2004 IEEE International Symposium on
Computer Aided Control Systems Design
Taipei, Taiwan, September 2-4, 2004

YALMIP : A toolbox for modeling and optimization in MATLAB

Johan Löfberg
Automatic Control Laboratory, ETHZ
CH-8092 Zürich, Switzerland.
loefberg@control.ee.ethz.ch

***Abstract*— The MATLAB toolbox YALMIP is introduced. It is described how YALMIP can be used to model and solve optimization problems typically occurring in systems and control theory.**

I. Introduction

Two of the most important mathematical tools introduced in control and systems theory in the last decade are probably semidefinite programming (SDP) and linear matrix inequalities (LMI). Semidefinite programming unifies a large number of control problems, ranging from the more than 100 year old classical Lyapunov theory for linear systems, modern control theory from the 60's based on the algebraic Riccati equation, and more recent developments such as $\mathcal{H}_\infty$ control in the 80's. More importantly, LMIs and SDP has led to many new results on stability analysis and synthesis for uncertain system, robust model predictive control, control of piecewise affine systems and robust system identification, just to mention a few applications.

In the same sense that we earlier agreed that a control problem was solved if the problem boiled down to a Riccati equation, as in linear quadratic control, we have now come to a point where a problem with a solution implicitly described by an SDP can be considered solved, even though there is no analytic closed-form expression of the solution. It was recognized in the 90's that SDPs are convex optimization problems that can be solved efficiently in polynomial time [13]. Hence, for a problem stated using an SDP, not only can we solve the problem but we can solve it relatively efficiently.

The large number of applications of SDP has led to an intense research and development of software for solving the optimization problems. There are today around 10 public solvers available, most of them free and easily accessible on the Internet. However, these solvers typically take the problem description in a very compact format, making immediate use of the solvers time-consuming and error prone. To overcome this, modeling languages and interfaces are needed.

This paper introduces the free MATLAB toolbox YALMIP, developed initially to model SDPs and solve these by interfacing external solvers. The toolbox makes development of optimization problems in general, and control oriented SDP problems in particular, extremely simple. Rapid prototyping of an algorithm based on SDP can be done in matter of minutes using standard MATLAB commands. In fact, learning 3 YALMIP specific commands will be enough for most users to model and solve their optimization problem.

YALMIP was initially indented for SDP and LMIs (hence the now obsolete name Yet Another LMI Parser), but has evolved substantially over the years. The most recent release, YALMIP 3, supports linear programming (LP), quadratic programming (QP), second order cone programming (SOCP), semidefinite programming, determinant maximization, mixed integer programming, posynomial geometric programming, semidefinite programs with bilinear matrix inequalities (BMI), and multiparametric linear and quadratic programming. To solve these problems, around 20 solvers are interfaced. This includes both freeware solvers such as SeDuMi [16] and SDPT3 [17], and commercial solvers as the PENNON solvers [7], LMILAB [4] and CPLEX [1]. Due to a flexible solver interface and internal format, adding new solvers, and even new problem classes, can often be done with modest effort.

YALMIP automatically detects what kind of a problem the user has defined, and selects a suitable solver based on this analysis. If no suitable solver is available, YALMIP tries to convert the problem to be able to solve it. As an example, if the user defines second order cone constraints, but no second order cone programming solver is available, YALMIP converts the constraints to LMIs and solves the problem using any installed SDP solver.

One of the most important extension in YALMIP 3 compared to earlier versions is the possibility to work with nonlinear expression. This has enabled YALMIP users to define optimization problems involving BMIs, which then can be solved using the solver PENBMI [6], the first public solver for problems with BMI constraints. These optimization problems are unfortunately extremely hard to solve, at-least globally, but since an enormous amount of problems in control theory falls into this problems class, it is our hope that YALMIP will inspire researchers to develop efficient BMI solvers and make them publicly available.

Another introduction in YALMIP 3 is an internal branch-and-bound framework. This enables YALMIP to solve integer programs for all supported convex optimization classes, i.e. mixed integer linear, quadratic, second order

0-7803-8636-1/04/$20.00 ©2004 IEEE

cone and semidefinite programs. The built-in integer solver should not be considered a competitor to any dedicated integer solver such as CPLEX [1]. However, if the user has no integer solver installed, he or she will at-least be able to solve some small integer problems using YALMIP. Moreover, there are currently no other free public solvers available for solving mixed integer second order cone and semidefinite programs.

