SATAW-Tool: a SATuration AWare matlab Toolbox

Isabelle Queinnec, Sophie Tarbouriech *

March 25, 2013

Abstract

This is a user guide for the SATAW toolbox. At this date the toolbox performs analysis and control design operations for linear systems interconnected with saturation elements. LMI based functionalities are provided. All the theoretical material used in this toolbox has been published in [6].

1 Introduction

The toolbox manipulates a flexible description of the system, controller and actuator using simple structure elements. The models for systems and controllers are in state-space, and only the continuous-time case is provided in the toolbox. For the saturation modeling, sector conditions are used. In this representation, the saturation term is replaced by a dead-zone nonlinearity. Hence, sector conditions, locally or globally valid, can be used to provide stability and stabilization conditions.

Based on these modeling functionalities, the toolbox allows to build Linear Matrix Inequality (LMI) optimization formulations for stability and performance analysis and design. For stability problem, optimization of the size of the region of asymptotic stability is performed in the local case, or a simple feasibility problem is solved in the global case. For performance problem, amplitude-bounded or energy-bounded external signals are considered. The optimization may focus both on the size of the domain where the trajectory is confined or on the size of admissible disturbance. State feedback and dynamic feedback controllers are considered in the case of a simple saturation element. Static or full-order anti-windup compensators may be designed, and any-order anti-windup compensators may be analyzed. The results are based on theoretical results that manipulate quadratic Lyapunov functions and ellipsoidal sets.

All analysis and design problems are formulated as LMIs with the YALMIP format, [3], [4]. Optimization can therefore be performed with any Semi-Definite Programming (SDP) solver interfaced by YALMIP. The users can therefore, if they desire, fully use the potentialities of the solvers and the YALMIP functionalities. Moreover, the manipulated state-space models of the plant and/or controller may be built either using a simple structure or, for those familiar with RoMulOC [5], using the dedicated Matlab object `ssmodel`.

The package then includes several functions for:

- state feedback or output feedback analysis, in presence of position saturation and/or rate saturation;
- state feedback or output feedback design, in presence of rate saturation;
- state feedback design, in presence of position saturation;
- static and dynamic anti-windup analysis, in presence of various saturated-based nonlinear elements;
- static and full-order anti-windup design, in presence of various saturated-based nonlinear elements.

*I. Queinnec and S. Tarbouriech are with CNRS, LAAS, 7 Av. du Colonel Roche, F-31400 Toulouse, France and with Univ de Toulouse, LAAS, F-31400 Toulouse, France. `queinnec,tarbour@laas.fr`

2 Getting started

2.1 Install SATAW-Tool

SATAW-Tool is entirely based on m-code, and is thus easy to install. Remove any old version of SATAW-Tool, unzip the file sataw.zip and add the directory to your MATLAB path as:

```
>> addpath whereIputTHEdirectory/SATAW-Tool
```

When this is done, you need yet to install YALMIP:

`http://users.isy.liu.se/johanl/yalmip/`

Then, except if you intend to use the SDP solver of the robust control toolbox (LMILAB), one SDP solver (at least) among those listed in

`http://www.mathworks.com/matlabcentral/fx_files/2092/1/content/yalmip/htmldata/solvers.htm`

has also to be installed, as described in the solver manuals. Make sure to add required paths. To test your installation, run the commands `yalmiptest.m` and `sataw.m`.

It is also recommended to install RoMulOC:

`http://www.laas.fr/OLOCEP/romuloc/`

to use the Matlab object `ssmodel` to define systems and controllers. It is entirely based on m-code and you just need to unzip `romuloc.zip` and to add the required path (see [5]). Actually, systems and controllers may be defined, in the current version, using a structure or the Matlab object `ssmodel`. However, it is expected, in the close future, to unify RoMulOC and SATAW-Tool platform, in which case, the description of the systems with structures would not be maintained.

2.2 License agreement

SATAW-Tool is free of charge and openly distributed, in the hope that the will be useful, but without any warranty. You are free to use any of the files of the following toolbox for personal or academic use. It may not be re-distributed as a part of a commercial product. Neither the authors nor CNRS accept any responsibility or liability with regard to this software that is licensed on an "as is" basis.

