Data-driven SISO Controller Synthesis

Vaibhav Gupta

April 2024

This program aims to solve the following problem in a data-driven framework for SISO plant(s):

$$\min_{K} \max_{i=1,\dots,n} \left\| \begin{matrix} W_1\mathcal{S}_i \\ W_2\mathcal{T}_i \\ W_3\mathcal{U}_i \\ W_4\mathcal{V}_i \end{matrix} \right\|_{2/\infty} + \left\| W(L_d - G_iK) \right\|_{2/\infty} \tag{1}$$

such that,

$$\begin{aligned} \left\| W_1\mathcal{S}_i \right\|_\infty &\leq 1, \quad \forall i \\ \left\| W_2\mathcal{T}_i \right\|_\infty &\leq 1, \quad \forall i \\ \left\| W_3\mathcal{U}_i \right\|_\infty &\leq 1, \quad \forall i \\ \left\| W_4\mathcal{V}_i \right\|_\infty &\leq 1, \quad \forall i \end{aligned}$$

1 Table of Contents

2 Installation

Prerequisites for this program are:

- MATLAB
- YALMIP
- One of the following convex optimisation solvers:
  - Mosek (Recommended solver)
  - Sedumi

2.1 MATLAB

Recommended version is **2023b** or newer, while the minimum version is **2019b**. Furthermore, following MATLAB toolbox(-es) are required:

- Control System Toolbox

2.2 `tbxmanager` for YALMIP and/or sedumi

Refer for more details to tbxManager Website. A quick summary of steps is given here.

1. Create a `tbxmanager` folder, ideally in your MATLAB startup folder `{{HOME}}/Documents/MATLAB`.
2. Go to the `tbxmanager` folder.
3. Run the following command to download `tbxmanager`

```
websave("tbxmanager.m", "http://www.tbxmanager.com/tbxmanager.m");
```

4. Install `tbxmanager` and save the path to MATLAB.

```
tbxmanager
savepath
```

5. Install required toolboxes.

```
tbxmanager install yalmip
tbxmanager install sedumi
```

6. Edit/create `startup.m` in your Matlab startup folder `{{HOME}}/Documents/MATLAB` and put the following line there.

```
tbxmanager restorepath
```

NOTE: If you are using Apple's new chip M1 or newer, `sedumi` might *not* work for you. In that scenario, use `mosek` which is also the recommended solver.

2.3 MOSEK

Mosek is a commercial convex optimisation solver which is the recommended solver for this program. The details on how to obtain the license can be found in license subsection.

1. Download appropriate installer from MOSEK website and install the software.
2. Add MOSEK to the MATLAB path. Do not forget to replace `{{MOSEK_PATH}}` with the appropriate path to the MOSEK installation folder

```
addpath("{{MOSEK_PATH}}/mosek/10.1/toolbox/r2022b");
savepath
```

3. To validate if the installation has been successful, run the following command in MATLAB

```
mosekdiag
```

Regarding MOSEK License

1. For academic uses, a free license could be requested. For commercial purposes, a 30-day trial license is also available.
2. The obtained license file (`mosek.lic`) should be saved to `{{HOME}}/mosek` folder.

2.3.1 Fusion interface for MOSEK (Not supported)

MOSEK provides a fusion API which allows for a faster problem formulation, hence faster controller synthesis. To use this interface set the solver to 'fusion' in the program.

1. To add the MOSEK Fusion java `.jar` to the MATLAB environment, use the following command

```
javaaddpath("{{PATH}}/mosek/10.1/tools/platform/{{ARCH}}/bin/mosek.jar");
```

2. To validate if MOSEK Fusion has been successful added, run the following command in MATLAB

```
import mosek.fusion.*;
M = Model()
```

NOTE: If you are using Apple's new chip M1 or newer, MOSEK Fusion might *not* work for you. In that scenario, use MOSEK without Fusion API.

3 Usage

A template for usage is provided in `template_file.m`. Breakdown of the file is given here.

