YALMIP: A Matlab interface to SP, MAXDET and SOCP

Johan Löfberg

Department of Electrical Engineering
Linköping University, SE-581 83 Linköping, Sweden
WWW: http://www.control.isy.liu.se
Email: johanl@isy.liu.se

January 8, 2001

Report no.: LiTH-ISY-R-2328

Technical reports from the Automatic Control group in Linköping are available by anonymous ftp at the address ftp.control.isy.liu.se. This report is contained in the compressed postscript file 2328.pdf.
YALMIP: A Matlab interface to
SP, MAXDET and SOCP

Johan Löfberg

January 8, 2001

Abstract

We introduce the MATLAB package YALMIP. The purpose of
YALMIP is to support rapid definition and solution of LMI prob-
lems without the hassle of learning the syntax in the solvers SP,
SOCP and MAXDET.

Keywords: LMI

1 Introduction

This report gives an overview of the LMI parser YALMIP (Yet Another LMI
Parser). The YALMIP package is developed with the aim to have a parser that
is completely integrated and developed in the MATLAB environment. The ad-
vantages with this approach is the platform independence and the ease to extend
the package with new features. The drawback is of course the performance loss.
The package is therefore intended for small problems with 10-100 variables and
constraints.

Details around semidefinite programming and LMIs will not be discussed in
this report. However, YALMIP is used to define semidefinite programs, typically
in the form

\[
\begin{align*}
\min & \quad c^T x - \log \det (G(x)) \\
\text{subject to} & \quad F(x) \succeq 0 \\
& \quad G(x) \succeq 0
\end{align*}
\]

Optimization problems of this kind, and special cases, can be solved with, e.g.,
SP [5, 3], SOCP [4, 2] and MAXDET [6, 1]. YALMIP serves as an interface
between MATLAB and these solvers.

YALMIP is an alternative to the freely available package SDPSOL. SDPSOL is
based on a compiled parser, and the LMIs have to be specified in files which
are fed to the program, hence making integration of LMI calculations into
MATLAB code harder. Furthermore, SDPSOL does not support SOCP.
2 System Requirements

The package uses the object oriented capabilities of MATLAB and therefore MATLAB 5 or higher is needed. The package is currently used on UNIX MATLAB 5.3.1 and PC MATLAB 5.2.1 without problems.

Since YALMIP works as an interface to the free solvers SP, SOCP and MAXDET, these packages are required (or at least those solvers needed to solve the specific problems).

3 Installation

Place and unzip the file yalmip.zip in the directory where you want to have your YALMIP directory. Make sure to put the YALMIP directory in your MATLAB path. The installation of SP, MAXDET and SOCP should be done as recommended in the manuals for these packages. Just make sure that sp, socp and maxdet can be found in your MATLAB path. For PC machines, it is extra important that the additional dlls MKL, BLAS and LAPACK (explained in the installation information for the solvers) are in the Windows path. A simple way to guarantee that everything works is to place these dlls in the same directory as the solvers. To test your system, run the command yalmip(’test’).

4 The Package

The package is based upon three classes, the sdpvar class, the lmi class and the socc class.

4.1 The sdpvar class

The class sdpvar is used to define and handle the matrix variables in the optimization problem. Some typical examples are

- sdpvar(’clear’): Should be executed before a new problem is defined.
- sdpvar(’nvars’): Counts the total number of free variables in the defined matrices.
- x = sdpvar(1,1): Defines a scalar variable.
- x = sdpvar(n,n): Defines a $n \times n$ symmetric matrix.
- x = sdpvar(n,n,’rect’): Defines a $n \times n$ full matrix.
- x = sdpvar(n,m): Defines a $n \times m$ full matrix.
- x = sdp2mat(x,sol): Converts a solution to numerical result (see solvelmi and solvesocp).

When working with sdpvar variables, most MATLAB commands are applicable (linear operators only of course), such as $x*y,A*x’,\text{trace}(x), [x;y],x(1,1)$.
4.2 The lmi class
LMIs are handled in the lmi class. Typical use is
- \( F = \text{lmi} \): Creates an empty LMI object.
- \( F = \text{lmi}('x>0') \): Creates an LMI structure with one LMI.
- \( F = \text{addlmi}(F,'[x\ \text{zeros(size}(x,1));\ \text{zeros(size}(x,1))\ x]>0') \): Adds an LMI to existing collection of LMIs. If \( x \) is a symmetric matrix, the constraint will mean positive definiteness, otherwise element-wise positivity.
- \( F = \text{addlmi}(F,'\text{eye(size}(x)) < 5*x') \): All standard operators can be used inside the definitions of the LMIs.
- \( F = \text{addlmi}(F,5*x-\text{eye(size}(x))) \): Performs faster but is equivalent to the command above.