SATAW-Tool and YALMIP must be referenced when used in a published work:

```
@manual{sataw/manual,
author = "I. Queinnec and S. Tarbouriech",
 title = "{SATAW-Tool}: a SATuration AWare Toolbox",
 url = {http://homepages.laas.fr/queinnec/sataw-tool.html},
 year = "2012"}

@manual{lof/04a,
     Author = {J. L{\"o}fberg},
     Title = "{YALMIP}: A Toolbox for Modeling and Optimization in MATLAB",
     url = {http://users.isy.liu.se/johanl/yalmip/},
     year = {2004}}

@InProceedings{lof/04b,
      author = "J. L{\"o}fberg",
      title = "{YALMIP}: A Toolbox for Modeling and Optimization in {MATLAB}",
      booktitle = "Proceedings of the {CACSD} Conference",
      year = "2004",
      address = "Taipei, Taiwan"}
```

so as RoMulOC if the Matlab object `ssmodel` is used:

```
@manual{romuloc,
      author = "D. Peaucelle",
      title = "{RoMulOC} a YALMIP-MATLAB based Robust Multi Objective Control Toolbox",
      url = {http://www.laas.fr/OLOCEP/romuloc},
      year = "2005"}
```

3 A short example for a start

The following example, partially taken from [6] (Example 3.1), is used to illustrate various aspects of the toolbox. It corresponds to the state feedback control of an unstable open-loop system through a saturation element, such as schemed in Figure 1,

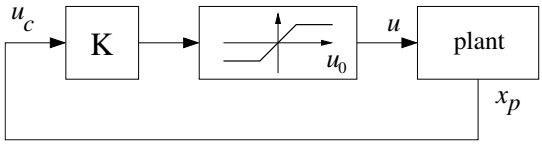

Figure 1: State feedback architecture for the position saturation problem

and described by the following data:

$$A = \begin{bmatrix} 0.1 & -0.1 \\ 0.1 & -3 \end{bmatrix} \; ; \qquad B = \begin{bmatrix} 5 & 0 \\ 0 & 1 \end{bmatrix}$$
$$K = \begin{bmatrix} -0.7283 & -0.0338 \\ -0.0135 & -1.3583 \end{bmatrix} \; ; \quad u_0 = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

Let us follow step by step the example without getting into details of the functions used. Simply build the model, controller and actuator as follows[1]:

```
>> clc
>> clear
>> sysP = ssmodel('A short example');              % This line is optional
>> sysP.A = [0.1 -0.1;0.1 -3];
>> sysP.Bu = [5 0;0 1];
>> sysC = ssmodel('Its state feedback control');   % This line is also optional
>> sysC.Dyu = [-0.7283 -0.0338;-0.0135 -1.3583];
>> sysACT.u0 = [5;2];
```

At this stage, we have declared a continuous-time system, with a static feedback loop, and considering the default case (`sysP.Cy` not defined) of the state feedback case. The stability analysis of this system, considering a Lyapunov approach using a sector non-linearity model approach is implemented in SATAW-Tool in `PsatAL`. Its simplest use is of the form:

```
>> resu1 = psatal(sysP,sysACT,sysC);

You have been solving the following problem:
    -------------
Position saturation - use of Sector nonlinearity model
LOCAL ANALYSIS (state feedback), max(trace(W)), no conditioning on W
No disturbance
```

[1]The two optional lines correspond to the use, or not, of the Matlab class `ssmodel` borrowed from RoMulOC

```
The problem has been found feasible
    -------------
>> resu1.W

ans =

   1.0e+07 *

    0.0055    0.0464
    0.0464    1.7694
```

The matrix W builds the region of asymptotic stability $\mathcal{E}(W^{-1}, \eta)$, with $\eta = 1$ in the case without disturbance. In this execution of the function, the default case has been considered for the optimization problem, that is, the solver 'lmilab' has been used, with an optimization of the region of asymptotic stability $\mathcal{E}(W^{-1}, 1)$, thanks to the optimization criterion $\max trace(W)$.

