Actuator Tuning Results Document

Actuator Tuning Results Document:

Publication date 21-May-2019 09:51:03

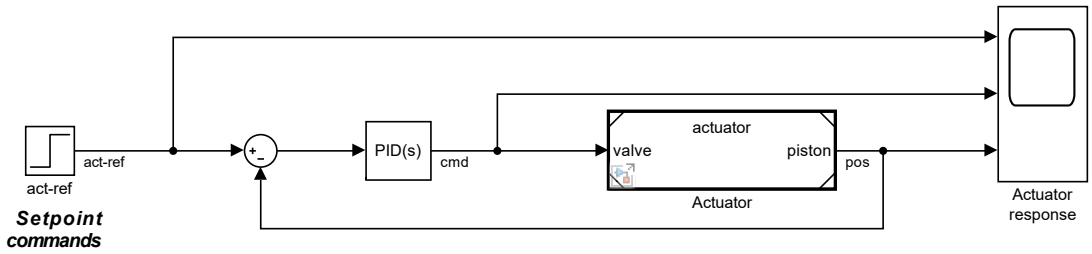
Abstract

Hydraulic actuator tuning example

Table of Contents

Chapter 1. Actuator Loop Model

The following model contains the Actuator Control system and plant model used for tuning the gains. This is a continuous time model that will be linearized as part of the tuning process.



Chapter 2. Code: Act_tuning_script

MATLAB Tuning Code. The following code uses Simulink Control Design to tune the actuator loop gains for the helicopter control system. The actuator loop control is a lead-lag filter.

```
% Hydraulic actuator tuning example

% Copyright 2019 The MathWorks, Inc.

% first, test for a Simulink Control Design, Control System Toolbox
 and
% Robust Control Toolbox licenses
if license('test','simulink_control_design')&&...

 license('test','control_toolbox')&&license('test','robust_toolbox')

    % bring command window to the front
    commandwindow;

    ST0 = slTuner('act_loop_Sdomain','PID Controller');
    ST0.OperatingPoints = 0.2; %set time point such that valve is open
    addPoint(ST0,'act-ref');   % setpoint commands
    addPoint(ST0,'pos');       % corresponding outputs
    addPoint(ST0,{'cmd','pos'});

    % Less than 5% mismatch with reference model 1/(s+1)
    TrackReq = TuningGoal.StepTracking('act-ref','pos',0.0159,10);
    TrackReq.RelGap = 0.05;

    % Gain and phase margins at plant inputs and outputs
    MarginReq1 = TuningGoal.Margins('cmd',5,40);
    MarginReq2 = TuningGoal.Margins('pos',5,40);

    % Limit on decay, damping, frequency
    PoleReq = TuningGoal.Poles(0,0,inf);

    AllReqs = [TrackReq,MarginReq1,MarginReq2,PoleReq];
    [ST1,fSoft,~,Info] = systune(ST0,AllReqs);

    T1 = getIOTransfer(ST1,'act-ref','pos');
    opt = stepDataOptions('StepAmplitude',0.02);
    step(T1,1,opt);

    figure('Position',[100,100,900,474]);
    viewGoal(AllReqs,ST1);

    showTunable(ST1);

    % Clean up
    bdclose('all');
    clearvars;
```

```
else % open the pre-existing report and post a warning
    % Get the ProjectManager.
    prj = simulinkproject;
    prjRoot = prj.RootFolder;

 open(fullfile(prjRoot, 'ARP_03_SystemArchitecture', 'validation', 'Act-
Tuning-Report.pdf'));
    warndlg(['There is a license missing for Simulink Control Design
 '...
        'or Control System Toolbox or Robust Control Toolbox, a ' ...
        'pre-existing tuning report has been opened'],...
        'Missing Product License');
end
```

Chapter 3. Command Window Output

Gain Results From Tuning Script. When Act_tuning_script.m is run in the workspace, the following results are displayed to the command window for the actuator loop transfer function:

