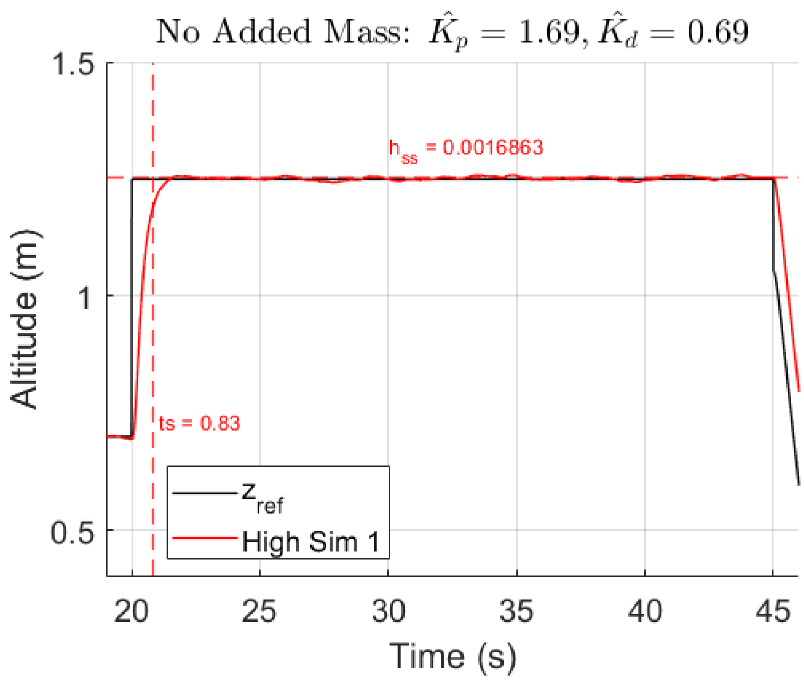
No Added Mass: $\hat{K_p} = 1.43, \hat{K_d} = 0.2$

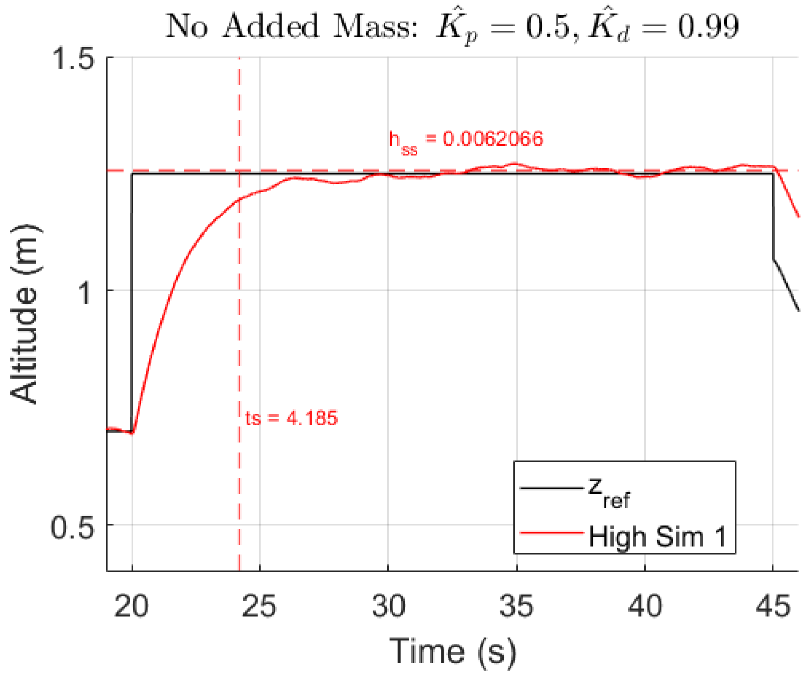


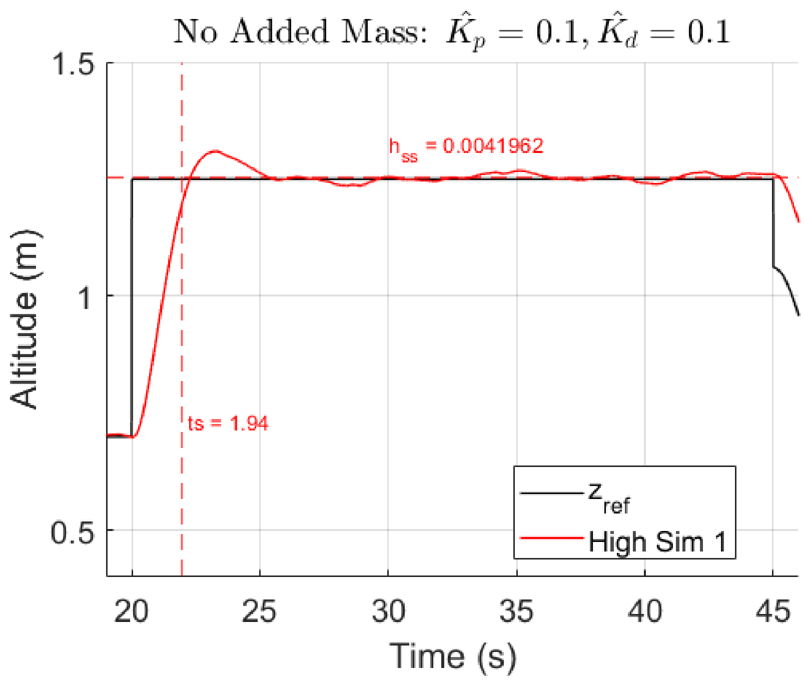
No Added Mass: $\hat{K_p} = 0.99, \hat{K_d} = 0.5$




No Added Mass: $\hat{K}_p = 0.83, \hat{K}_d = 0.8$
h_{ss} = 0.0037724
ts = 1.975
z_{ref}
High Sim 1
Altitude (m)
Time (s)
1.5
1
0.5
20
25
30
35
40
45


No Added Mass: $\hat{K}_p = 1.69, \hat{K}_d = 0.69$
h_{ss} = 0.0016863
ts = 0.83
z_{ref}
High Sim 1
Altitude (m)
Time (s)
1.5
1
0.5
20
25
30
35
40
45





Published with MATLAB® R2020b