Czech Technical University, Prague, Czech Republic

April, 4th 2002

LMI Solvers and Interfaces

A new tool: SeDuMi Interface

by Dimitri Peaucelle & Denis Arzelier & Didier Henrion

Laboratoire d'**A**nalyse et d'**A**rchitecture des **S**ystèmes du **C.N.R.S.**
Toulouse, FRANCE

Motivation

- ✗ LMIs are widely used in Automatic Control theory.
- ✗ Existing solvers are quite slow and limited in size.
- ✗ Is every Semi-Definite Programming tool suitable to solve LMI problems?
- ✗ Build together a tool specifically adapted to the needs.
- ✗ Develop a tool-box that illustrates theoretical results.

Outline

Outline

1 – Select a solver and specify the interface

- Selected solver: SeDuMi.
- Necessity to build an Interface.
- No graphics, buttons and windows.

Selected solver: SeDuMi

SeDuMi is neither ideal nor ultimate but

- ✓ low computation time compared to other solvers,
- ✓ data given in sparse format (reduced memory burden),
- ✓ solves problems of small to medium size dimensions,
- ✓ works with both equality and inequality constraints,
- ✓ data and variables can be complex or real valued,
- ✓ free software,
- ✓ new developments in progress and potentialities.

Canonical definition of LMI constraints

Each solver takes a different format for defining the data in LMI expressions.

Example [Boyd et al. 1994]

$$p^{opt.} = min \quad c^T x \qquad s.t. \quad F_0 + \sum_{i=1}^{m} x_i F_i \geq \mathbb{0}$$

$x^T = (x_1, \ldots, x_m)$: vector of decision variables,

$F_0, F_1, \ldots, F_m$: data matrices.

Canonical definition of LMI constraints

Canonical formulation applied to the Lyapunov inequality:

$$A^T P + PA < \mathbb{0} \qquad A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \qquad P = \begin{bmatrix} p_1 & p_2 \\ p_2 & p_3 \end{bmatrix}$$

leads to three decision variables:

$$x^T = (p_1, p_2, p_3)$$

and four data matrices:

$$F_0 = \mathbb{0} \qquad F_2 = \begin{bmatrix} -2a_{21} & -a_{11}-a_{22} \\ -a_{11}-a_{22} & -2a_{12} \end{bmatrix}$$

$$F_1 = \begin{bmatrix} -2a_{11} & -a_{12} \\ -a_{12} & \mathbb{0} \end{bmatrix} \qquad F_3 = \begin{bmatrix} 0 & -a_{21} \\ -a_{21} & -2a_{22} \end{bmatrix}$$

Canonical definition of LMI constraints

Specifications for the interface

- Help the user as much as possible to define structured LMIs:
  - Structured matrix variables $\neq$ scalar variables.
  - Various linear terms (including Kronecker products).
  - Classical optimisation objectives (such as the trace of a matrix).
- Conversion to the canonical formulation:
  - Fast (faster than the computation time of the solver).
  - Low memory burden (sparse format).
  - Allow any evolution (contribution of users, eg. block partitioning).

Not a GUI tool

Existing GUI tools LMITOOL, sdpsol, LMI Control Toolbox.

- More or less easy to use.
- Very slow conversion to the canonical form.
- Limitations for programming problem-dependent LMIs.