Let us now consider for the optimization criterion the maximization in the directions formed by the vertices of the unit square box. `optsat` is declared as

```
>> optsat.W = 3;
>> optsat.Xi0 = [1 1 -1 -1;1 -1 1 -1];
```

The other elements of `optsat` takes their default value. The execution of `psatal` is now:

```
>> resu2 = psatal(sysP,sysACT,sysC,'',optsat);

You have been solving the following problem:
    -------------
Position saturation - use of Sector nonlinearity model
LOCAL ANALYSIS (state feedback), max beta (scaling of Xi0), no conditioning on W
No disturbance

The problem has been found feasible
    -------------
```

Note that the fourth element of `psalal` not used in the current case, is replaced by `''` (or by []). In plus of matrix W, one can be interested in the value of β which expresses the scale expansion of $\mathcal{X}_0$.

```
>> resu2.beta

ans =

  244.4049
```

Consider now that the state feedback controller is replaced by a dynamic output feedback controller given by:

$$A_c = \begin{bmatrix} -171.2 & 27.2 \\ -68 & -626.8 \end{bmatrix} \; ; \quad B_c = \begin{bmatrix} -592.2 & 5.539 \\ -4.567 & 149.8 \end{bmatrix}$$
$$C_c = \begin{bmatrix} 0.146 & 0.088 \\ -6.821 & -5.67 \end{bmatrix} \; ; \quad D_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The stability analysis of this problem, considering for the optimization criterion the maximization in the directions formed by the vertices of the unit square box in the plant state, is then updated as follows

```
>> clear sysC.D
>> sysC = ssmodel('A dynamic feedback control'');      % This line is optional
>> sysC.A = [-171.2 27.2;-68 -626.8];
>> sysC.Bu = [-592.2 5.539;-4.567 149.8];
```

```
>> sysC.Cy = [0.146 0.088;-6.821 -5.67];
>> optsat.Xi0 = [1 1 -1 -1;1 -1 1 -1;0 0 0 0;0 0 0 0];
>> resu3 = psatal(sysP,sysACT,sysC,'',optsat);

You have been solving the following problem:
    -------------
Position saturation - use of Sector nonlinearity model
LOCAL ANALYSIS (dynamic feedback), max beta (scaling of Xi0), no conditioning on W
No disturbance

The problem has been found feasible
    -------------
>> resu3.beta

ans =

  250.2228
```

Note that `optsat.Xi0` has ben updated such that each vector which forms the matrix $\mathcal{X}_0$ is of the dimension of the augmented system $n + n_c$.

To go further, consider now that the system is subject to an additive energy-bounded disturbance, defined by matrix R and bound δ (see later equation (4)), which acts on its dynamics through an input matrix

$$B_w = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

The optimization problem (with the dynamic controller and same criterion) is then updated as follows

```
>> sysP.Bw = [1;1];
>> sysD.opt = 'energy';
>> sysD.R = 1;
>> sysD.delta = 1.2;
>> resu4 = psatal(sysP,sysACT,sysC,sysD,optsat);

You have been solving the following problem:
    -------------
Position saturation - use of Sector nonlinearity model
LOCAL ANALYSIS (dynamic feedback), max beta (scaling of Xi0), no conditioning on W
Energy bounded disturbance

The problem has been found feasible
    -------------
>> resu4.beta

ans =

   15.7648
```

Another problem which may be of interest is to evaluate the maximal admissible disturbance ($\min \delta$), with δ^{-1} is the $\mathcal{L}_2$-bound of the disturbance, without restriction on the size of the admissible region of initial states. This is done as follow:

```
>> optsat.D = 1;
>> optsat.W = 0;
>> resu5 = psatal(sysP,sysACT,sysC,sysD,optsat);
```

```
Warning: optsat.Xi0 is not used with your selection for optsat.W

You have been solving the following problem:
    -------------
Position saturation - use of Sector nonlinearity model
LOCAL ANALYSIS (dynamic feedback), no optimization of W, no conditioning on W
Energy bounded disturbance, optim delta

The problem has been found feasible
    -------------
>> delta

delta =

    0.0012
```

4 Modeling the problem

4.1 Basic architecture of the problems solved in the toolbox

The full problem, including an anti-windup compensation, is basically described in Figure 2.

