LMI parser for NSP Software Package

J.-Ph. Chancelier * P. V. Pakshin ** S. G. Soloviev ***

* Université Paris-Est, CERMICS, École des Ponts, 6 & 8 av. B. Pascal, 77455 Marne-la-Vallée, France (e-mail: jpc@cermics.enpc.fr)
** Arzamas Polytechnic Institute of R.E. Alekseev Nizhny Novgorod State Technical University 19, Kalinina Street, Arzamas 607227, Russia (e-mail: pakshinpv@gmail.com)
*** N.I. Lobachevsky State University of Nizhny Novgorod 23, Prospekt Gagarina, Nizhny Novgorod 603950, Russia (e-mail: sergey.sol@mail.ru)

Abstract: In this paper an extension of SCIYALMIP parser to NSP platform is proposed. SCIYALMIP is a language for semidefinite programming problems description on freely distributed SCILAB/SCICOSLAB platforms. This language provides an ability to work with stand-alone SDP solvers such as CSDP, SDPA, however, many of the effective SDP solvers have been developed as MATLAB toolboxes such as SeDuMi, SDPT3 etc. and are deeply dependent on it. For this reason SCIYALMIP has less possibilities in comparison with well known YALMIP parser for MATLAB. NSP, also known as TUMBI, is a MATLAB-like Scientific Software Package developed under the GPL license. NSP emulates the MATLAB environment and in basic scientific researches. In this paper we introduce an extension of SCIYALMIP modeling language called NSPYALMIP and working under open source environment. While some of the MATLAB dependent solvers are commercial, some others, as SeDuMi (Sturm, 1999) are freely distributed (one need only a MATLAB license to work with YALMIP+SeDuMi). It follows from the last fact that it might be a good advantage to have YALMIP+SeDuMi as fully free to use software in education and in basic scientific researches. In this paper we introduce an extension of SCIYALMIP modeling language called NSPYALMIP and working under open source environment (see Chancelier et al., 2007) which is supporting SeDuMi toolbox.

NSP, SeDuMi and NSPYALMIP can be downloaded from http://cermics.enpc.fr/~jpc/nsp-tiddly/. The packages SeDuMi and NSPYALMIP are located here in Toolboxes section.

1. INTRODUCTION

Many problems in control theory can be reduced to the following optimization problem (Boyd et al., 1994)

\[
\begin{align*}
\text{minimize} & \quad f(X_1,...,X_N) \\
\text{subject to} & \quad G_i(X_1,...,X_N) = 0, \quad i = 1,...,p, \\
& \quad H_j(X_1,...,X_N) \geq 0, \quad j = 1,...,q.
\end{align*}
\]

Here \(X_1,...,X_N\) are unknown real matrices, \(f\) is a real scalar objective function of unknown matrices \(X_1,...,X_N\). \(G_i (i = 1,...,p)\) are the constraints called linear matrix equations (LME), these are matrices with entries which are affine functions of \(X_1,...,X_N\). \(H_j (j = 1,...,q)\) are the constraints called linear matrix inequalities (LMI), these are matrices with entries which are affine functions of \(X_1,...,X_N\).

Pakshin and Soloviev (2009) have proposed a new open source tool called SCIYALMIP to describe and numerically solve LMI based optimization problems. This tool is based on YALMIP ideas (see Löfberg, 2004) and represents SCILAB based language (see Campbell et al., 2006) for description and solving advanced optimization problems. This software tool can be downloaded and installed from http://spiderman-2.laas.fr/OLCERP/Sciyalmip, where also one can find detailed installation instructions.

SCIYALMIP provides a unique interface to work with the different LMI solvers such as CSDP (Borchers, 1999), SDPA (Fujisawa et al., 1995). The main obstacle to extend SCIYALMIP interface on more amount of solvers is the fact that most of the solvers have been developed as MATLAB toolboxes and work only with the MATLAB environment. While some of the MATLAB dependent solvers are commercial, some others, as SeDuMi (Sturm, 1999) are freely distributed (one need only a MATLAB licence to work with YALMIP+SeDuMi). It follows from the last fact that it might be a good advantage to have YALMIP+SeDuMi as fully free to use software in education and in basic scientific researches. In this paper we introduce an extension of SCIYALMIP modeling language called NSPYALMIP and working under open source environment (see Chancelier et al., 2007) which is supporting SeDuMi toolbox.

