Multiple-Input Multiple-Output Control

Master of Science in Aerospace Engineering

The Flying Chardonnay Final Challenge

Author:

Rajashree Srikanth rajashree.srikanth@student.isae-supaero.fr

Tutor:
Leandro Lustosa

January 31, 2024

2023/2024 – 3rd Semester

1 Objective

The goal of this assignment is to design a controller that can allow the flying chardonnay to follow the trajectory represented in figure 1 with a final percentage of drink larger than 65%.

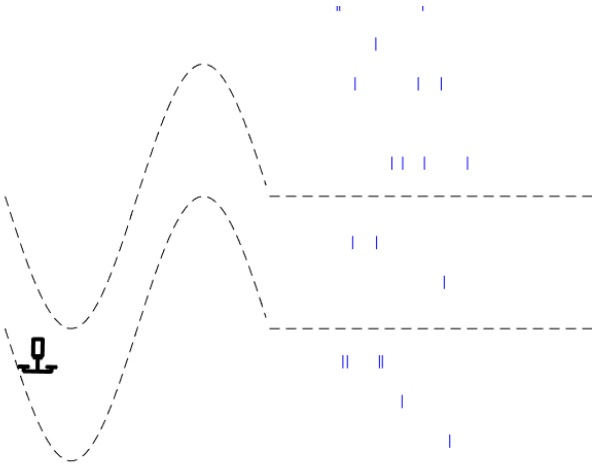

Figure 1: Trajectory

The mechanisms in which the drink might spill are:

- Inclination of cup spills drink;
- There is a drink leakage rate of 1.5% per second;
- Out-of-bounds violation empties the cup.

Additionally, the controller should be tolerant to noise and constant wind disturbances that have been modelled in the simulator. Finally, the controller should also be robust against parameter uncertainty in $\pm 5\%$ on m_d, m_c, l, l_d and J.

For the challenge given, eigenstructure assignment technique is utilized for the controller design, under a non-linear optimization framework. Following this, robustness of the system is studied by plotting σ-plots and disk margin and introducing upper and lower bounds on parameter uncertainties in the model.

2 Design of Controller

2.1 Initial Tuning of Controller

The code provided had a pre-tuned controller gain, which was capable of retaining 56.88% of the drink in the cup. From an initial observation, we can see that the chardonnay executes very large deflections in γ and θ, and the recovery from such large deflections is slow. Thus, modifying the dynamics relating to these parameters could help improve the performance.

Further, this static controller is then manually tuned through trial and error. This exercise was found to be tedious, and it was difficult to find a logic or reasoning on the tuning process, more so since the static controller gain is of size $2X8$. Thus, an alternative approach for the choice of control design is pursued.

2.2 Eigenstructure Assignment

Eigenstructure Assignment shares a logic similar to pole placement. In this technique, we can assign pole locations as eigenvalues to the system and fix eigenvector directions along which these individual eigenvalues can act on. A major advantage to this method is the flexibility in choosing the poles, thereby give large leeway in specifying the system performance.

For this system containing 8 states, we choose 8 poles that will enable us to achieve the desired performance, projected along 8 mutually perpendicular/independent directions (or basis). For convenience, we project these eigenvalues along unit-axis directions on a vector space of dimension 8. The columns of an $8X8$ identity matrix (I_8) form the basis or the eigenvectors for this technique.

$$I_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

This method requires state matrix A and input-to-state matrix B. Thus, we need to linearize this model about a certain trim point. For this reason, we consider trim points about hover condition. We take trim state (x_{trim}), input (u_{trim}) and wind (w_{trim}) as:

$$x_{trim} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$u_{trim} = \begin{bmatrix} \frac{W}{2} & \frac{W}{2} \end{bmatrix}$$

