```
clear; close all; clc
fig = uifigure;
d = uiprogressdlg(fig, "Title", "Processing", "Message", "Performing
 Calculations", "Cancelable","on","Indeterminate","on");
drawnow
conversion = 273.15;
temp_outside = 25 + conversion;
temp_desired = 100 + conversion;
steps = 5000;

c_a = 1.0035;       % J/K, (c_a) thermal_mass_air
c_s = 1.1;          % J/K, (c_s) thermal_mass_structure. Sheet metal:
 approx 0.1 - 0.2 for a few mm thickness
u_s = 0.015;        % W/(m*K), (u_s)
 conductance_between_air_and_structure
u_a = 0.01;         % W/(m*K), (u_a) conductance_structure_envelope
q_i = 0;            % heat flux due to internal heat sources (q_i)
q_h = 1;            % heat flux due to solar radiation (q_h)
q_a = q_i + q_h;    % total heat flux for the air (q_a)
q_s = 1;            % heat flux to the interior solid mass (q_s)

control_vector = [q_a; q_s; q_h];
A = [-(u_a + u_s)/c_a, u_s/c_a, u_a/c_a;...
    u_s/c_s, -u_s/c_s, 0;...
    0, 0, -1];
B = [1/c_a, 0, 0;...
    0, 1/c_s, 0;...
    0, 0, 1/c_s];
C = eye(3);
% D is zero because there is no feedthrough
D = 0*eye(3);

% Define time vector
t = linspace(0, steps / 10, steps + 1);
delta_t = t(2) - t(1);

% Define frequency domain variable, s
s = tf('s');

% Initial conditions
initial_conditions = [273.15, 283.15];
d.Message = "Creating state-space object";
drawnow
% Create state-space model object
model_state_space = ss(A,B,C,D)
d.Message = "Tuning Closed-loop System";
drawnow
% Prepare a tunable controller to connect to above plant model
decoupler = tunableGain('gain_block', eye(3));
decoupler.InputName = 'e';
decoupler.OutputName = 'p';
```

```
PI_1 = tunablePID('PI_1', 'pi');
PI_1.InputName = 'p(1)';
PI_1.OutputName = 'q(1)';
% PI_1.Kp.Value = 0.0146;
% PI_1.Ki.Value = 0.000244;

PI_2 = tunablePID('PI_2', 'pi');
PI_2.InputName = 'p(2)';
PI_2.OutputName = 'q(2)';

PI_3 = tunablePID('PI_3', 'pi');
PI_3.InputName = 'p(3)';
PI_3.OutputName = 'q(3)';
G = idtf(model_state_space,...
    'InputName', 'q', ...
    'OutputName', 'y');
% Look for poles with positive real parts
if ~isempty(find(pole(G) > 0, 1))
    disp('Unstable: Poles in the right hand plane.')
    disp(pole(G))
    close(d)
    delete(fig)
    return
end

% Create a sum block to match the desired feedback architecture
sum_block = sumblk('e = r - y', 3);
% Connect all the components of the controller
C0 = connect(PI_1, PI_2, PI_3, decoupler, sum_block, {'r','y'}, 'q');

% target_step_response = 500;
% requirement = TuningGoal.StepResp({'r(1)'}, {'y(1)'},
 target_step_response);
% reference_system = tf(requirement.ReferenceModel);

% Select the number of optimization runs to conduct
runs = 6;
options = looptuneOptions('RandomStart', runs, 'UseParallel', true);
loop_bandwidth_range = [0.1, 100];
max_error_percent = 0.1;
requirement = TuningGoal.Tracking('r(1)', 'y(1)', max_error_percent);
[G, C, gam, info] = looptune(G, C0, loop_bandwidth_range, requirement,
 options);
% disp('Peak Gain < 1 indicates looptune was able to find parameter
 values that achieve the target loop bandwidth.')

if gam <= 1
    disp('All tuning requirements satisfied.')
else
    disp('At least 1 requirement is unmet. Try increasing target
 bandwidth or relax another tuning requirement.')
    close(d)
    delete(fig)
    return
```

```
end
showTunable(C); % show the tuned controller parameters
% Connect the controller and plant model in a feedback loop
T = connect(G,C,'r','y');

% Provide initial conditions
step_options = stepDataOptions('InputOffset',
 initial_conditions(1),...
    'StepAmplitude', temp_desired-initial_conditions(1));

% Plot the step response
step_fig = figure;
step(T, step_options)
grid on
% Save simulation results

try
    working_folder = pwd;
    dir_info = dir;
    make_folder = true;
    for i = 1:length(dir_info)
        if strcmpi(dir_info(i).name, 'arda_sim_results')
            make_folder = false;
            break
        end
    end

    if make_folder
        mkdir('arda_sim_results\')
    end

    % cd(fullfile(pwd, 'arda_sim_results'))
    datetime_str = datestr(now,'yyyy-mm-dd_HH-MM-SS');

    variables_filename = sprintf('arda_sim_results_%s', datetime_str);
    save(fullfile(working_folder, 'arda_sim_results',
 variables_filename), '-v7.3', '-nocompression');

    figure_filename = sprintf('step_response_%s.fig', datetime_str);
    savefig(fullfile(working_folder, 'arda_sim_results',
 figure_filename))
    % cd(working_folder)
catch
    error('Unable to save simulation results.')
end
% Save pdf and latex of .mlx
export_options = struct('format', 'pdf', 'outputDir',
 fullfile(working_folder, 'arda_sim_results'),...
 'figureSnapMethod', 'print', 'evalCode', false, 'showCode', false);
publish("arda_sim.mlx", export_options)
% Close / Delete the progress dialog
close(d)
delete(fig)
```

Published with MATLAB® R2021a