NSP, SeDuMi and NSPYALMIP can be downloaded from http://cermics.enpc.fr/~jpc/nsp-tiddly/. The packages SeDuMi and NSPYALMIP are located here in Toolboxes section.

2. NSP

NSP is a Matlab-like Scientific Software Package developed under the GPL license (Chancelier et al., 2007). It is a high-level programming language which can be used as a scripting language which gives an easy access to efficient numerical routines. It can be used as an interactive computing environment or as a programming language. It

* This work is supported in part by Russian Foundation for Basic Research under grants 10-08-00843, 11-01-97025.
supports imperative programming and features a dynamic type system and automatic memory management. It contains internally a class system with simple inheritance and interface implementation, this class system is visible at the NSP programming level but not extendable at the NSP level. When used as an interactive computing environment, it comes with online help facilities and an easy access to GUI facilities and graphics.

A large set of libraries are available and it is moreover easy to implement new functionalities. It requires to write some wrapper code also called interfaces to give glue code between the external library and NSP internal data. The interface mechanism can be either static or dynamic. Using dynamic functionalities we are able to build toolboxes. NSP emulates the Matlab mex library which is a way to write wrapper code for interfaces. This facility was used to obtain a SeDuMi port from Matlab to NSP.

NSP shares many paradigms with other Matlab-like Scientific Softwares as for example: Matlab, Octave, ScilabGtk (Campbell et al., 2006) and also with scripting languages such as Python for instance.

NSP toolboxes were used and developed for this work. The first one is the SeDuMi port to NSP the second one is the Yalmip port to NSP. For this last port, new NSP objects were created as for example \texttt{lim}, \texttt{constraint} and \texttt{sdpvar} objects.

Figure 1 presents the NSP main screen.

![NSP main screen](image)

Fig. 1. NSP main screen

In NSP to draw two dimensional and three dimensional graphics one may use \texttt{plot2d} and \texttt{plot3d} functions, here is an example of source code and corresponding result:

```plaintext
t=4*%pi*(0:20)/20;
ptc=[t.*sin(t);t.*cos(t);0*ones(size(t))];
xpol=[];
ypol=[];
zpol=[];
for i=1:(size(ptc,"c")-1)
do
pt=ptc(:,i);
ptn=ptc(:,i+1);
u=ptn-pt;
likely computation
u=u/sqrt(sum(u.*u))
// trouver un vecteur ds le plan orthogonal
I=find(u==0.0);
if isempty(I) then
v=0*u;
v(I(1))=1;
else
v=[u(2);-u(1);0];
v=v/sqrt(sum(v.*v))
end
w=[u(2)*v(3)-u(3)*v(2);-(u(1)*v(3)-u(3)*v(1));u(1)*v(2)-u(2)*v(1)];
n=10;
alpha=2*%pi*(0:n)/n;
r=t(i);
pts=r*v*cos(alpha)+r*w*sin(alpha);
if i==1 then
ptg=pts+pt*ones(size(alpha));
else
ptg=ptd;
end
ptd=pts+ptn*ones(size(alpha));
xpol=[xpol,[ptg(1,1:$-1);ptd(1,1:$-1);ptd(1,2:$);ptg(1,2:$)]];
ypol=[ypol,[ptg(2,1:$-1);ptd(2,1:$-1);ptd(2,2:$);ptg(2,2:$)]];
zpol=[zpol,[ptg(3,1:$-1);ptd(3,1:$-1);ptd(3,2:$);ptg(3,2:$)]];
end
plot3d1(xpol,ypol,zpol);
```

Fig. 2. 3d surface example

Via NSP interpreter one may write scripts or functions in MATLAB likely syntax. A Google group for NSP is available at http://groups.google.fr/group/tumbi where one may submit a question or NSP further development
suggestion. Also a bug report data base is available at http://code.google.com/p/tumbi/issues/list.

In the table below we represent a few comparison tests between Matlab, Scilab and NSP. Benchmark has been taken from NSP web site (Chancelier et al., 2007).

