Robust and Optimal Control

Master of Science in Aerospace Engineering

Robustness Analysis of a Spark Ignition Engine

Authors:

Rajashree Srikanth rajashree.srikanth@student.isae-supaero.fr
Muhamad Abdul Aziz Muhamad-abdul.AZIZ@student.isae-supaero.fr

Tutor:
Clément Roos

February 2, 2024

2023/2024 – 3rd Semester

1 Modeling

A spark ignition engine model which has been linearised, is presented in a block diagram in Figure 1. To control the system, a feedback controller which consists of a proportional and integral controller is implemented as shown in Figure 2.

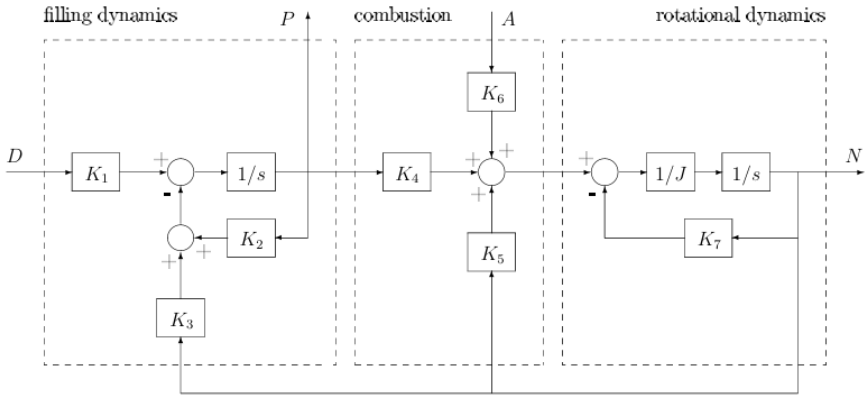

Figure 1: The block diagram of the open-loop engine model.

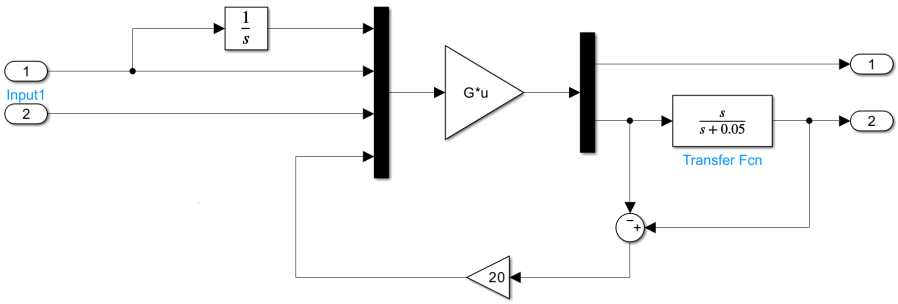

Figure 2: The block diagram of the controller.

1.1 Open-loop parametric model

The dynamic equation of the open-loop engine model was computed based on the block diagram given in Figure 1. Then, equation 1 can be arranged in the state-space representation by taking $x = \begin{bmatrix} N & P \end{bmatrix}^T$ as the state vector, $u = \begin{bmatrix} D & A \end{bmatrix}^T$ as the input vector, and $y = x$ as the output vector.

$$
\begin{aligned}
\dot{N} &= \frac{1}{J}((K_5 - K_7)N + K_4 P + K_6 A) \\
\dot{P} &= -K_3 N - K_2 P + K_1 D
\end{aligned}
\tag{1}
$$

$$
\dot{x} = \begin{bmatrix} (K_5 - K_6)\frac{1}{J} & K_4\frac{1}{J} \\ -K_3 & -K_2 \end{bmatrix} x + \begin{bmatrix} 0 & K_6\frac{1}{J} \\ K_1 & 0 \end{bmatrix} u \tag{2}
$$

From equation 2, the corresponding expressions for $\delta_1, \dots, \delta_7$ are obtained as follow:

$$\begin{aligned}
\delta_1 &= K_1 \\
\delta_2 &= -K_2 \\
\delta_3 &= -K_3 \\
\delta_4 &= K_4 \\
\delta_5 &= K_5 - K_7 \\
\delta_6 &= K_6 \\
\delta_7 &= K_8
\end{aligned} \tag{3}$$