$$w_{trim} = \begin{bmatrix} 0 & 2 \end{bmatrix}$$

We can linearize the non-linear dynamics of the flying chardonnay about the above trim point to obtain matrices A, B. We want to use full-state feedback, and the state-to-output matrix C is taken as I_8. Feedthrough matrix D is taken as 0_{8X8}. The linearized model is obtained as follows:

```
x_trim = [0 0 0 0 0 0 0 0];
W = (chardonnay.m_c + chardonnay.m_d)*chardonnay.g;
u_trim = [W/2 W/2];
w_trim = [0 2];

% Obtaining linearized model about trim point
delta = 1e-3;
A = zeros(n,n);
B = zeros(n,2);

for k = 1:n
    dx = zeros(1,n);
    dx(k) =  1i*delta;
    a = chardonnay_dynamics(x_trim+dx,u_trim,w_trim,chardonnay)
    A(:,k) = imag(a)/delta
end
for k = 1:2
    du = zeros(1,2);
    du(k) =  1i*delta;
    b = chardonnay_dynamics(x_trim,u_trim+du,w_trim,chardonnay)
    B(:,k) = imag(b)/delta;
```

where the function *chardonnay_dynamics* contains the dynamics of the non-linear plant.

Assuming a certain required set of eigenvalues p, the code snippet used to compute the respective controller gain K is as follows:

```
n = 8 % no of. states
% x = (pn, pd, vn, vd, the, thed, gam, gamd)
for i = 1:n
    % compute S matrix for pole
    Sm1 = [(A-p(i)*eye(n)) -B];
    % compute nullspace for S
    Ke = null(Sm1);
    % Compute linear combination of vectors of nullspace that aligns with
    % desired basis
    VM = Ke*(Ke(1:n,1:2)\basis(:,i));
    v1 = VM(1:n);
    m1 = VM(9:end);
    V(:,i) = v1;
    M (:,i) = m1;
end
K = (M*inv(V))
```

2.2.1 Choice of Eigenvalues

A starting point for choosing poles is based on the eigenvalues for the open-loop linearized plant, obtained using the MATLAB function *eig(A)*. We get:

$$poles = [0,\ 4.4475, -4.4975,\ -0.0500,\ 0,\ 0,\ 0,\ -0.0500]$$

We can observe that the system has one unstable pole and 3 poles on the imaginary axis. First, we try to place poles that can stabilize the system by correcting for these system poles. Following this, we attempted manual pole placement through trial and error to understand their coupling with the 8 states of the system. It is observed that the poles corresponding to vertical displacement and velocity, p_d and v_d are uniquely coupled together, and their values do not affect the performance of the other set 6 states, and vice-versa for the latter set of states as well.

Additionally, we would like the dynamics of $\theta_d, \gamma, \gamma_d$ to be fast enough to recover to the trimmed hover state, so that the controller obtained for linearized dynamics can still be valid, and to ensure that minimal amount of drink is spilled from the cup.

Keeping these considerations in mind, we get an initial estimate of the location of poles as:

$$poles = [-15.2,\ -5.4475,\ -4.4975,\ -0.5,\ -0.5,\ -2.7,\ -4.0,\ -3.5]$$

However, it was difficult to achieve drink retention higher than 65.32%, thus necessitating an optimzation algorithm to improve performance.

2.3 Optimization Algorithm

An optimization algorithm that uses the deviation of water level about 100% as the cost function. We modify the *chardonnay_animate.m* code provided, that takes pole location as input to compute the associated water level remaining after end of simulation. The pseudo-code is provided below:

```
    function cost = level(x)
        %% chardonnay parameters
        chardonnay.m_d = 1;
        chardonnay.m_c = 1;
        chardonnay.l = 1;
```

```
        chardonnay.l_d = 1;
        chardonnay.J = 0.1;
        chardonnay.C_D = 0.1;
        chardonnay.g = 10

        %% linearizing model about trim point
        %% creating eigenstructure based on pole location passed as function input
        %% computing associated controller gain K
        .
        .
        .

        %% obtaining water level
        modelname = 'chardonnay_simulinkR2018a';
        simIn = Simulink.SimulationInput(modelname);
        simIn = setVariable(simIn,'K',K);
        out = sim(simIn);
        t = out.simout.Time;
        r = out.simout.Data(:,1:2)';
        r(2,:) = -r(2,:);
        theta = out.simout.Data(:,5)';
        gamma = out.simout.Data(:,7)';

        %% code from animate_chardonnay.m to calculate water level
        .
        .
        .
        cost = (100-water_level)
    end
```