<table>
<thead>
<tr>
<th>test</th>
<th>matlab 7.1</th>
<th>nsp</th>
<th>scilab-unix</th>
<th>scilab-win</th>
</tr>
</thead>
<tbody>
<tr>
<td>fibonnaci</td>
<td>2.153</td>
<td>1.280</td>
<td>1.460</td>
<td>1.362</td>
</tr>
<tr>
<td>subsets</td>
<td>1.883</td>
<td>1.484</td>
<td>1.430</td>
<td>2.854</td>
</tr>
<tr>
<td>irn55_rand</td>
<td>1.372</td>
<td>1.750</td>
<td>1.560</td>
<td>1.692</td>
</tr>
<tr>
<td>tri fusion</td>
<td>0.571</td>
<td>1.407</td>
<td>1.340</td>
<td>1.282</td>
</tr>
<tr>
<td>tri rapide</td>
<td>1.973</td>
<td>1.494</td>
<td>1.900</td>
<td>1.612</td>
</tr>
<tr>
<td>harmvector</td>
<td>1.702</td>
<td>1.619</td>
<td>1.230</td>
<td>1.152</td>
</tr>
<tr>
<td>harmloop</td>
<td>0.030</td>
<td>1.876</td>
<td>1.790</td>
<td>1.692</td>
</tr>
<tr>
<td>fannkuch</td>
<td>0.831</td>
<td>1.334</td>
<td>0.900</td>
<td>0.831</td>
</tr>
<tr>
<td>monسجن</td>
<td>2.023</td>
<td>1.544</td>
<td>1.850</td>
<td>1.953</td>
</tr>
<tr>
<td>cridle</td>
<td>0.981</td>
<td>0.985</td>
<td>1.210</td>
<td>1.432</td>
</tr>
<tr>
<td>make_perm</td>
<td>1.021</td>
<td>1.146</td>
<td>1.740</td>
<td>1.472</td>
</tr>
<tr>
<td>inv_perm</td>
<td>0.010</td>
<td>1.128</td>
<td>0.910</td>
<td>0.831</td>
</tr>
<tr>
<td>ftx</td>
<td>3.104</td>
<td>1.737</td>
<td>7.290</td>
<td>7.140</td>
</tr>
<tr>
<td>hmeans</td>
<td>0.481</td>
<td>1.084</td>
<td>1.310</td>
<td>2.964</td>
</tr>
<tr>
<td>simplex</td>
<td>1.833</td>
<td>1.605</td>
<td>3.090</td>
<td>4.987</td>
</tr>
<tr>
<td>loop_call_f</td>
<td>0.102</td>
<td>1.306</td>
<td>1.710</td>
<td>1.292</td>
</tr>
<tr>
<td>loop_call_p</td>
<td>0.300</td>
<td>1.039</td>
<td>0.470</td>
<td>0.461</td>
</tr>
<tr>
<td>form_vect1</td>
<td>2.664</td>
<td>0.029</td>
<td>2.110</td>
<td>4.266</td>
</tr>
<tr>
<td>form_vect2</td>
<td>2.864</td>
<td>0.815</td>
<td>1.390</td>
<td>3.024</td>
</tr>
<tr>
<td>loop1</td>
<td>0.010</td>
<td>1.407</td>
<td>1.530</td>
<td>1.242</td>
</tr>
<tr>
<td>loop2</td>
<td>0.040</td>
<td>2.115</td>
<td>1.960</td>
<td>1.853</td>
</tr>
<tr>
<td>loop3</td>
<td>0.010</td>
<td>1.644</td>
<td>1.550</td>
<td>1.452</td>
</tr>
<tr>
<td>test bool</td>
<td>3.986</td>
<td>1.238</td>
<td>2.840</td>
<td>3.175</td>
</tr>
<tr>
<td>test find</td>
<td>2.563</td>
<td>1.417</td>
<td>4.490</td>
<td>4.857</td>
</tr>
<tr>
<td>prime_factors</td>
<td>0.060</td>
<td>0.851</td>
<td>5.000</td>
<td>4.236</td>
</tr>
<tr>
<td>extraction</td>
<td>1.562</td>
<td>1.189</td>
<td>3.080</td>
<td>3.124</td>
</tr>
<tr>
<td>insertion</td>
<td>1.242</td>
<td>1.177</td>
<td>2.270</td>
<td>2.293</td>
</tr>
</tbody>
</table>

| Total time (sec) | 36.24 | 36.45 | 60.26 | 66.31 |

3. NSPYALMIP+SEDUMI INTERFACE

NSPYALMIP interface to work with ScDuMi solver in general is quite simple and is devided on the following parts:

- Create matrix variables $X_1, ..., X_N$
- Specify LMI and LME constraints (1) just as it, in the simple and natural form in terms of NSP matrix algebra syntax, no need to create additional functions and computation modules. More detailed description will be provided in the "Syntax" subsection.
- Objective function $f$ has to be provided in terms of NSP matrix algebra syntax.