The latest release of YALMIP has been extended to include a set of mid-level commands to facilitate advanced YALMIP programming. These commands have been used to develop scripts for moment relaxation problems [10] and sum-of-square decompositions [14], two recent approaches, based on SDP and LMIs, for solving global polynomial optimization problems. There are dedicated, more efficient, packages available for solving these problems (GloptiPoly [5] and SOSTOOLS [15]), and the inclusion of these functionalities are mainly intended to give advanced users hints on how the mid-level commands can be used. The sum-of-square functionality does however have a novel feature in that the sum-of-squares problem can be nonlinearly parameterized. In theory, this means that this function can be used, e.g., to synthesize controllers for nonlinear systems. However, the resulting optimization problem is a semidefinite program with BMIs instead of LMIs.

Although SDPs can be solved relatively efficiently using polynomial time algorithms, large-scale control problems can easily become problematic, even for state-of-the-art semidefinite solvers. To reduce computational complexity, problem-specific solvers are needed in some cases. One problem class where structure can be exploited is KYP problems, a generalization of Lyapunov inequalities. YALMIP comes with a specialized command for defining KYP constraints, and interfaces the dedicated solver KYPD [20].

Other features worth mentioning are the capabilities to work transparently with complex-valued data and constraints, easy extraction of dual variables and automatic reduction of variables in equality constrained problems.

II. Preliminaries and notation

A symmetric matrix P is denoted positive semidefinite ($P \succeq 0$) if $z^T P z \geq 0 \; \forall z$. Positive definite ($P \succ 0$) is the strict version $z^T P z > 0 \; \forall z \neq 0$. Linear matrix inequality (LMI) denotes a constraint of the form $F_0 + \sum_{i=1}^{n} F_i x_i \succeq 0$, where F_i are fixed symmetric matrices and $x \in \mathbb{R}^n$ is the decision variable. Constraints $F_0 + \sum_{i=1}^{n} F_i x_i + \sum_{j=1}^{n} \sum_{i=1}^{n} F_{ij} x_i x_j \succeq 0$ are denoted BMIs (bilinear matrix inequalities). Constraints involving either LMIs or BMIs are called semidefinite constraints. Optimization problems involving semidefinite constraints are termed semidefinite programs (SDPs).

MATLAB commands and variables will be displayed using `typewriter` font. Commands will be written on separate lines and start with >>.

III. Introduction to YALMIP

This paper does not serve as a manual to YALMIP. Nevertheless, a short introduction to the basic commands is included here to allow novel users to get started. It is assumed that the reader is familiar with MATLAB.

A. Defining decision variables

The central component in an optimization problem is the decision variables. Decision variables are represented in YALMIP by `sdpvar` objects. Using full syntax, a symmetric matrix $P \in \mathbb{R}^{n \times n}$ is defined by the following command.

```
>> P = sdpvar(n,n,'symmetric','real');
```

Square matrices are by default symmetric and real, so the same variable can be defined using only the dimension arguments.

```
>> P = sdpvar(n,n);
```

A set of standard parameterizations are predefined and can be used to create, e.g., fully parameterized matrices and various type of matrices with complex variables.

```
>> Y = sdpvar(n,n,'full');
>> X = sdpvar(n,n,'hermitian','complex');
```

Important to realize is that most standard MATLAB commands and operators can be applied to `sdpvar` variables. Hence, the following construction is valid.

```
>> X = [P P(:,1);ones(1,n) sum(sum(P))];
```

B. Defining constraints

The most commonly used constraints in YALMIP are element-wise, semidefinite and equality constraints. The command to define these is called `set`[1].

The code below generate a list of constraints, gathered in the `set` object `F`, constraining a matrix to be positive definite, having all elements positive, and with the sum of all elements being n.

```
>> P = sdpvar(n,n);
>> F = set(P > 0);
>> F = F + set(P(:) > 0);
>> F = F + set(sum(sum(P)) == n);
```

Note that the operators > and < are used to describe both semidefinite constraints and standard element-wise constraints[2]. A constraint is interpreted in terms of semidefiniteness if both left-hand side and the right-hand side of the constraint is symmetric, and as an element-wise constraints otherwise. In addition to these standard constraints, YALMIP also supports convenient definition of integrality constraints, second order cone constraints and sum-of squares constraints. Without going into details, typical notation for these constraints would be

[1]Not to be confused with the built-in function `set` in MATLAB

[2]Non-strict inequalities (>= and <=) are supported also. The reader is referred to the YALMIP manual for details.

```
>> F = set(integer(x));
>> F = set(cone(A*x+b,c'*x+d));
>> F = set(sos(1+x+x^7+x^8));
```