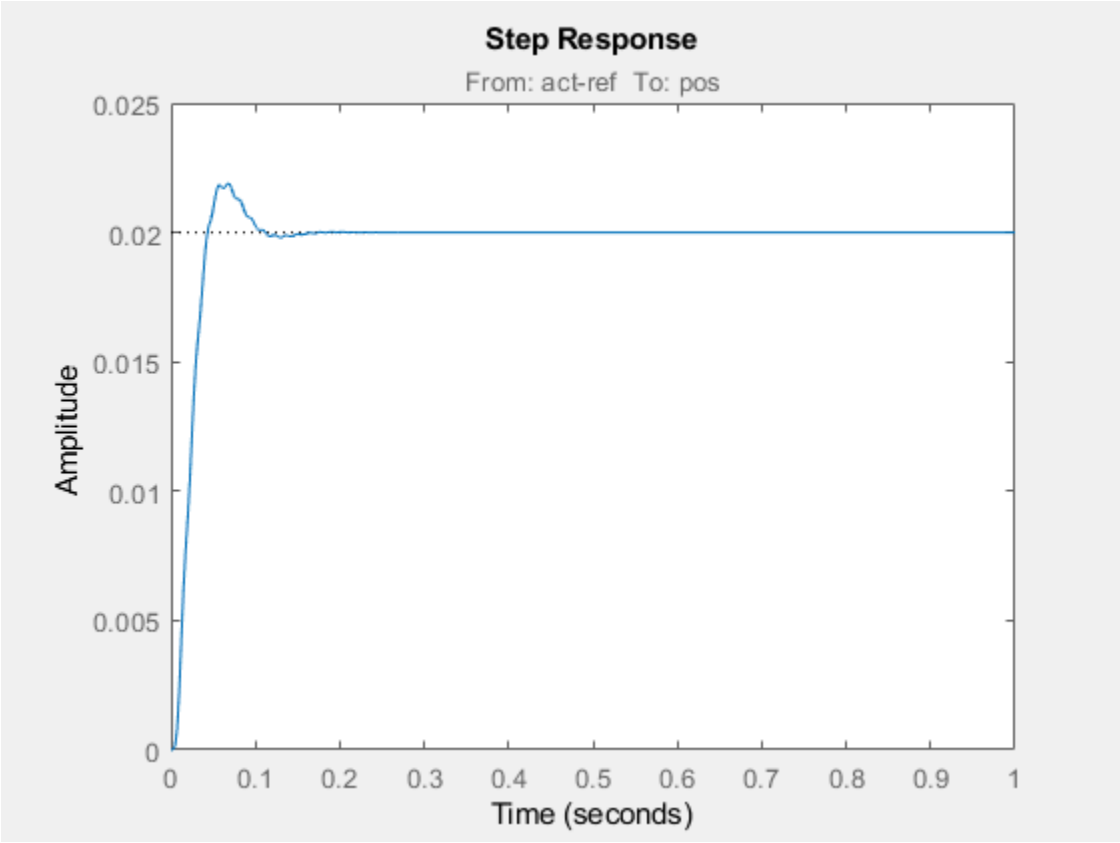
```
Final: Soft = 0.707, Hard = -Inf, Iterations = 46

Block 1: <a href="matlab:hilite_system('act_loop_Sdomain/PID
 Controller','find');pause(1);hilite_system('act_loop_Sdomain/PID
 Controller','none');">act_loop_Sdomain/PID Controller</a> =

             1            s
  Kp + Ki * --- + Kd * --------
             s          Tf*s+1

  with Kp = 0.135, Ki = 0.00122, Kd = -0.00134, Tf = 0.0157

Name: PID_Controller
Continuous-time PIDF controller in parallel form.
```

Chapter 4. Results

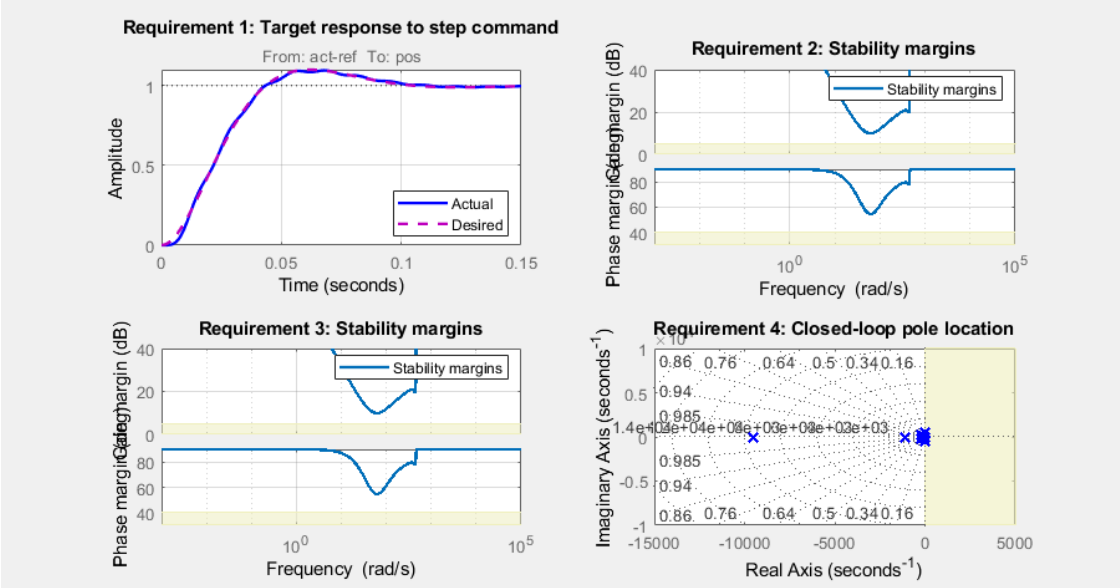
Table of Contents

Control System Responses and Stability Margins. Two figures are provided here, one that shows the step responses, and the other that shows the target responses, stability margins and closed loop pole locations. These responses are based on the selected transfer function. Running Act_tuning_script.m creates 2 figure(s). A snapshot of each is displayed below.

4.1. Figure



4.2. Figure


Requirement 1: Target response to step command
From: act-ref To: pos
Amplitude
Time (seconds)
Actual
Desired
Requirement 2: Stability margins
Stability margins
Phase margin (deg)
Gain margin (dB)
Frequency (rad/s)
Requirement 3: Stability margins
Stability margins
Phase margin (deg)
Gain margin (dB)
Frequency (rad/s)
Requirement 4: Closed-loop pole location
Imaginary Axis (seconds^-1)
Real Axis (seconds^-1)