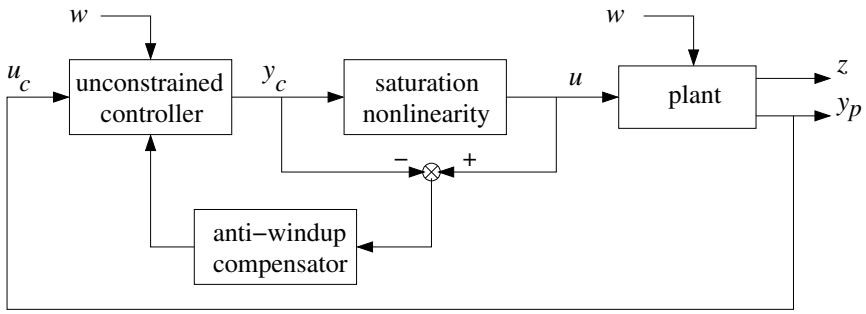

Figure 2: Full architecture for the saturation problem, including anti-windup compensation

The toolbox allows considering several saturation nonlinearities, from a simple position saturation to some complex nested saturation situations (see subsection 4.4). For the case without additional anti-windup compensation, both static state feedback and dynamic output feedback may be analyzed. Only static state feedback controllers may be designed by manipulating LMIs problems. Dynamic feedback design (both of controller and anti-windup compensator) necessitates some alternative treatments (see [6] for details). Various anti-windup schemes are proposed.The simplest case solved by the toolbox is the state feedback problem in presence of a position saturation in the control loop, such as it has been given in Figure 1.

4.2 Define the plant model

We consider the following state-space representation for the continuous-time linear plant[2]:

$$\left\{ \begin{array}{rcl} \dot{x}_p & = & A_p x_p + B_{pu} u + B_{pw} w \\ y_p & = & C_{py} x_p + D_{pyu} u + D_{pyw} w \\ z & = & C_z x_p + D_{zu} u + D_{zw} w \end{array} \right. \quad (1)$$

[2]For simplicity, the time dependence in the vector will be omitted.

where $x_p \in \Re^{n_p}$, $u \in \Re^m$, $w \in \Re^q$ and $y_p \in \Re^p$ are the state, the input, the exogenous input and the measured output vectors of the plant, respectively. $z \in \Re^l$ is the regulated output vector used for performance purposes. Matrices A_p, B_{pu}, B_{pw}, C_{py}, C_z, D_{pyu}, D_{pyw}, D_{zu} and D_{zw} are real constant matrices of appropriate dimensions. Pairs (A_p, B_{pu}) and (C_{py}, A_p) are assumed to be controllable and observable, respectively.

The plant model may then be built either as a simple structure or as the dedicated Matlab object `ssmodel` borrowed from `RoMulOC`[3] where only the requested elements for the analysis or design are defined. In the full model case, the plant model is defined as in the following example:

```
    %% SysP is a structure %%

>> sysP

sysP =

      A: [npxnp double]
     Bu: [npxm double]
     Bw: [npxq double]
     Cy: [pxnp double]
    Dyu: [pxm double]
    Dyw: [pxq double]
     Cz: [lxnp double]
     Dz: [lxm double]
    Dzw: [lxq double]
```

```
        %% SysP is a ssmodel %%

>> sysP
name: Plant model with all matrices involved
          n=2    mw=1    mu=1
 n=2 dx  =   A*x +  Bw*w +  Bu*u
pz=1  z  =  Cz*x + Dzw*w + Dzu*u
py=2  y  =  Cy*x + Dyw*w + Dyu*u
continuous time ( dx : derivative operator )
```

Remark 1 *All unspecified matrices of a problem are set equal to null matrices of appropriate dimensions, except* `sysP.Cy` *(C_p) which is set equal to the identity matrix when used in the analysis and design functions. This is done by* `testsysp` *and this is particularly used in the case of state feedback analysis or design where the output is the state but generally not explicitly defined.*

The simplest case corresponding to a state feedback control problem, with additive disturbance, is simply defined as:

```
    %% SysP is a structure %%

>> sysP

sysP =

      A: [npxnp double]
     Bu: [npxm double]
```

```
        %% SysP is a ssmodel %%

>> sysP
name: Plant model for state feedback pbs
            n=2    mu=1
 n=2 dx  =   A*x +  Bu*u
continuous time ( dx : derivative operator )
```

Remark 2 *For any problem, at least* `sysP.A` *and* `sysP.Bu` *have to be defined.*

4.3 Define the controller

In its more general form, we assume an n_cth-order dynamic output stabilizing compensator

$$\left\{ \begin{array}{rcl} \dot{x}_c & = & A_c x_c + B_c u_c + B_{cw} w + v_x \\ y_c & = & C_c x_c + D_c u_c + D_{cw} w + v_y \end{array} \right. \tag{2}$$

where $x_c \in \Re^{n_c}$ is the controller state, $u_c \in \Re^p$ is the controller input, $y_c \in \Re^m$ is the controller output and v_x, v_y are extra inputs used for anti-windup purposes.