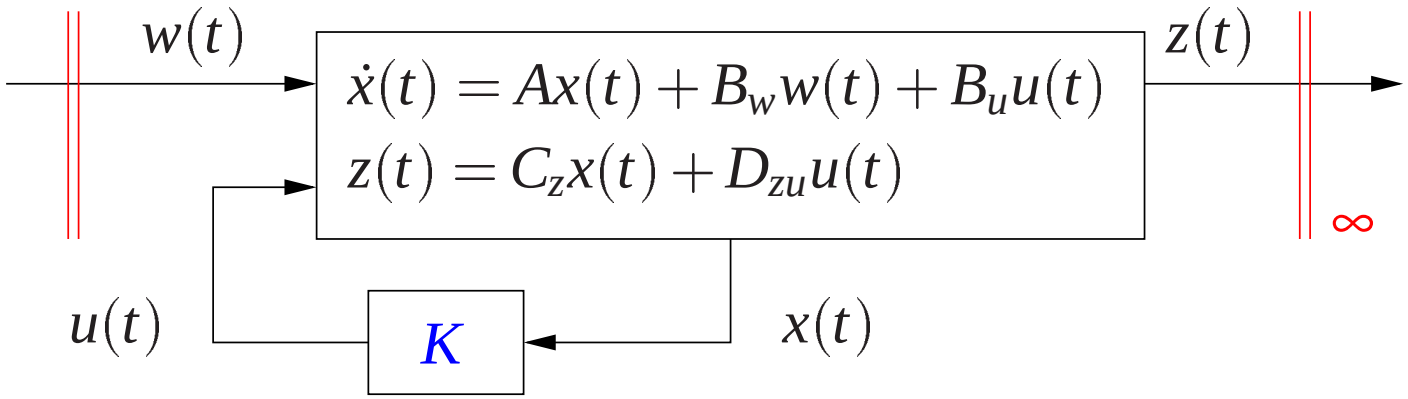
m-file based interface LMI Toolbox, YALMIP, LMIlab translator.

- Declare / modify an LMI problem
  → initialise, declare variables / constraints / objective
- Solve the optimisation problem with the linear constraints.
- Analyse the computed solution.

2 – Description of SeDuMi Interface

- A linear constrained optimisation problem.
- Initialise the problem.
- Define the variables.
- Define the constraints.
- Define the objective.
- Solve the problem with SeDuMi.
- Analyse the computed result.

Tutorial problem : H_∞ optimal state feedback



A convex formulation of this problem writes as:

$$K^{opt.} = YQ^{-1} \quad \arg\min\ \gamma \quad : \quad \begin{cases} Q = Q^T > \mathbb{0} \\ \begin{bmatrix} AQ + QA^T + B_u Y + Y^T B_u^T + B_w B_w^T & QC_z^T + Y^T D_{zu}^T \\ C_z Q + D_{zu} Y & -\gamma^2 \mathbb{1} \end{bmatrix} < \mathbb{0} \end{cases}$$

Tutorial problem : H_∞ optimal state feedback

The problem is entirely defined for SEDUMI INTERFACE as:

var.	$Q = Q^T \in \mathbb{R}^{n\times n}$, $Y \in \mathbb{R}^{m\times n}$, $\gamma^2 \in \mathbb{R}$
ineq.	$\mathbb{0} < Q$ $\text{sym}\left\{ \begin{bmatrix} A \\ C_z \end{bmatrix} Q E_1^T + \begin{bmatrix} B_u \\ D_{zu} \end{bmatrix} Y E_1^T \right\} + E_1 B_w B_w^T E_1^T - \gamma^2 E_2 E_2^T < \mathbb{0}$
obj.	max $-\gamma^2$

- ✓ Variables are real or complex structured matrices.
- ✓ Constraints are linear, real or complex, equalities and/or inequalities.
- ✓ Maximisation objective.

Initialise the problem in MATLAB

Optimisation problem over linear matrix constraints

→ a single object (`sdmpb object`) in SEDUMI INTERFACE.

step 1 Initialise the object and give a label.

```
>> quiz = sdmpb('Hinfty state-feedback synthesis')
optimisation problem: Hinfty state-feedback synthesis
no matrix variable
no equality constraint
no inequality constraint
no linear objective
unsolved
```

Define the matrix variables

step 2 Matrix variables : dimensions, complex or real, structure, label.

```
>> [quiz, Yindex] = sdmvar(quiz, m,  n , 'Y');
>> [quiz, Gindex] = sdmvar(quiz, 1,  1 , 'gamma^2');
>> [quiz, Qindex] = sdmvar(quiz, n, 's', 'Q=Q.''');
>> quiz
optimisation problem: Hinfty state-feedback synthesis
matrix variables:        index           name
                         1               Y
                         2               gamma^2
                         3               Q=Q.'
no equality constraint
no inequality constraint
no linear objective
unsolved
```

Define the constraints

step 3 Matrix inequalities and/or equalities: dimension, label.