C. Solving optimization problems

Once all variables and constraints have been defined, the optimization problem can be solved. Let us for simplicity assume that we have matrices c, A and b and we wish to minimize $c^T x$ subject to the constraints $Ax \leq b$ and $\sum x_i = 1$. The YALMIP code to define and solve this problem is extremely intuitive and is essentially a one-to-one mapping from the mathematical description. The command `solvesdp`[3] is used for all[4] optimization problems and typically take two arguments, a set of constraints and the objective function.

```
>> x = sdpvar(length(c),1);
>> F = set(A*x < b)+set(sum(x)==1);
>> solvesdp(F,c'*x);
```

YALMIP will automatically categorize this as a linear programming problem and call a suitable solver. The optimal solution can be extracted with the command `double(x)`. A third argument can be used to guide YALMIP in the selection of solver, setting display levels and change solver specific options etc.

```
>> ops = sdpsettings('solver','glpk');
>> ops = sdpsettings(ops,'glpk.dual',0);
>> ops = sdpsettings(ops,'verbose',1);
>> solvesdp(F,c'*x,ops);
```

IV. Control related optimization using YALMIP

As stated in the introduction, YALMIP is a general purpose toolbox for modeling and solving optimization problems using MATLAB. The focus in the remainder of this paper will however be on control related problems, and we will illustrate how straightforward it is to model complex optimization problems using YALMIP.

A. Standard SDP problems in control

The perhaps most fundamental problem in control and systems theory is stability analysis using Lyapunov theory. A linear system $\dot{x} = Ax$ is asymptotically stable if and only if the real part of all eigenvalues of A are negative, or equivalently, there exist a solution P to the following Lyapunov inequality.

$$A^T P + PA \prec 0, \; P = P^T \succ 0$$

It is easy to realize that this is a linear matrix inequality, and the decision variables are the elements of the matrix P. The YALMIP implementation of this feasibility problem is given below.

```
>> P = sdpvar(n,n);
>> F = set(P > 0) + set(A'*P+P*A < 0);
>> solvesdp(F)
```

The problem above can be addressed more efficiently by solving the classical Lyapunov equation, and the benefit of semidefinite programming and YALMIP is appearant first when we try to solve more complex problems. Consider the problem of finding a common Lyapunov function for two different systems with state matrices A_1 and A_2. Furthermore, let us assume that we want to find a diagonal solution P satisfying $P \succ Q$ for some given symmetric matrix Q, and moreover, we want to find the minimum trace solution. Stating this as an SDP using YALMIP is straightforward.

```
>> P = diag(sdpvar(n,1));
>> F = set(P > Q);
>> F = F + set(A1'*P+P*A1 < 0);
>> F = F + set(A2'*P+P*A2 < 0);
>> solvesdp(F,trace(P))
```

To make things even more complicated, consider the minimum Frobenius norm ($\mathbf{Tr}PP^T$) problem. The changes in the code are minimal.

```
>> solvesdp(F,trace(P*P'))
```

YALMIP will analyze the objective function and detect that it is a convex quadratic function. Since no public SDP solver currently support quadratic objective functions, YALMIP will internally convert the problem by performing suitable epigraph formulations, and solve the problem using any available SDP solver.

B. Determinant maximization problems

As an example of a more advanced optimization problem based on semidefinite programming, let us address the problem of computing a state feedback controller $u = Lx$ together with a maximally large invariant region $\mathscr{R}$, for a saturated single-input system $\dot{x} = Ax + Bu$, $|u| \leq 1$. If we work with an ellipsoidal region $\mathscr{R} = \{x : x^T Px \leq 1$, this can addressed using semidefinite programming. To see this, let us first state the problem in a mathematical framework.

$$\begin{aligned} (A+BL)^T P + P(A+BL) &\preceq 0 \\ P &\succeq 0 \\ |Lx| &\leq 1 \forall x : x^T Px \leq 1 \end{aligned}$$

The first constraint ensures invariance of $\mathscr{R}$ ($x^T Px$ is non-increasing), the second constraint ensures that $x^T Px \leq 1$ defines an ellipsoidal region, while the last constraint ensures $|u| \leq 1$ in $\mathscr{R}$.

The constraints above are not LMIs, but a couple of standard tricks can be used to overcome this [3]. To begin with, multiply the first constraint from left and right with P^{-1} and introduce the new variables $Q = P^{-1}$ and $Y =$

[3]The name `solvesdp` was chosen since YALMIP initially only solved semidefinite programs. For compatibility issues, the name is kept even though the command now is used also for LP, QP, SOCP etc.