By using the gss function in Matlab, the parameters $\delta_1, \dots, \delta_7$ were defined as varying parameters $[\delta_1^-, \delta_1^+], \dots, [\delta_7^-, \delta_7^+]$ and the open-loop parametric model was built using the Matlab ss function. This function gss finds the varying parametrs such that:

$$\dot{x} = \begin{bmatrix} \delta_5 \delta 7 & \delta_4 \delta 7 \\ \delta_3 & \delta_2 \end{bmatrix} x + \begin{bmatrix} 0 & \delta_6 \delta_7 \\ \delta_1 & 0 \end{bmatrix} u \tag{4}$$

```
d1 = gss('delta1',[],[2.1608 3.4329]);
d2 = gss('delta2',[],[-0.1627 -0.1027]);
d3 = gss('delta3',[],[-0.1139 -0.0357]);
d4 = gss('delta4',[],[0.2539 0.5607]);
d5 = gss('delta5',[],[1.7993-2.0283 2.1999-1.8201]);
d6 = gss('delta6',[],[1.8078 3.8429]);
d7 = gss('delta7',[],[0.1000 1.000]);

A = [d5*d7 d4*d7;
     d3 d2];
B = [0 d6*d7;
    d1 0];
C = eye(2);
D = zeros(2,2);
olsys = ss(A,B,C,D);
```

1.2 Controller

The model of the controller was built using Simulink presented in Figure 2. Using the Matlab linmod function, the corresponding state-space representation can be obtained as follows:

$$\begin{aligned}
A &= \begin{bmatrix} 0 & 0 \\ 0.0187 & -0.0276 \end{bmatrix} & B &= \begin{bmatrix} 0 & 0 \\ 0.1826 & 0.0848 \end{bmatrix} \\
C &= \begin{bmatrix} 0.0081 & 0.1202 \\ 0.0187 & -0.0276 \end{bmatrix} & D &= \begin{bmatrix} 0.0872 & 0.1586 \\ 0.1826 & 0.0848 \end{bmatrix}
\end{aligned} \tag{5}$$

1.3 Closed-loop parametric model

The closed-loop parametric model can be obtained by closing the loop using the feedback function in Matlab:

```
controller = ss(Ac,Bc,Cc,Dc); % State-space rep. obtained from the linmod
function
clsys = feedback(olsys*controller,eye(2));
```

It was observed that the order of the LTI model ($M(s)$) in the LFR system, is 4, which is true since the model consists of an integrator, a filter, and a second-order system. Regarding the values of $q_1, \ldots, q_7$, it was observed that the size is equal to $[1 \times 1]$ for $q_1, \ldots, q_6$ and $[3 \times 3]$ for q_7. The result matches with the equation 2 since $q_1, \ldots, q_7$ represent the number of occurrences of each corresponding parameter in the state equation.

2 Stability Analysis

2.1 Closed-loop stability at the operating points

All 3 operating points of the engine system were checked by applying the function eval. Then, the three closed-loop systems were obtained and the stability was checked using the Matlab damp function. It is shown from the result that the three operating conditions are stable in the closed loop.

```
% Delta at the operating points
d1A = kA(1);d2A = -kA(2);d3A = -kA(3);d4A = kA(4);d5A = kA(5)-kA(7);d6A = kA(6); d7A = kA(8);
d1B = kB(1);d2B = -kB(2);d3B = -kB(3);d4B = kB(4);d5B = kB(5)-kB(7);d6B = kB(6); d7B = kB(8);
d1C = kC(1);d2C = -kC(2);d3C = -kC(3);d4C = kC(4);d5C = kC(5)-kC(7);d6C = kC(6); d7C = kC(8);

clsysA = eval(clsys, {'delta1','delta2','delta3','delta4','delta5','delta6','delta7'},{d1A, d2A, d3A, d4A, d5A, d6A, d7A});
clsysB = eval(clsys, {'delta1','delta2','delta3','delta4','delta5','delta6','delta7'},{d1B, d2B, d3B, d4B, d5B, d6B, d7B});
clsysC = eval(clsys, {'delta1','delta2','delta3','delta4','delta5','delta6','delta7'},{d1C, d2C, d3C, d4C, d5C, d6C, d7C});

%%%%%%%%%%%%%%%%%%%%%%%%

>> eig(clsysA)
ans =
    -0.3622 + 0.1989i
    -0.3622 - 0.1989i
    -0.1827 + 0.0000i
    -0.0960 + 0.0000i
>> eig(clsysB)
ans =
      -0.5470
      -0.3069
      -0.1403
      -0.0914
>> eig(clsysC)
ans =
    -0.3773 + 0.0000i
    -0.0380 + 0.0848i
    -0.0380 - 0.0848i
    -0.0919 + 0.0000i
```