[3] See its user guide in [5] for a full description of this object.

Remark 3 *Note that the closed-loop system has to be well-posed, i.e. matrix* $\Delta = I_m - D_c D_{pu}$ *is invertible. Furthermore the unconstrained closed-loop dynamic matrix*

$$\mathbb{A} = \begin{bmatrix} A_p + B_{pu}\Delta^{-1} D_c C_p & B_{pu}\Delta^{-1} C_c \\ B_c(I_p + D_{pu}\Delta^{-1} D_c) C_p & A_c + B_c D_{pu}\Delta^{-1} C_c \end{bmatrix} \tag{3}$$

is necessarily Hurwitz, i.e., in the absence of control bounds, the closed-loop system is globally exponentially stable.

In a control design problem, the controller is not a priori defined. In the analysis problem, as for the plant model, a structure or a `ssmodel` object is used as well for the output feedback case:

```
    %% SysC is a structure %%

>> sysC

sysC =

     A: [ncxnc double]
    Bu: [ncxp double]
    Bw: [ncxq double]
    Cy: [mxnc double]
   Dyu: [mxp double]
    Dw: [mxq double]
```

```
        %% SysC is a ssmodel %%

>> sysC
name: Controller with all matrices involved
          n=2    mw=1    mu=2
 n=2 dx  =   A*x +  Bw*w +  Bu*u
py=2  y  =  Cy*x + Dyw*w + Dyu*u
continuous time ( dx : derivative operator )
```

and for the simple state feedback case:

```
    %% SysC is a structure %%

>> sysC

sysC =

    Dyu: [mxn double]
```

```
        %% SysC is a ssmodel %%

>> sysC
name: State feedback controller
static gain   mu=2
py=2     y  = Dyu*u
continuous time ( dx : derivative operator )
```

Remark 4 *All unspecified matrices of a problem are set equal to null matrices of appropriate dimensions.*

Remark 5 *The dimension of the disturbance vector* w *is imposed by the plant. This signifies that at least one matrix of* `sysP` *related to the disturbance has to be defined (even null), to explicit the presence of disturbance (and its dimension* q*) and then to use it in the controller dynamics. The consistency of* `sysC` *with respect to the size of* `sysP` *(interconnection and disturbance vector) is checked with the function* `testsysc`.

4.4 Define the actuator

The actuator is formed with interconnected saturation elements, each of them being described by their symmetric bound.

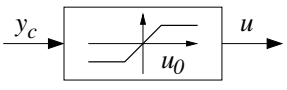

Figure 3: The saturation block

For the case of a position saturation given in Figure 3, the actuator is defined by:

```
>> sysACT

sysACT =

    u0: [mx1 double]
```

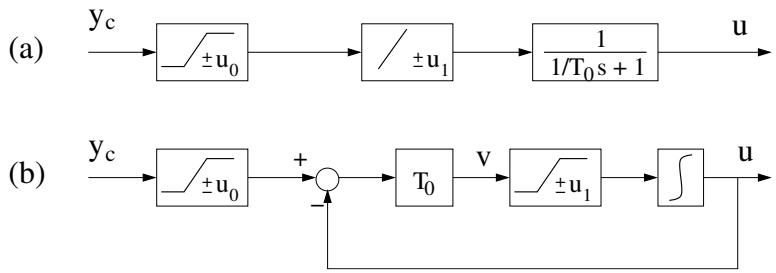

Figure 4: (a) Actuator with position and rate saturations. (b) Surrogate structure for the rate saturation. Figure given for the scalar case

For the case of a position and rate saturation such as given in Figure 4, the actuator is defined by:

```
>> sysACT

sysACT =

    u0: [mx1 double]
    u1: [mx1 double]
    Tr: [mxm double]  % diagonal matrix
```

Remark 6 *The consistency of* `sysACT` *with respect to the size of* `sysP` *is checked with the function* `testsysact`.

4.5 Define the disturbance

For stability problem, the exogenous signal w is set equal to 0. For performance problem, two kinds of disturbances, namely amplitude-bounded and energy-bounded are considered.