The above function is passed in as the cost function to minimized to the in-built MATLAB non-linear optimizer *fmincon*, taking the pole location obtained in previous section as the initial estimate.

```
    fun = @(x) level(x);
    x0 = [-15.2   -5.4475    -4.4975    -0.500   -0.5    -2.7   -4.0    -3.5]
    options= optimset('algorithm','active-set','Display','Iter','TolX',1e-4,'TolCon',
    1e-4, 'TolFun',1e-4, 'PlotFcn','optimplotfunccount');

    [x, fval,ExitFlag, Output] = fmincon(fun, x0, [],[], [], [], [], zeros(1,8))
```

On running the optimizer above, we obtain the final set of eigenvalues as:

$$poles = [-15.2382,\ -5.5377,\ -4.3416,\ -0.6719,\ -0.0019,\ -2.4971,\ -3.7963,\ -3.5184]$$

.

These new set of values yield in drink retention of 73.2%, a 12.2% increase in performance.

2.4 Altering Reference Trajectory

To further improve performance, we attempt to modify the reference trajectory to:

```
    function x_ref = fcn(t)
    % x = (pn, pd, vn, vd, the, thed, gam, gamd)

    x_ref = zeros(8,1);

    TIME_CTE = 10;

    if (t<TIME_CTE)
```

```
        x_ref(1) = 1*t/TIME_CTE*20;
        x_ref(2) = 9*sin( x_ref(1)/20*2*pi );
        x_ref(3) = 1/TIME_CTE*20;
        x_ref(4) = 13/20*2*pi*cos( x_ref(1)/20*2*pi );
    elseif (t<2*TIME_CTE)
        x_ref(1) = 20+(t-TIME_CTE)/TIME_CTE*20;
        x_ref(3) = 4.5/TIME_CTE*20;
    else
        x_ref(1) = 40;
        x_ref(3) = 0;
    end
```

By essentially smoothening the position trajectory and increasing speed, we observe a *final drink retention of* ***78.3%***.

3 Analysis of Controller Performance

The controller tuning process provided insight into the system performance. From an initial observation, it can be concluded that larger the speed of the chardonnay, higher the percentage of drink left in the cup. Faster dynamics result in a higher risk of straying outside the trajectory bounds. Nevertheless, this does not alter the robustness of the system to uncertainties in the model and external disturbances/noise. Thus, for a good drink retention metric, we require a combination of a robust controller and a reasonable reference trajectory.

3.1 Measure of Robustness

In figure 2, we can observe a large variation in the value of σ, which implies that the system is less sensitive to perturbations. Figure 3 shows the phase and gain margins for the MIMO system. The phase margin PM $= [-4.95, 4.95]$ and the gain margin GM $= [0.92, 1.09]$. We can see that the MIMO robust margins are very low, and does not allow for very large uncertainties in the system. The figure 4 represents the σ-plot of the sensitivity transfer matrix. We can see that at high frequencies (or for transient time response), the magnitude of the sensitivity function equals 1, which implies good noise rejection.

3.1.1 Effect of Model Uncertainties

We introduce uncertainties on $\pm 5\%$ on the mass of the drone (m_d) and the cup (m_c), the lengths (l, l_d) and the inertia (J) to study how robust the controller is in effect of these variations. We realize this consideration by analysing the drink level metric and robustness margins for the upper and lower bounds. The results of this study are tabulated below in table 1.