2.2 Simulation of closed-loop system using random sampling

One way to check the system's robustness is by performing many simulations with randomized uncertain parameters and then checking whether there is a combination of parameters that could make the closed-loop unstable. The simulation was performed by using 1000 samples and the eigenvalues were saved. It can be seen that the eigenvalues plotted in Figure 3 do not give any poles which has a real value greater than zero, indicating that all of the simulated closed-loop systems are stable. It is also can be observed from the variable N_unstable which is null, showing that there is no unstable pole.

```
[sys_sim, sample] = dbsample(clsys,1000);
eigvals = [];
N_unstable = [];
for i=1:1000
    eigvals = [eigvals; eig(sys_sim{i})];
    if real(eig(sys_sim{i})) > 0
        N_unstable = [N_unstable i];
    end
end

%%%%%%%%%%%%%%%%%%%%%%

>> N_unstable
N_unstable =
     []
```

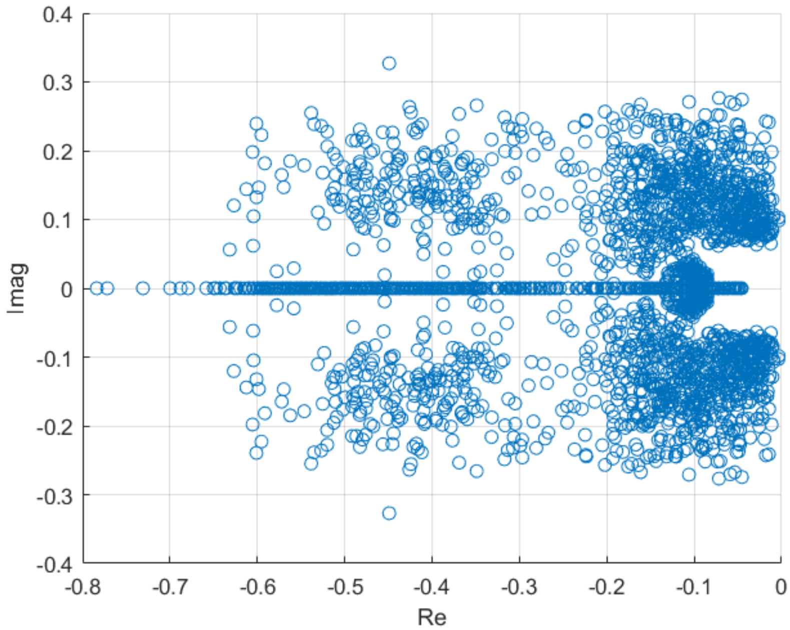

Figure 3: Simulation of the closed-loop system with sampled parameters. The simulated closed-loop systems are stable since the poles are in the left half plane of the complex domain.

2.3 μ-analysis

Another way to check the robustness of the closed-loop system is by performing μ analysis. The exogenous inputs and outputs are removed from the system since it is not used in the analysis, and the upper and lower bounds of μ are computed using the following code.

```
sys_wo_io = clsys([],[]); % remove exogenous inputs outputs
optub.lmi=1;optub.tol=0;
ubnd=muub(sys_wo_io,[],optub);
[lbnd,wc,pert,iodesc]=mulb(sys_wo_io);

```

```
%%%%%%%%%%%%%%%

>> lbnd
lbnd =
    1.0710
>> ubnd
ubnd =
    1.2436
```

Using the above method, the corresponding robustness margin, k_r, can be computed using the following relations:

$$\begin{aligned} \underline{\mu}_\Delta &= 1/\bar{k_r} \\ \bar{\mu}_\Delta &= 1/\underline{k_r} \end{aligned} \tag{6}$$

