ME - 524 - Advanced Control Systems
`datadriven` command additional material

Philippe Schuchert − Last revision April 28, 2021

Get the latest version using `git clone https://c4science.ch/source/AdvancedControlSystems.git`

Introduction

The provided `MATLAB` script, implements $\mathcal{H}_2$ or $\mathcal{H}_\infty$ mixed-sensitivity controller synthesis using SISO Frequency Response Functions (FRFs), contrasting with `mixsyn`, `h2syn` or `hinfstruct` using parametric models. The problems this function can solve are as follows:

$$\begin{array}{c} \min\limits_{K} \gamma \\ \text{subject to} \\ \left\| \ P_{\text{soft}}(G,K) \ \right\|_p \leq \gamma, \quad \left\| \ P_{\text{hard}}(G,K) \ \right\|_\infty \leq 1 \end{array}, \tag{1}$$

where p is either the 2 or ∞ norm, or a combination of both. $P_{\text{soft}}(G,K)$ is the augmented plant describing the objectives for the controller synthesis:

$$\|P_{\text{soft}}(G,K)\|_p = \left\| \begin{bmatrix} W_1\mathcal{S} \\ W_2\mathcal{T} \\ W_3\mathcal{U} \\ W_4\mathcal{D} \end{bmatrix} \right\|_p. \tag{2}$$

The definition of the different sensitivity functions can be found in the Appendix. $P_{\text{hard}}(G,K)$ are hard constraints on the various sensitivity functions:

$$\begin{array}{c} \left\| \ P_{\text{hard}}(G,K) \ \right\|_\infty \leq 1 \implies \\ \left\|\overline{W_1}\mathcal{S}\right\|_\infty \leq 1 \quad \left\|\overline{W_2}\mathcal{T}\right\|_\infty \leq 1 \\ \left\|\overline{W_3}\mathcal{U}\right\|_\infty \leq 1 \quad \left\|\overline{W_4}\mathcal{D}\right\|_\infty \leq 1 \end{array}. \tag{3}$$

An initial stabilizing controller K_0 must be provided. This controller is parametrized by:

$$K_0 = \frac{X_{\text{var}}(z)\cdot F_x(z)}{Y_{\text{var}}(z)\cdot F_y(z)}, \tag{4}$$

where F_x and F_y fixed parts in the controller's numerator and denominator. X_{var} and Y_{var} are the variable parts that can change throughout iterations in the iterative tuning process. F_y must include all poles of the controller on the unit circle. Since the final controller and initial controller must have the same order, the initial controller can be zero padded accordingly. As example, if a stabilizing controller

$$K_0 = 0.1\frac{z}{z-1}$$

is known, and the requested order for the final controller is 4, then the initial controller can be chosen as

$$K_0 = 0.1\frac{z}{z-1} = \frac{\overbrace{0.1z^4}^{X_{\text{var}}}}{\underbrace{z^3}_{Y_{\text{var}}}\underbrace{(z-1)}_{F_y}}, \tag{5}$$

and $F_x = 1$. The provided implementation always first tries to find a controller satisfying the constraints. This is achieved by adding slack-variables in the constraints, and optimizing over this slack variables.

Interface and numerical solver

The relevant scripts are located in the folder `datadriven`. The path to this folder has to be added to the `MATLAB` environment. The `+utils` folder inside corresponds to a namespace, and does not have to be added to the environment. The optimization problem is solved using `YALMIP`, which should be installed. An appropriate solver second-order conic solver (SOCP) is required. `SEDUMI` or `bnb` available freely (and already configured correctly when installing `YALMIP` from `MPT3`!) and will be used. To check if `YALMIP` and a SOCP solver is available, run the command `yalmiptest(sdpsettings('verbose',0))`. If the configuration has been done properly, the output in the console when running the script should look like:

```
 iter |   slack    |     obj     |  decrease   | Total SOLVE time
------|------------|-------------|-------------|-----------------
  001  | 0.0000e+00 |             |             | 1.3289e-02
  002  | 0.0000e+00 | +2.9567e+02 |             | 6.9408e-02
  003  | 0.0000e+00 | +8.7772e+01 | -2.0790e+02 | 1.1563e-01
  004  | 0.0000e+00 | +8.3596e+01 | -4.1766e+00 | 1.6091e-01
  005  | 0.0000e+00 | +8.2194e+01 | -1.4017e+00 | 2.1970e-01
  006  | 0.0000e+00 | +7.7230e+01 | -4.9638e+00 | 2.6382e-01
```

```
034  | 0.0000e+00 | +4.1448e+01 | -2.5442e-04 | 1.8328e+00
035  | 0.0000e+00 | +4.1447e+01 | -7.8685e-04 | 1.8890e+00
036  | 0.0000e+00 | +4.1445e+01 | -2.3849e-03 | 1.9377e+00
037  | 0.0000e+00 | +4.1437e+01 | -7.3866e-03 | 1.9841e+00
038  | 0.0000e+00 | +4.1423e+01 | -1.4346e-02 | 2.0430e+00
039  | 0.0000e+00 | +4.1422e+01 | -1.3388e-03 | 2.0987e+00
040  | 0.0000e+00 | +4.1422e+01 | -9.3259e-06 | 2.1462e+00
Elapsed time is 28.957170 seconds.
```