[4]Special higher level problems such as moment relaxations, sum-of-squares decompositions and multiparametric programs are invoked using specialized, but syntactically similar, commands.

LP^{-1}. This yields the LMI $A^TQ+QA+Y^TB^T+BY \preceq 0$. Furthermore, it can easily be shown that $\max_{x^TPx\leq 1} |Lx| = \sqrt{LP^{-1}L^T}$. Squaring this expression, inserting the definition of Q and Y, and performing a Schur complement shows that the third constraint is equivalent to $\begin{bmatrix} 1 & Y \\ Y^T & Q \end{bmatrix} \succeq 0$.

A natural objective function is the volume of the ellipsoid $\mathscr{R}$. The volume of this ellipsoid is proportional to $\det P^{-1}$, or equivalently $\det Q$. Hence, if we search for the maximal volume invariant ellipsoid, and the corresponding state feedback, we need to solve the following problem.

$$\begin{aligned} \max_{Q,Y} \quad & \det Q \\ \text{s.t. } A^TQ+QA+Y^TB^T+BY &\preceq 0 \\ Q &\succeq 0 \\ \begin{bmatrix} 1 & Y \\ Y^T & Q \end{bmatrix} &\succeq 0 \end{aligned}$$

Still, this is not a standard SDP, but we have a so called determinant maximization (MAXDET) problem [19]. Surprisingly, this class of problems can be solved with just slightly extended SDP solvers [21], or be converted to a standard SDP problem [13]. YALMIP supports the dedicated MAXDET solver [21], but can also use the construction in [13] to convert the problem to a standard SDP and solve the problem using any installed SDP solver. The following code implements the whole synthesis problem.

```
>> Q = sdpvar(n,n);
>> Y = sdpvar(1,n);
>> F = set(Q>0);
>> F = F + set(A'*Q+Q*A+Y'*B'+B*Y < 0);
>> F = F + set([1 Y;Y' Q]>0);
>> solvesdp(F,-logdet(Q));
>> P = inv(double(Q));
>> L = P*double(Y);
```

Notice the objective function `logdet(Q)`. This command is the key to declaring a MAXDET problem in YALMIP.[5]

C. Large-scale KYP-SDPs

Despite the celebrated polynomial complexity of convex SDPs, they do admittedly scale poorly when applied to large-scale system analysis and control synthesis. The reason is most often the introduction of a large Lyapunov-like matrix in the problem.

A substantial number of problems in systems and control theory can be addressed using the Kalman-Yakubovic-Popov lemma, often giving rise to SDPs of the following form.

$$\begin{aligned} \min_{x,P_i} \quad & c^Tx \\ \text{s.t. } \begin{bmatrix} A_i^TP_i+P_iA_i & P_iB_i \\ B_i^TP_i & 0 \end{bmatrix} + M_i(x) &\preceq 0, \; i=1\ldots N \end{aligned}$$

[5]This somewhat strange notation is a heritage from earlier version of YALMIP when MAXDET problems only could be solved using [21]. In this solver, the objective function is $c^Tx - \log\det Q(x)$

By exploiting connections to classical Lyapunov equalities and using duality theory for SDP, specially crafted algorithms can eliminate the computational impact of the complicating matrices P_i, and improve performance several orders of magnitude [11]. A clever implementation of the ideas in [11] can be found in the MATLAB package KYPD [20]. This special purpose solver can be used together with YALMIP.

Consider the problem of computing the worst case $\mathscr{L}_2$ gain from u to y for the the system $\dot{x}=Ax+Bu, y=Cx$. This can be written as an SDP with one KYP constraint.

$$\begin{aligned} \min_{t,P} \quad & t \\ \text{s.t. } \begin{bmatrix} A^TP+PA+C^TC & PB \\ B^TP & -tI \end{bmatrix} &\preceq 0 \end{aligned}$$

A KYP constraint can conveniently be defined in YALMIP by using the command `kyp`. The use of `kyp` not only simplifies the code but, more importantly, enables YALMIP to categorize the problem as a KYP-SDP and call KYPD if available.

```
>> P = sdpvar(n,n);
>> t = sdpvar(1,1);
>> M = blkdiag(C'*C,-t*eye(m));
>> F = set(kyp(A,B,P,M) < 0);
>> solvesdp(F,t,ops);
```

D. Non-convex semidefinite programming

Although many problems in control and systems theory can be modeled using LMIs and solved using convex semidefinite programming, even more problems turn out to be non-convex.