The obtained values are the following: $\bar{k_r} = 0.9337$ and $\underline{k_r} = 0.8041$. It can be concluded that the system is not robustly stable since $\bar{k_r} < 1$ or $\bar{\mu} > 1$. There is a combination of the parameters that could make the closed-loop unstable, this combination of parameter is saved in the iodesc variable. Using these parameters, the parametric closed-loop was then evaluated as follows:

```
clsys_unst = eval(clsys, {'delta1','delta2','delta3','delta4','delta5','delta6','delta7'}, {iodesc{1}.value, iodesc{2}.value, iodesc{3}.value, iodesc{4}.value, iodesc{5}.value, iodesc{6}.value, iodesc{7}.value});

%%%%%%%%%%%%%%%%%%
>> eig(clsys_unst)
ans =
  -0.3894 + 0.0000i
   0.0000 + 0.0730i
   0.0000 - 0.0730i
  -0.0900 + 0.0000i
```

2.4 Discussion

Both the randomized simulation and μ analysis are used to validate the robustness of a closed-loop system. Using the first method, it can be observed that all of the real values of the poles are in the left half plane, which means that the system is stable for all of the simulated closed-loop systems. However, the opposite result is obtained using the second method, in which the system is not robustly stable to the given uncertainty. This is because the first method relies on the randomization of the uncertain parameters hence it can miss the worst-case combination. The second method, however, does not have this drawback because it checks all of the admissible operating points.

3 Adjustment of Combustion Chamber

In this section, some combustion chamber parameters are modified. New values are identified for K_5 and K_6, and are tabulated below in Table 1.

Parameters	K_5	K_6
Point A	2.0183	4.4962
Point B	1.7993	2.0247
Point C	1.7993	4.4962

Table 1: New parameter values

3.1 Parametric model

For the above new parameters, the corresponding uncertainty parameter $\delta_1, ..., \delta_7$ are computed, taking into account their ranges of variation. The corresponding closed-loop parametric model $\sum_2(\delta)$ (or *sysCL_adjust* in MATLAB) is computed. This is shown in the code snippet below:

```
% K5, K6 are changed
d1=gss('delta1',[],[2.1608, 3.4329])
d2=gss('delta2',[],[-0.1627,-0.1027])
d3=gss('delta3',[],[-0.1139,-0.0357])
d4=gss('delta4',[], [0.2539, 0.5607])
d5=gss('delta5', [], [1.7993-2.0283, 2.0183-1.8201])
d6 = gss('delta6', [], [2.0247, 4.4962])
d7 = gss('delta7', [], [0.1, 1])

%
A = [d5*d7, d4*d7; d3, d2]
B = [0, d6*d7; d1, 0];
C = eye(2);
D = zeros(2,2)
sysOL = ss(A,B,C,D)

[Ac, Bc, Cc, Dc] = linmod('closed_loop_model')
sys_control_adjust = ss(Ac,Bc,Cc,Dc)
sysCL_adjust = feedback(sysOL, sys_control_adjust)
```

3.2 μ-analysis

The stability of the closed loop parametric model $\sum_2(\delta)$ is evaluated by investigating the upper and lower bounds of μ (or K_r).

```
sysCL_adjust_new = sysCL_adjust([],[]) %closed loop system without inputs and outputs

optub.lmi = 1;
optub.tol = 0;
ubnd = muub(sysCL_adjust_new, [], optub);
[lbnd, wc, pert, iodesc] = mulb(sysCL_adjust_new);
%% computing mu bounds
delta_lbnd = {iodesc{1}.value, iodesc{2}.value, iodesc{3}.value, iodesc{4}.value, iodesc{5}.value, iodesc{6}.value, iodesc{7}.value}
sysCL_lbnd = eval(sysCL_adjust_new, {'delta1', 'delta2', 'delta3', 'delta4', 'delta5', 'delta6', 'delta7'}, delta_lbnd)
k_lbnd_adjust = 1/ubnd
k_ubnd_adjst = 1/lbnd
```

As in the previous section, the upper and lower bounds of μ are computed by iteratively finding the set of uncertainties that pushes the system to the limit of stability, for a large range of frequency values.

$$\mu_\Delta = \frac{1}{k_r} = 0.89$$

$$\underline{\mu}_\Delta = \frac{1}{k_r} = 1.0808$$

The corresponding lower and upper bounds of the parametric domain $\mathbf{K}$ is computed, as is found to be:

$$\underline{k_r} = 0.9252$$

$$k_r = 1.2157$$

Since $K_2 \geq 1$ is true, we can conclude that the system is robustly stable, however, the lower bound $\underline{K_r}$ is not greater than 1, hence, it does not have a large pessimistic robustness margin. The upper bound for μ is 21% larger than the lower bound.