Note that for this example, the total solve time is only 2.14s, but the total elapsed time 28.9s. `YALMIP` is extremely inefficient in parsing data to the various solvers and should absolutely avoided when possible. `YALMIP` is still used here for convenience and clarity, as solve-time remain reasonable. The first iteration is always spent checking if controller satisfying the constraints (3) can be found. Here is such controller can be found after iteration 1 as `slack` $= 0$.

Inputs for `datadriven`

The function `datadriven` take four arguments as input:

`SYS`: Informations about the system. Structure with fields: `model`, `W`, `controller`. `model` is the FRF of the system. Must be a `frd` or equivalent[1]. Can be multiple models stacked for simultaneous stabilization. `W` is the vector of frequencies values where the problem will be solved. `controller` is a structure with the information about the initial controller, see provided template or example hereafter.

`OBJ`: Structure with weighting filters for the objectives. Has two fields, `o2` and `oinf`, each specifying the weights for the $\mathcal{H}_2$ and $\mathcal{H}_\infty$ mixed-sensitivity problem as described in (2). Must be a `frd` or equivalent.

`CON`: Structure with weighting filters for the constraints. Stores the information about the different weighting filter used for the constraints as described in (3). Must be a `frd` or equivalent.

`PAR`: Additional parameters to the optimization. Structure with the following parameters `tol`: controls when the iterative tuning process stops (minimal decrease in the objective), `maxIter`: maximal number of iteration in the iterative tuning process, `radius`: max radius of the poles of the controller ($r = 0.99$ works well in practice), `robustNyquist`: use additional stability condition with improved stability closed-loop guarantees.

When a value is left empty, the corresponding entry is ignored in the optimization problem (1). `SYS` must be specified correctly. In the structure controller, the coefficients of $X_{\text{var}}, Y_{\text{var}}, F_x, F_y$ should be specified as a row vector. For the example controller given in (5), this would equate to:

```
controller.num = [.1, 0, 0, 0, 0];
controller.den = [1, 0, 0, 0];
controller.Fx = 1;
controller.Fy = [1 -1];
controller.Ts = Ts;
```

and `Ts` the sampling time of the controller.

Outputs

The function returns two outputs:

`controller`: Structure with the optimal controller information, same field-names as `SYS.controller`.

`obj`: Structure with information about the optimal solution (value of the objective in (1) , values of the different norms, number of iterations, `slack`). If `slack` $\neq 0$, then no controller satisfying (3) could be found.

Appendix

The different closed-loop sentivity function of interest are:

$$\begin{aligned} \mathcal{S} &= \frac{1}{1+GK} & \mathcal{T} &= \frac{GK}{1+GK} \\ \mathcal{U} &= \frac{K}{1+GK} & \mathcal{D} &= \frac{G}{1+GK} \end{aligned}, \tag{6}$$

where $\mathcal{S} : n \to e$ is called the sensitivity, $\mathcal{T} : n \to y$ the complementary sensitivity, $\mathcal{U} : n \to u$ the noise sensitivity and $\mathcal{D} : d \to y$ the load disturbance sensitivity. The corresponding block diagram can be found in Fig. 1. The transfer-functions from the reference r to the according output are the same as the transfer function from n to the same output.

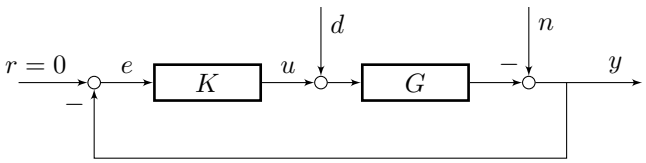

Figure 1: Block diagram of the feedback interconnection

[1] SISO LTI object (`TF`, `ss`, `zpk`, etc)