Amplitude-bounded exogenous signals w are bounded by a quadratic norm which reflects amplitude. They belong to the set:

$$\mathcal{W} = \left\{ w \in \Re^q \ ; \ w'Rw \leq \delta^{-1} \right\} \tag{4}$$

with $\delta > 0$ and $R = R' > 0$.

On the other hand, energy-bounded exogenous signals w belong to the set:

$$\mathcal{W} = \left\{ w \in \Re^q; \ ; \ \int_0^\infty w(\tau)' R w(\tau) \ d\tau \leq \delta^{-1} \right\} \tag{5}$$

with $\delta > 0$ and $R = R' > 0$.

The disturbance characteristics is set in the structure `sysD`.

- At least the `sysD.opt` has to be defined if disturbance is considered in the problem. `sysD.opt` may take value `'none'`, `'amplitude'` or `'energy'`.
- `sysD.R` may be omitted. In that case, it is set to the identity matrix of dimension q.
- `sysD.delta` may be omitted. n that case, it is set to 1.

- `sysD.tau1` and `sysD.tau2` are parameters of the S-procedure used only in the case of amplitude-bounded disturbance. They may be omitted, in which case default values are `sysD.tau1 = 1` and `sysD.tau2 = 1`. In case of no disturbance or energy-bounded disturbance they are forced to `sysD.tau1 = 0` and `sysD.tau2 = 1`, even if specified otherwise.

Remark 7 *Note that for the case of amplitude-bounded disturbance,* `sysD.tau1` *has to satisfy* $\tau_1 > |2\lambda_{max}(A_{cl})|$*, where* A_{cl} *corresponds to the closed-loop dynamics when the saturation element is omitted.*

Remark 8 *When no structure* `sysD` *is defined, the default case is used (*`sysD.opt = 'none'`*) AND the dimension q of the disturbance is forced to 0 (with appropriate associated null matrices, independently of what has been set in* `SysP` *and* `sysC`*). More generally, the consistency of* `sysD` *with respect to the model plant* `sysP` *and controller* `sysC` *(if given) is checked in the function* `testsysd`*.*

4.6 Define the anti-windup

For anti-windup compensation, the general form of the anti-windup scheme is a linear filter which produces the signals v_x and v_y as an output:

$$\left\{ \begin{array}{rcl} \dot{x}_{aw} & = & A_{aw}x_{aw} + B_{aw}u_{aw} \\ \left[\begin{array}{c} v_x \\ v_y \end{array} \right] & = & C_{aw}x_{aw} + D_{aw}u_{aw} \end{array} \right. \tag{6}$$

where $x_{aw} \in \Re^{n_a}$ is the anti-windup state, u_{aw} is the anti-windup input formed with dead-zone elements around each saturation block (of full dimension m_{aw}), and $\left[\begin{array}{cc} v_x' & v_y' \end{array} \right]' \in \Re^{n_c+m}$ is the anti-windup output interconnected with system (2).

Yet again, a structure or a `ssmodel` may be used to define the anti-windup compensator which displays as:

```
    %% SysAW is a structure %%

>> sysAW

sysAW =

    A: [naxna double]
   Cy: [(nc+m)xna double]
   Bu: [naxm double]
  Dyu: [(nc+m)xm double]
```

```
        %% SysAW is a ssmodel %%


 >> sysAW
name: Dynamic antiwindup
                n=na    mu=m
 n=na   dx  =   A*x +  Bu*u
py=nc+m  y  =  Cy*x + Dyu*u
continuous time ( dx : derivative operator )
```

Remark 9 *The consistency of* `sysAW` *with respect to the size of* `sysC` *(and in particular that the output of* `sysAW` *is of dimension* $n_c + m$*) is checked with the function* `testsysaw`*. All unspecified matrices of a problem are set equal to null matrices of appropriate dimensions.*

Remark 10 *In the current version of the toolbox, the anti-windup cannot involve disturbance input.*

In case of anti-windup compensator design, in plus of the selection of a static, dynamic given or dynamic compensator selected through the choice of the dedicated function, three cases studies may be considered as options of the problem (using `optsat.AW`, defined in section 5.2):

- `optsat.AW = 0`: acts both on the dynamics and output of the controller `sysC` (Default)
- `optsat.AW = 1`: acts on the controller dynamics only ($v_y = 0$);
- `optsat.AW = 2`: acts of the controller output only ($v_x = 0$).