1. First an empty problem structure is loaded.

```
[SYS, OBJ, CON, PAR] = datadriven.utils.emptyStruct();
```

2. The problem structure is then filled with desired requirements for controller synthesis problem. (Refer to Problem Structure)

3. Solve the data-driven controller synthesis problem.

```
[K, sol] = datadriven.datadriven(SYS, OBJ, CON, PAR, verbosity, solver);
```

Inputs

- `verbosity` is a boolean which activates the iteration logs if `true`.
- `solver` is the name of convex optimiser to use.
  - `mosek` : Use MOSEK via YALMIP (Recommeded)
  - `sedumi` : Use SeDuMi via YALMIP
  - `fusion` : Use MOSEK via its Fusion API (Broken for Apple M chips, not maintained)

Outputs

- `K` gives the synthesised controller in transfer function form.
- `sol` gives the some diagnostic information about the solution.
  - `Hinf` : $\mathcal{H}_\infty$ objective (including Loop Shaping)
  - `H2` : $\mathcal{H}_2$ objective (including Loop Shaping)
  - `obj` : Value of the objective for the solution.
  - `slack` : Slacks for the constraints. This should be 0 if the constraints are satisfied!

4 Problem Structure

Problem structure contains following structures:

- `SYS` : System and Initial Controller
- `OBJ` : Objectives
- `CON` : Constraints
- `PAR` : Parameters

4.1 System and Initial Controller

The `SYS` object contains the information about the system(s) and the initial stabilising controller.

```
SYS
├─ controller ..... Initial stabilising controller
│   ├─ num ..... Numerator without fixed part
│   ├─ den ..... Denominator without fixed part
│   ├─ Ts ..... Sampling time
│   ├─ Fx ..... Fixed part of the numerator
│   └─ Fy ..... Fixed part of the denominator
├─ model ..... Model of the system(s)
└─ W ..... Frequencies for controller synthesis
```

4.1.1 Intial stabilising controller

Initial controller which stabilises all systems. The numerator and denominator can be extracted from the controller object `Kc` and set to the same order as the desired controller.

```
[num, den] = tfdata(Kc, 'v');   % Extract numerator and denominator
den(order + 1) = 0;     % Zero padding to have same order as the desired controller
num(order + 1) = 0;     % Zero padding to have same order as the desired controller
```

Then, the fixed terms `Fx` and `Fy` can be removed from the extracted numerator and denominator using

```
num = deconv(num, Fx);  % Remove fixed part of numerator
den = deconv(den, Fy);  % Remove fixed part of denominator
```

4.1.2 Model of the system(s)

Accepts any type of SISO model than can be called by the `freqresp` function (`ss` , `tf` , `frd` , etc). For the multi-model case, the different models should be stacked.

```
G = stack(1, G1, G2, G3)
```

4.1.3 Frequencies for controller synthesis

`W` gives the frequency vector at which the optimisation problem is to be solved.

4.2 Objectives

`OBJ` object contains the information about the design objectives.

- **OBJ**
  - `oinf` ..... $\mathcal{H}_\infty$ Objective
    - `W1`
    - `W2`
    - `W3`
    - `W4`
  - `o2` ..... $\mathcal{H}_2$ Objective
    - `W1`
    - `W2`
    - `W3`
    - `W4`
  - `LSinf` ..... $\mathcal{H}_\infty$ Loop Shaping
    - `W`
    - `Ld`
  - `LS2` ..... $\mathcal{H}_2$ Loop Shaping
    - `W`
    - `Ld`

4.2.1 $\mathcal{H}_\infty$ Objective

The `oinf` object specifies the filters for $\mathcal{H}_\infty$ objective. `W1`, `W2`, `W3`, and `W4` corresponds to the filters W_1, W_2, W_3, W_4 in the following objective.

$$\left\| \begin{matrix} W_1\mathcal{S}_i \\ W_2\mathcal{T}_i \\ W_3\mathcal{U}_i \\ W_4\mathcal{V}_i \end{matrix} \right\|_\infty \tag{2}$$

where,

$$\mathcal{S}_i = \frac{1}{1 - G_i K}$$
$$\mathcal{T}_i = \frac{G_i K}{1 - G_i K}$$
$$\mathcal{U}_i = \frac{K}{1 - G_i K}$$
$$\mathcal{V}_i = \frac{G_i}{1 - G_i K}$$

All the filters should either be callable by the `freqresp` function or `[]` to denote that they are not used.