3.1 Syntax

Syntax can be demonstrated on solving Riccati inequality, by reducing it to LMI problem. For example, suppose that some unknown matrix $P > 0$ of Lyapunov quadratic function is a solution to Riccati inequality

$$A^T P + PA + Q - PBR^{-1}B^T P > 0,$$

where $Q$ and $R$ are known square matrices appropriate dimensions and $R > 0$. Using Schur complement theorem (1) can be rewritten in the following equivalent form

$$\left( \begin{array}{cc} A^T P + PA + Q & PB \\ B^T P & R \end{array} \right) > 0. \quad (2)$$

LMI constraint (2) with respect to $P$ is positive definite can be easily assigned in NSPYALMIP as

$$\text{opt} = \text{sdpsettings}(\text{‘solver’, ’sedumi’});$$

$$\text{A} = \text{sdpvar}(n,n, \text{‘symmetric’});$$

$$\text{lmirest} = \text{set}(\text{A}*P+P*A+Q \quad P\ast B; \quad B\ast P \quad R) > 0;$$

$$\text{lmirest} = \text{lmirest} + \text{set}(P > 0);$$

$$\text{res} = \text{solvesdp}(\text{lmirest}, [], \text{opt})$$

where $n$ is dimension of the problem.

We omit detailed description of the interface of NSPYALMIP here, since it has been already described in more details in Pakshin and Soloviev (2009) and closely related to YALMIP interface (see Löfberg, 2004).

4. LMI PROBLEMS AND SOLUTIONS

In this section we will provide some useful examples to demonstrate NSPYALMIP potentialities.

4.1 NSPYALMIP application to robust stabilization problem via static output feedback

Consider uncertain linear system with polytopic model of uncertainty described by the following equations:

$$\dot{x}(t) = \sum_{i=1}^{\nu} \xi_i(t)(A_i x(t) + B_i u(t)), \quad y(t) = C x(t),$$

where $x(t)$ is $n_x$-dimensional state vector, $u$ is $n_u$-dimensional control vector, $A_i$ are $n_x \times n_x$ matrices; $B_i$ are $n_x \times n_u$ matrices; $\xi_i(t) \geq 0, \quad i \in \{1, ..., \nu\}$, $\sum_{i=1}^{\nu} \xi_i(t) = 1$. It is assumed that matrix $C$ has full row rank.

Consider the static output feedback control law

$$u(t) = -F y(t).$$

Since matrix $C$ has full row rank, its singular value decomposition can be written as

$$C = U S V^T = U [S_0 \quad 0] \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix},$$

where $U$ and $V$ are orthogonal matrices of compatible dimensions.

We use here a parametrization and convexification approach to output feedback stabilization problem of the system (3) (see Pakshin and Peaucelle, 2009). Based on this approach the algorithm for computation of the stabilizing gain can be formulated as follows

Algorithm 1.

1. Assign scalar $\mu > 0$ matrices $Q, R$ based on LQR reasons on the vertices of the polytope and solve the following LMI/LME problem with respect to variables $P, L_i$
where $V_2$ is defined in (5),

$$(B_i^T P + L_i) V_2 = 0,$$

is feasible, then the control law (4) with the gain matrix $F$ given by formula

$$F = R^{-1}[B_i^T P_i + L_i]C^+$$

for some $i \in \mathbb{N}$ is quadratically stabilizing one. Here superscript $+$ denotes Moore-Penrose inverse.