5 Solving problems

5.1 List of problems addressed by SATAW-Tool

Once the closed-loop structure has been defined, you can solve several problems, either related to analysis:

- `psatal`: Position saturation, stability Analysis (controller `sysC` given) in the local context;
- `psatag`: Position saturation, stability analysis (controller `sysC` given) in the global context;
- `prsatal`: Position + rate saturation, stability Analysis (controller `sysC` given) in the local context;
- `prsatag`: Position + rate saturation, stability analysis (controller `sysC` given) in the global context;
- `pawal`: Position saturation, anti-windup analysis (controller `sysC` and anti-windup `sysAW` given) in the local context;
- `paw`: Position saturation, anti-windup analysis (controller `sysC` and anti-windup `sysAW` given) in the global context;

or to controller design:

- `psatsl`: Position saturation, state feedback design in the local context;
- `psatsg`: Position saturation, state feedback design in the global context;
- `psatdl`: Position saturation, dynamic output feedback design in the local context;

or to anti-windup design:

- `pawsl`: Position saturation, static anti-windup design (controller given) in the local context;
- `pawsg`: Position saturation, static anti-windup design (controller given) in the global context.

Remark 11 *Note that other functions will be added soon. This is only a test version of the toolbox.*

5.2 Input and output of the functions

All these functions are used with the same syntax including

- INPUT parameters
  - `sysP`: the state-space model of the plant (see section 4.2);
  - `sysACT`: the actuator (see section 4.4);
  - `sysC` : the state-space model of the controller (see section 4.3). This variable is optional in the design functions (but it may be used to add a given disturbance for the dynamics or output of the controller to be designed);
  - `sysD`: the disturbance elements (see section 4.5. This variable is optional. If not used, no disturbance is considered (even if disturbance input matrices are set to non-null matrices). On the other hand, if all disturbance matrices are set equal to 0, this variable is not considered.
  - `sysAW`: the anti-windup compensator. Note that this variable is optional, and appears only in the functions related to anti-windup analysis and design.
  - `optsat`: the optimization options (see below).
- OUTPUT parameters
  - `resu`: includes the variables issued from the optimization step. `resu.problem` is the error code generated by yalmip (it may be interpreted with `yalmiperror`). `resu.W`, `resu.eta` and `resu.beta` are related to the set of admissible initial states (see section 5.3). When used, `resu.delta` and `resu.gamma` are related to the disturbance (see section 5.4);

– `sysP`: it is almost not changed from the input parameter `sysP`, except that `sysP.Cy` may have been forced to the identity matrix if initially undefined.

– `sysC`: It returns the controller, unchanged in the analysis case, solution to the design problem otherwise.

– `sysD`: it returns the updated variable `sysD`. Note that this variable may have been arbitrarily changed in the analysis and design functions to be in coherence with the problem (see for example Remark 8). It allows to verify the exact disturbance problem evaluated. Note also that, if `sysD` has been changed during the execution of the design or analysis function, a message is displayed in the summary of the problem solved.

– `sysAW`: only for the functions related to anti-windup analysis and design. It returns the anti-windup, unchanged in the analysis case, solution to the anti-windup design problem otherwise.

All the conditions included in the Matlab functions provided in the toolbox are given as a set of LMI conditions. It is allowed to check the feasibility of the conditions (without optimization criteria) or to compute solutions to optimization problems considering various possible cost functions. All the optimization options used by these functions are set in the structure `optsat` which includes five optional elements:

- `optsat.W`: related to the set of "stability" (see subsection 5.3).
- `optsat.Xi0`: related to a criterion on the set of "stability" (see subsection 5.3).
- `optsat.D`: related to the size of the disturbance (see subsection 5.4).
- `optsat.cond`: the value sets some conditioning on matrix W. Default case is 'no conditioning", or set `optsat.cond = 0`.
- `optsat.AW`: related to the anti-windup output structure (see subsection 4.6).
- `optsat.SOLVER`: choice of solver in sdpsettings. It may be 'lmilab', 'sedumi'...