4.2.2 $\mathcal{H}_2$ Objective

The `o2` object specifies the filters for $\mathcal{H}_2$ objective. `W1`, `W2`, `W3`, and `W4` corresponds to the filters W_1, W_2, W_3, W_4 in the following objective.

$$\left\| \begin{matrix} W_1\mathcal{S}_i \\ W_2\mathcal{T}_i \\ W_3\mathcal{U}_i \\ W_4\mathcal{V}_i \end{matrix} \right\|_2 \tag{3}$$

where,

$$\mathcal{S}_i = \frac{1}{1 - G_i K}$$
$$\mathcal{T}_i = \frac{G_i K}{1 - G_i K}$$
$$\mathcal{U}_i = \frac{K}{1 - G_i K}$$
$$\mathcal{V}_i = \frac{G_i}{1 - G_i K}$$

All the filters should either be callable by the `freqresp` function or `[]` to denote that they are not used.

4.2.3 $\mathcal{H}_\infty$ Loop Shaping

The `LSinf` object specifies the filter and the model for $\mathcal{H}_\infty$ loop shaping. `W` and `Ld` corresponds to the filter W and the model L_d in the following objective.

$$\|W(L_d - G_i K)\|_\infty \tag{4}$$

Both the filter and the model should either be callable by the `freqresp` function or `[]` to denote that they are not used.

4.2.4 $\mathcal{H}_2$ Loop Shaping

The `LS2` object specifies the filter and the model for $\mathcal{H}_2$ loop shaping. `W` and `Ld` corresponds to the filter W and the model L_d in the following objective.

$$\|W(L_d - G_i K)\|_2 \tag{5}$$

Both the filter and the model should either be callable by the `freqresp` function or `[]` to denote that they are not used.

4.3 Constraints

`CON` ..... $\mathcal{H}_\infty$ Constraints
- `W1`
- `W2`
- `W3`
- `W4`

The `CON` object specifies the filters for $\mathcal{H}_\infty$ constraint. `W1`, `W2`, `W3`, and `W4` corresponds to the filters W_1, W_2, W_3, W_4 in the following constraints.

$$\|W_1 \mathcal{S}_i\|_\infty \leq 1 \tag{6}$$
$$\|W_2 \mathcal{T}_i\|_\infty \leq 1 \tag{7}$$
$$\|W_3 \mathcal{U}_i\|_\infty \leq 1 \tag{8}$$
$$\|W_4 \mathcal{V}_i\|_\infty \leq 1 \tag{9}$$

where,

$$\mathcal{S}_i = \frac{1}{1 - G_i K}$$
$$\mathcal{T}_i = \frac{G_i K}{1 - G_i K}$$
$$\mathcal{U}_i = \frac{K}{1 - G_i K}$$
$$\mathcal{V}_i = \frac{G_i}{1 - G_i K}$$

All the filters should either be callable by the `freqresp` function or `[]` to denote that they are not used.

4.4 Synthesis parameters

The `PAR` object contains the information about the design objectives.

```
PAR
├─ tol ..................................... Numerical tolerance for convergence
├─ maxIter ................................. Maximum number of allowed iterations
├─ radius .................................. Ensure that the controller poles are within given radius
├─ robustNyquist ........................... Ensure robustness of closed-loop system (True/False)
└─ robustEllipsoid ......................... Ensure robustness to ellipsoidal uncertainty set
```

- `tol`: Numerical tolerance for convergence of the data-driven controller synthesis iterations.
  Default: 10^{-6}
- `maxIter`: Maximum number of allowed iterations of the data-driven controller synthesis.
  Default: 100
- `radius`: Radius of the circle within which the controller poles should lie.
  Default: 1
- `robustNyquist`: Add robustness condition for closed-loop stability at inter-frequency points.
  Default: `true`
- `robustEllipsoid`: Ensure robustness to ellipsoidal uncertainty set. [1]
  Default: `[]`

[1] V. Gupta, E. Klauser, and A. Karimi, "Data-driven IQC-Based Uncertainty Modelling for Robust Control Design," IFAC-PapersOnLine, vol. 56, no. 2, pp. 4789–4795, Jan. 2023, doi: 10.1016/j.ifacol.2023.10.1244. (Available at `https://infoscience.epfl.ch/record/301605`)