The pole location for the system corresponding to the limit of stability uncertainty is computed as follows:

```
>> eig(sysCL_lbnd)

    ans =
      -0.4398 + 0.0000i
       0.0000 + 0.0159i
       0.0000 - 0.0159i
      -0.0781 + 0.0000i
```

As expected, the system is neutrally stable at this configuration. Another observation is that adjusting the combustion chamber parameters accordingly allows for achieving system robustness.

4 Improvement of analysis results

While robustness was achieved, reducing the gap between the upper and lower bounds of μ (or improving the upper and lower bounds of k_r) reduces conservatism and serves as a measure of the proximity to the robustness margin. This goal can be achieved through two methods that are detailed in this section.

4.1 Applying branch-and-bound

The branch-and-bound method reduces the conservatism. It divides the system into subsystems and performs the μ analysis by reducing the gap between the upper and lower bounds. This allows to better estimate the robustness of the controller. A major drawback of using this technique is the associated exponential computation cost as a function of the number of real uncertainty parameters.

```
optbb.maxgap=0.05;
[lbnd,wc,pert,iodesc]=mubb(sysCL_adjust_new,[],optbb);

delta_lbnd = {iodesc{1}.value, iodesc{2}.value, iodesc{3}.value, iodesc{4}.value,
 iodesc{5}.value, iodesc{6}.value, iodesc{7}.value}
sysCL_lbnd = eval(sysCL, {'delta1', 'delta2', 'delta3', 'delta4', 'delta5', '
delta6', 'delta7'}, delta_lbnd)
k_adbndjust_br = 1./lbnd
```

The upper and lower bounds of μ and k_r are:

$$[\underline{\mu}_\Delta, \mu_\Delta] = [0.8182, 0.859]$$

$$[\underline{k_r}, k_r] = [1.1640, 1.2222]$$

The upper and lower bounds on μ are within a gap of 5%, proving the efficacy of this method to improve the system robustness. Upon inspection of the upper and lower bounds of k_r, it is evident that both conditions for robust stability: $\underline{k_r} \geq 1$ and $k_r \geq 1$ are satisfied. Thus, we can conclude that this method was able to successfully improve the robustness margin and reduce conservatism of the system.

4.2 Reducing the size of Δ

Yet another technique that can be employed to improve robustness properties is to reduce the size of the uncertainty matrix. This is performed by factorization to obtain a minimal realization of the system. Equation 4 contains repetition of the term δ_7, which is factorized to obtain the minimal realization of the system $(\sum_3(\delta))$ as follows:

```
A = [d7*d5, d7*d4; d3, d2]
B = [0, d7*d6; d1, 0];
C = eye(2);
D = zeros(2,2)
sysOL_min = ss(A,B,C,D)

[Ac, Bc, Cc, Dc] = linmod('closed_loop_model')
sys_control_min = ss(Ac,Bc,Cc,Dc)
sysCL_min = feedback(sysOL_min, sys_control_min)
```

The Δ matrix is now of size $7X7$ and contains 7 blocks, each of size $1X1$ in contrast to a $9X9$ matrix for the previous cases. Following this, the closed loop stability of this model is studied using μ-analysis.

The upper and lower bounds of μ and k_r are:

$$[\underline{\mu}_{\Delta}, \mu_{\Delta}] = [0.8182, 0.859]$$

$$[\underline{k_r}, k_r] = [1.1640, 1.2222]$$

and are found to be exactly the same as computed from the branch-and-bound method. This solution is arrived at even without using branch-and-bound method, thereby being advantageous in terms of the computational cost. Thus, it can be concluded that reduction in Δ can prove to be a much more effective method in performing robustness analysis of a system in terms of computation as well as in simplicity.