One of the most basic problem in control theory is static output feedback where we search for a controller $u=Ky$ and Lyapunov function x^TPx. The closed-loop Lyapunov stability condition gives the following constraints.

$$(A+BKC)^TP+P(A+BKC) \prec 0, \; P=P^T \succ 0$$

Due to products between elements in P and K, the constraint is not linear, but a bilinear matrix inequality (BMI). Optimization problems with BMIs are known to be non-convex and NP-hard in general [18], hence intractable in theory. However, there is code available to attack these problems, and it is possible to find solutions in some practical cases.

YALMIP code to define BMI problems is no more complicated than code to define standard LMIs. The following program defines stability conditions for the output feedback problem, and tries to obtain a feasible solution by calling any installed BMI solver (YALMIP can currently only interface the BMI solver PENBMI [6])

```
>> P = sdpvar(n,n);
>> K = sdpvar(m,n);
>> Ac = A+B*K*C;
>> F = set(P > 0);
>> F = F + set(Ac'*P+P*Ac < 0);
>> solvesdp(F)
```

E. Sum-of-squares decompositions

Sum-of-squares decompositions (SOS) is a recent technique for analyzing positivity of polynomials using semidefinite programming [14]. The basic idea is to decompose a polynomial $p(x)$ as a product $v(x)^T Q v(x)$ for some polynomial vector $v(x)$ and positive semidefinite matrix Q, thus trivially showing non-negativity. As an example, consider the following decomposition.

$$p(x) = 1+x+x^4 = \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix}^T \begin{pmatrix} 1 & 1/2 & -1/8 \\ 1/2 & 1/4 & 0 \\ -1/8 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix}$$

Since the matrix on the right-hand side is positive semidefinite, the polynomial on the left side is non-negative. Finding a decomposition for this example in YALMIP is done with the following code.

```
>> x = sdpvar(1,1);
>> p = 1+x+x^4;
>> F = set(sos(p));
>> solvesos(F)
```

A SOS-solver essentially derives a set of constraints that have to hold on the matrix Q, and then solves a semidefinite program to find a positive semidefinite Q satisfying all constraints. There is already user-friendly and efficient software available for these decompositions [15], but there might be cases when YALMIP is a valuable alternative.

As an example of a unique feature of the SOS-functionality in YALMIP, let us study nonlinear control synthesis for the following model (taken from [8]).

$$\begin{aligned} \dot{x}_1 &= -1.5x_1^2 - 0.5x_1^3 - x_2 \\ \dot{x}_2 &= u \end{aligned}$$

Define the vector $z = \begin{bmatrix} x_1 & x_2 & x_1^2 \end{bmatrix}^T$. Our goal is to find a stabilizing nonlinear controller $u = Kz$ and a non-quadratic Lyapunov function $V = z^T P z$, $P \succeq 0$. To prove stability, we would like to enforce $\dot{V} \leq -x^T x - u^T u$. Note that this is a polynomial inequality in x, parameterized in the decision variables P and K. In order to address this problem using SOS, we search for $P \succeq 0$ and K such that $-x^T x - u^T u - \dot{V}$ is a sum-of-squares.

For an implementation in YALMIP, we begin by defining states and decision variables.

```
>> x1 = sdpvar(1,1);x2 = sdpvar(1,1);
>> z = [x1;x2;x1^2];
>> K = sdpvar(1,3);
>> P = sdpvar(3,3);
```

Closed loop dynamics and differentiation of $V(x)$.

```
>> u = K*z;
>> f =  [-1.5*x_1^2-0.5*x_1^3-x_2;u];
>> Vdot = jacobian(V,x)*f)
```

The constraints are a mix of standard constraints and SOS constraints.[6]

```
>> F = set(P>0) + set(-25<K<25);
>> F = F + set(sos(-x'*x-u'*u-Vdot));
```

The important catch here is that the SOS-constraint is bilinearly parameterized in P and K. YALMIP will automatically realize this and formulate the SOS-decomposition using BMIs, and call a BMI solver to find the decomposition. The internal SOS-is invoked[7], using $\mathbf{Tr}P$ as objective function.

```
>> solvesos(F,trace(P));
```

If a feasible solution is found, `double(P)` and `double(K)` recovers the controller and the Lyapunov function.