Now apply these results to the problem of stabilization of the angular longitudinal aircraft motion under given flight parameters uncertainty. The linearized model of this motion is given by the following equations:

$$\dot{\vartheta} = \omega_2,$$

$$\dot{\omega}_2 = -a_{mz}^\alpha \vartheta - a_{mz}^\omega \omega_2 + a_{mz}^\alpha \Theta + a_{mz}^\delta \delta,$$

$$\dot{\Theta} = -a_{mz}^\alpha \vartheta + a_{mz}^\delta \Theta,$$

where $\vartheta$ is the pitch angle, $\omega_2$ is the angular velocity, $\Theta = \vartheta - \alpha$, $\alpha$ is the angle of attack, $\delta$ is the elevator angle. The state and control vectors of the considered system are

$$x(t) = [\vartheta, \omega_2, \Theta]^T, u(t) = \delta,$$

Usually only $\vartheta$ and $\omega_2$ are available for direct measurement and we have

$$y(t) = [\vartheta, \omega_2]^T.$$
Consider some interesting exercises for students studying LMI theory and applications.

Exercise 1. Using NSPYALMIP find a positive definite solution $P = P^T$ to the Riccati inequality

$$A^T P A - P - A^T P B (B^T P B + R)^{-1} B^T P A + C^T C < 0. \tag{7}$$

Two typical solutions are as follows.

Solution 1. By the formula (Boyd et al., 1994)

$$[U \pm VZWZ^T]^{-1} U^{-1} - U^{-1} V [WU^{-1} V + Z^{-1}]^{-1} WU^{-1}$$

we obtain

$$A^T P A - P - A^T P B (B^T P B + R)^{-1} B^T P A + C^T C = A^T [P^{-1} + B R^{-1} B^T]^{-1} A - P + C^T C. \tag{9}$$

Using Schur complement theorem (Boyd et al., 1994) and taking into account (9) we can rewrite the inequality (7) as

$$\begin{pmatrix} P - Q & A^T \\ A & P^{-1} + BR^{-1} B^T \end{pmatrix} > 0.$$
problems and solvers. Based on NSP platform we support
now popular and powerful SeDuMi solver. We expect
that in the nearest future we will propose absolutely
free software with friendly interface for solution of LMI
problems in control engineering and education.

REFERENCES

Borchers, B. (1999). CSDP, a C library for
semidefinite programming. Optimization
Methods & Software, 11-2(1-4), 613–623. URL:
https://projects.coin-or.org/Csdp/.

Boyd, S., El Ghaoui, L., Feron, E., and Balakrishnan,
V. (1994). Linear Matrix Inequalities in System and
Control Theory. SIAM studies in Applied Mathematics,
Vol. 15. SIAM, Philadelphia.

Campbell, S.L., Chancelier, J.P., and Nikoukhah,
Scilab/Scicos. Springer Verlag, New York, URL:

Chancelier, J.P., Delebecque, F., Lelong, J.,
Nikoukhah, R., Pincon, B., and Quadtrat, J.P.
(2007). NSP, also know as Tumbi. URL:
http://cermics.enpc.fr/ jpc/nsp-tiddly/.

Fujisawa, K., Kojima, M., Nakata, K., and Yamashita,
M. (1995). SDPA (semidefinite programming algo-
rithm) users manual - version 6.00. Tokyo Insti-
tute of Technology. Technical Report B-308, URL:
http://sdpa.indsys.chuo-u.ac.jp/sdpa/.

in lqr theory: analysis and remedy. Int. J. Control,
47, 697–709.

control and their analytical design. Nauka, Moscow.

Löfberg, J. (2004). YALMIP: A toolbox for model-
ing and optimization in MATLAB. URL: http://
control.ee.ethz.ch/ ~joloef/ yalmip.php.

Pakshin, P. and Peaucelle, D. (2009). Stabilization and
passification of uncertain systems via static output feed-
back. In Proceedings of 18th IEEE International Con-
ference on Control Applications, 507–512. Saint Peters-
burg, Russia.

Pakshin, P.V. and Soloviev, S.G. (2009). SCIYALMIP:
a free tool for solution to semidefinite programming
problems in SCILAB. In Proceedings of the 8th IFAC
Symposium on Advances in Control education, 1–5. Ku-
mamoto, Japan. Available on IFAC Papers Online.

Sturm, J. (1999). Using SeDuMi 1.02, a MAT-
LAB toolbox for optimization over symmetric cones.
Optimization Methods and Software, 11-12, 625–653.
URL:http://sedumi.mcmaster.ca/.