4.3 The socc class
Second order cone constraints (SOCC) are handled in the socc class. The class is basically equivalent to the lmi class, except the introduction of the \text{norm} operator (which has to be used every SOCC).
- \( F = \text{socc} \): Creates an empty SOCC object.
- \( F = \text{socc}('\text{norm}(A*x+b)<c'*x+d') \): Creates a SOCC structure with one SOCC.
- \( S = \text{addsoccc}(S,'\text{norm}(A*[x;y])<c'*([x;\text{trace}(Y)])') \): Adds a SOCC to existing collection of SOCCs. Of course, all standard operators can be used inside the definitions of the SOCCs.

4.4 Commands for optimization
Once the variables and constraints have been defined, the following commands are used to solve the optimization problems
- \( \text{ops} = \text{sdпsettings} \): Create a structure with various parameters used in the solvers.
- \( \text{ops} = \text{sdпsettings}('\text{RelTol}',1e-3) \): Create a parameter structure with relative tolerance changed.
- \( \text{sol} = \text{solvesdp}(F,[],\text{trace}(X)) \): Solve an SDP (using SP). Default solver settings will be used.
- \( \text{sol} = \text{solvesdp}(F,[],\text{trace}(X),\text{ops}) \): Solve an SDP with specified parameter settings.
- \( \text{sol} = \text{solvesdp}(F,G,\text{trace}(X),\text{ops},[1\ 0\ 1]) \): Solve a MAXDET problem with feasible solution given.
- \( \text{sol} = \text{solvesoccc}(S,\text{trace}(x),\text{ops}) \): Solve a SOCP. Note that only one constraint structure is passed to the function \( (G = 0) \).
5 Examples

We start with a dummy example,

\[ \begin{align*}
\min & \quad t \\
\text{subject to} & \quad t > 0.
\end{align*} \]

This is solved with

\[
\text{sdpvar ('clear')}
\text{t = sdpvar(1,1);}
\text{F = lmi('t>0');}
\text{F = addlmi(F,'t<1e15');}
\text{solution = solvesdp(F,[],t);}
\text{hopefullyzero = sdp2mat(t,solution);}
\]

The additional constraint \( t < 1e15 \) is added due to problems in SP. Typically, if problems are encountered, a fix that often works is to bound all variables.

Another fix might be to alter the acceleration-parameters in \texttt{sdpsettings}.

As a second example, we study the problem of finding the largest possible positively invariant ellipsoidal domain \( x^T W x \leq 1 \), for a linear discrete-time system \( x(k+1) = Ax(k) - BLx(k) \). The control constraint \( |Lx(k)| \leq 1 \) should be satisfied in the ellipsoid. By introducing \( Y = W^{-1} \), this can be written as

\[
\begin{align*}
\max & \quad \text{det}(Y) \\
\text{subject to} & \quad \begin{pmatrix} Y & (A - BL)^T \\ (A - BL)Y & Y \end{pmatrix} \succeq 0 \\
& \quad LYLT \leq 1
\end{align*}
\]

This can be solved with

\[
\text{sdpvar ('clear')}
\text{Y = sdpvar(2,2);}
\text{F = lmi('Y*Y(A-B*L)'';(A-B*L)*Y>0');}
\text{F = addlmi(F,'L*Y*L'<1');}
\text{G = lmi('Y>0');}
\text{solution = solvesdp(F,G,[]);}
\text{W = inv(sdp2mat(Y,solution));}
\]

Notice the double transpose operator that has to be used inside a string (standard MATLAB notation).

Finally, we solve a SOCP.

\[
\begin{align*}
\min & \quad c^T x \\
\text{subject to} & \quad \|x - a\| \leq 1 \\
& \quad \|x - b\| \leq 1
\end{align*}
\]

This can be solved with
sdpvar('clear')
x = sdpvar(2,1);
F = socc('norm(x-a)<1');
F = addsocc(F,'norm(x-b)<1');
solution = solvesocc(F,c'*x);
xopt = sdp2mat(x,solution);

References