Remark 12 *Classically in the case of multi-objective criteria, a compromise is done between the different variables to be minimized. In the current version of the toolbox, all the weights are set to 1 although this may not be the "best" selection of the weights.*

5.3 Optimization of the ellipsoidal estimate of the set of admissible initial states

Typically, in the *local context*, a main issue is to find the best ellipsoidal estimate

$$\mathcal{E}(W^{-1},\eta) = \left\{x \in \Re^{n_{tot}} \ ; \ \ xW^{-1}x \leq \eta^{-1}\right\} \tag{7}$$

for the set of admissible initial states for the closed-loop saturated system, i.e. the best ellipsoidal estimate of the region of asymptotic stability (RAS) in the case without disturbance or of the region where the trajectory is confined in the case with additive disturbance. x represents the augmented state vector incorporating the state of the plant, the state of the controller, the state of the actuator and the state of the anti-windup scheme. Several size criteria may be considered which generally orient the form of the ellipsoid [2], [1], related to matrix W and parameter η ($\eta = 1$ without loss of generality in cases without additive disturbance). The following cases have been programmed:

- `optsat.W = 1`: $\min -\text{trace}(W) + \eta$ (Default)
- `optsat.W = 2`: $\min \text{trace}(W^{-1}) + \eta$
- `optsat.W = 3`: $\min \mu + \eta$ with $\mu = (\sqrt{\beta})^{-1}$ represents the scaling of a set (or directions) defined by $\mathcal{X}_0$. $\mathcal{X}_0$ has to be specified with `optsat.Xi0`.Otherwise, default is used.

- `optsat.W = 4`: $\min -\text{geomean}(W) + \eta$
- `optsat.W = 0`: No optimization on the set.

`optsat.Xi0` has to be defined with the option `optsat.W = 3`. It is a matrix of dimension (n_{tot},n_r) of which columns are the n_r vertices which define a desired allowed set of initial conditions or directions in which increasing $\mathcal{E}(W^{-1}, \eta)$. If `optsat.Xi0` is provided but with `optsat.W` $\neq$ `3`, it is not used.

Remark 13 *In the global context,* `optsat.W` *is not used. Either the problem is solved as a simple feasibility problem or an optimization program may be built thanks to the disturbance (see subsection 5.4).*

5.4 Optimization of the size of the admissible disturbance

When the problem includes the presence of additive disturbance vector w, external stability problems are addressed. For a given bound on the disturbance δ (set in `sysD.delta`), the optimization is related to the size of $\mathcal{E}(W^{-1}, \eta)$. On the other hand, another issue may to find the largest admissible disturbance for which the trajectories of the closed-loop system remain bounded. This criterion is set by using the element `optsat.D`. Three different cases have been programmed:

- `optsat.D = 0`: No optimization of the disturbance (Default)
- `optsat.D = 1`: $\min \delta$. Even if *a priori* defined, δ becomes a decision variable.
- `optsat.D = 2`: $\min \gamma$. Optimization of the $\mathcal{L}_2$-gain γ from $w(t)$ to $z(t)$. It is recommended to set `sys.D` $= I_q$ in that case to actually manipulate the $\mathcal{L}_2$-gain. This case is only valid in design problems (static, dynamic or anti-windup). Alternatively, one can also use this option to compute the $\mathcal{L}_2$-gain of the problem in parallel of the feasibility checking of the global stability analysis problem.

Remark 14 *The case* `optsat.D = 1` *is not used in the anti-windup function. Only energy-bounded problem is computed, and with only the optimization of γ for given bound on the disturbance δ.*

6 Conclusion

You are ready now. You will find all the instructions to use a function in its help. I confess that it must remain many bugs in this first iteration of the toolbox. Please use the toolbox and return us all the comments you have so that we can correct and improve it.

References

[1] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization.* MPS-SIAM Series on Optimization. SIAM, Philadelphia, 2001.

[2] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory.* SIAM Studies in Applied Mathematics, 1994.

[3] J. Löfberg. *YALMIP: A Toolbox for Modeling and Optimization in MATLAB.* `http://users.isy.liu.se/johanl/yalmip/`, 2004.

[4] J. Löfberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[5] D. Peaucelle. *RoMulOC a YALMIP-MATLAB based Robust Multi Objective Control Toolbox.* `http://www.laas.fr/OLOCEP/romuloc`, 2005.

[6] S. Tarbouriech, G. Garcia, J.M. Gomes da Silva Jr., and I. Queinnec. *Stability and Stabilization of Linear Systems with Saturating Actuators.* Springer, London (UK), 2011.