→ sum of generic terms such that $LXR + (LXR)^*$ or $LR + (LR)^*$

```
>> [quiz, c1index] = sdmlmi(quiz, n, 'Q>0');

>> quiz = sdmineq(quiz, c1index, Qindex, -0.5, 1);

>> [quiz, c2index] = sdmlmi(quiz, n+p, 'Hinfty state feedback');

>> quiz = sdmineq(quiz, c2index, Qindex,   [A;Cz],         E1');
>> quiz = sdmineq(quiz, c2index,      0,    E1*Bw, 0.5*Bw'*E1');
>> quiz = sdmineq(quiz, c2index, Yindex, [Bu;Dzu],         E1');
>> quiz = sdmineq(quiz, c2index, Gindex,       E2,    -0.5*E2');
```

Define the objective

step 4 Linear objective.

→ sum of generic terms such that $e_l X e_r$ or $trace(X)$ and labels.

```
quiz = sdmobj(quiz, Gindex, -1, 1, '-gammaˆ2')
optimisation problem: Hinfty state-feedback synthesis
matrix variables:          index            name
                           1                Y
                           2                gammaˆ2
                           3                Q=Q.'
no equality constraint
inequality constraints:    index   meig     name
                           1       -Inf     Q>0
                           2       -Inf     Hinfty state feedback
maximise objective: -gammaˆ2
unsolved
```

Solve the optimisation problem

step 5 SeDuMi Computation with adapted algorithms parameters.

```
quiz = sdmsol(quiz);
...
optimisation problem: Hinfty state-feedback synthesis
matrix variables:         index            name
                          1                Y
                          2                gamma^2
                          3                Q=Q.'
no equality constraint
inequality constraints:   index   meig     name
                          1       eps      Q>0
                          2       eps      Hinfty state feedback
maximise objective: -gamma^2 = -27.3
feasible
```

Analyse the result

step 6 get the margins on each constraint.

```
inequality constraints:  index   meig    name
                         1       eps     Q>0
                         2       eps     Hinfty state feedback
maximise objective: -gammaˆ2 = -27.3
feasible
```

→ get the solution in as a matrix:

```
>> Y_opt = quiz(Yindex)
Y_opt =
  -41.2449  -53.0238   -0.8463  -25.0886
```

Outline

3 – Prospective work on SEDUMI INTERFACE

- Evolutions of SEDUMI.
- Evolution of the Interface into a platform.
- An Automatic control toolbox.

SeDuMi potentialities

Non exploited potentialities

- ✓ Linear programming $c_i - a_i^T y \geq 0$.
- ✓ Quadratic cone (second order Lorentz cone) $x_1 \geq \|x_2\|$.
- ✓ Rotated quadratic cone $x_1 x_2 \geq \frac{1}{2}\|x_3\|^2, x_1 + x_2 \geq 0$.

Future functionalities

- ✗ Initialisation with some feasible point.
- ✗ Pre-conditioning of problems.
- ✗ Improve speed and precision.

Evolution of the SeDuMi Interface

Interface functions

- ✗ Block partitioning of linear matrix constraints.
- ✗ Increased speed of the conversion to the canonical form.
- ✗ Call for examples and new needs.

Evolution into a platform

- ✗ A unique interface for various solvers (when the translation is possible).
- ✗ A toolbox with easy to use, up-to-date, pre-programmed control problems.

Example of pre-programmed problems

Analysis tool for the robust performance of LTI systems with respect to structured uncertainty.

```
>> quiz = analysis(sys, delta, 'PDLF', 'H2');
... problem built in 1.32 seconds

>> quiz = sdmsol(quiz);
... eqs m = 227, order n = 82, dim = 999, blocks = 15 ...
iter seconds digits        c*x                b*y
 24     18.5   Inf    -1.003670e-02     -1.003634e-2
```

→ about 1000 scalar variables, optimum =0.1 obtained in 18.5 seconds.

Outline

4 – Conclusion

- http://www.laas.fr/~peaucell/SeDuMiInt.html
- mailto:peaucelle@laas.fr.
- Contributions, critiques...