Uncertainty	Water Level	Phase Margin	Gain Margin
0 % (Nominal case)	78.3 %	$[-4.95, 4.95]$	$[0.92, 1.09]$
$+5\%$ on all parameters	86.25%	$[-4.8, 4.8]$	$[0.92, 1.08]$
-5% on all parameters	70.63%	$[-5.14, 5.14]$	$[0.91, 1.09]$

Table 1: Effect of Model Uncertainties

From the table above, it can be inferred that a higher percentage of drink comes at the cost of a lower phase margin and vice-versa, although the difference between these is marginal. However, there does not seem to be any effect in the gain margins of the system due to these uncertainties. From the

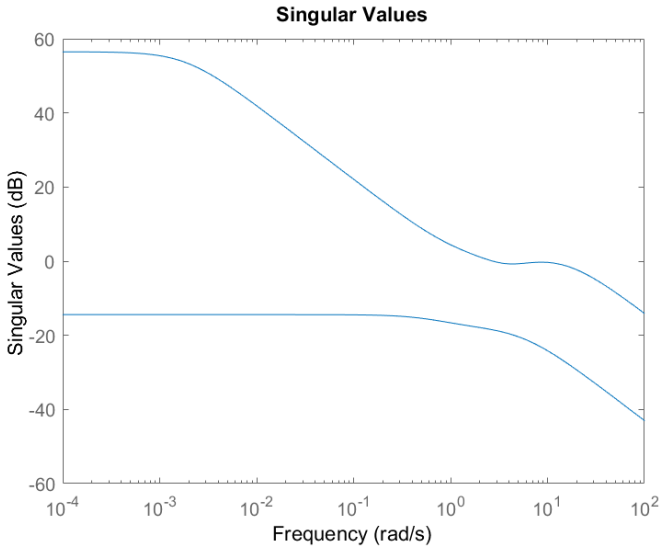

Figure 2: σ-plot

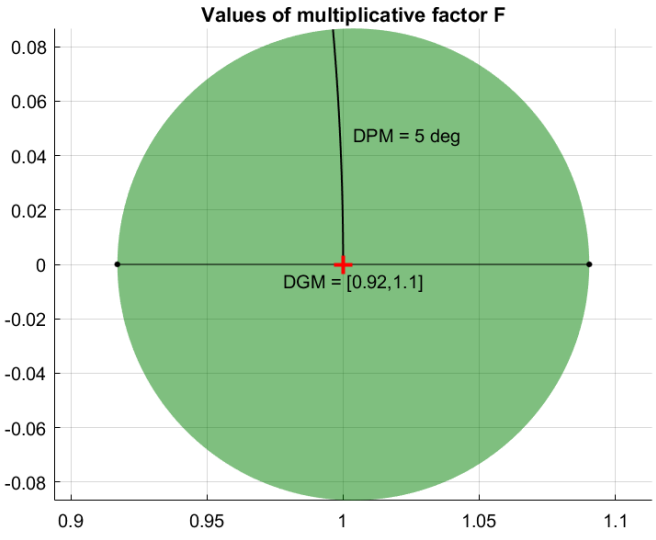

Figure 3: Disk Margin

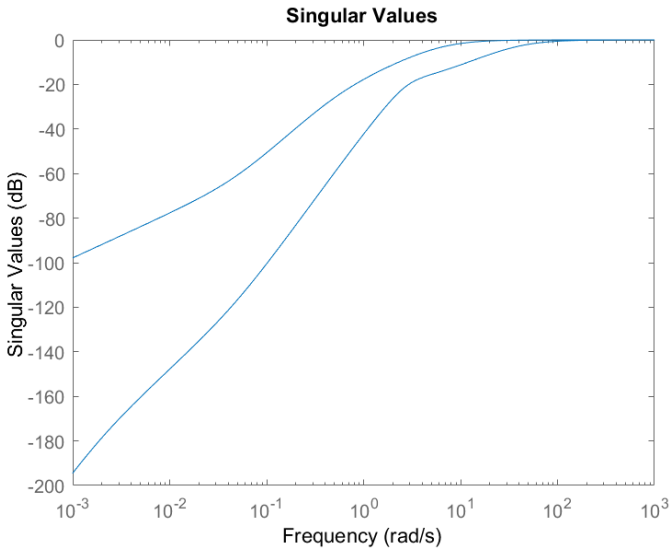

Figure 4: σ-plot of Sensitivity Transfer Matrix

above study, we can conclude that the controller is able to meet the robustness requirement due to parameter model uncertainties.