It should be stressed that this functionality currently is rather academic, since it requires the solution of a non-convex semidefinite program. Hence, it is only applicable to small systems. For this to become a viable approach, vastly improved robustness and efficiency of BMI solvers is needed.

F. Multiparametric programming

Another field in control theory where optimization has had a tremendous impact is model predictive control (MPC) [12]. The basic idea in model predictive control is to pose optimal control problems *on-line* and solve these optimization problems continuously. MPC has had a substantial impact in practice, and is probably one of the most successful modern control algorithms. However, since MPC is based on optimization, it requires a considerable amount of on-line computer resources to solve the optimization problems fast enough. Hence, the impact in systems requiring fast sampling or cheap on-line computers has been limited.

The on-line optimization problems for model predictive control, applied to a constrained linear discrete-time system $x_{k+1} = Ax_k + Bu_k, y_k = Cx_k$, essentially boils down to optimization problems of the following form.

$$\begin{aligned} z^*(x) = \arg\min_z \quad & \tfrac{1}{2} z^T H z + (c + Fx)^T z + d^T x \\ Gz \quad & \leq \quad w + Ex \end{aligned}$$

The variable x is typically the current state x_k, whereas the decision variable z normally denotes a control trajectory to be optimized. The variables H, F, c, d, G, w and E are constant data depending on the model and controller tuning. Note that the problem is a QP in z for *fixed* x.

The function $z^*(x)$, i.e., the optimal future control trajectory as function of the state, can be shown to piecewise affine. Hence, if we can find this function off-line, the on-line effort essentially reduce to a function evaluation. The concept of explicit solutions to parameterized optimization problems is called multiparametric programming.

[6] Constraints on K are added for numerical reasons

[7] YALMIP will automatically detect parametric decision variables.

An introduction to this field, with a bias towards control applications, can be found in [2].

Efficient algorithms to calculate explicit solutions to multiparametric LPs and QPs have recently been publicly available in the MATLAB toolbox MPT [9]. This toolbox is interfaced in YALMIP, enabling extremely convenient definition and solution of multiparametric problems.

Giving a detailed description of MPC is beyond the scope of this paper, so let us just state and solve a typical MPC problem, for a given state x_k, using YALMIP.

```
>> U = sdpvar(N,1);
>> Y = T*x_k+S*U;
>> F = set(-1 < U < 1) + set(Y > 0);
>> sol = solvesdp(F,Y'*Y+U'*U);
```

The variable U is the decision variable and describes the future control trajectory. The model of the dynamic system is captured in the matrices T and S, and gives the prediction of future outputs Y, given the current state x_k, and the control sequence U. The input U is constrained and the output Y has to be positive. The performance measure is the standard unweighted quadratic cost, so the optimization problem solved when `solvesdp` is called will be a QP.

The changes in the YALMIP code above to calculate an explicit solution $U^*(x_k)$ instead is minimal. To begin with, we define x_k as an `sdpvar` variable. The explicit solution can only be calculated over a bounded set, so we constrain x_k to the region $-10 \leq x_k \leq 10$. Instead of using the command `solvesdp`, we invoke the function `solvemp`, with one additional argument to define the so called parametric variable, in our case x_k. The function `solvemp` serves as an interface to MPT and returns a MATLAB object defining the function $U^*(x)$.

```
>> U = sdpvar(N,1);
>> x_k = sdpvar(n,1);
>> Y = T*x_k+S*U;
>> F = set(-10 < x_k < 10);
>> F = F + set(-1 < U < 1) + set(Y > 0);
>> sol = solvemp(F,Y'*Y+U'*U,x_k);
```

Of-course, explicit solutions can be applied also in other fields than MPC. It should however be kept in mind that the currently available algorithms to calculate explicit LP and QP solutions are limited to problems with a few number of parametric variables, typically 5 or less.

V. Conclusion and Future Perspectives

We hope that this paper has convinced the reader that YALMIP is a powerful tool for optimization based algorithm development in MATLAB. The reader is encouraged to download YALMIP and experiment.

YALMIP has grown substantially since its first public release in early 2001, but is still evolving. The goal is to simplify the whole process of using optimization as an engineering tool, bring state-of-the-art solvers and methods to the casual MATLAB user, and, ultimately, deliver a general framework for control relevant optimization in MATLAB.

References

[1] CPLEX. `http://www.ilog.com/products/cplex`.

[2] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

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

[4] P. Gahinet and A. Nemirovskii. *LMI Control Toolbox: the LMI Lab*. The MathWorks, Inc, 1995.

[5] D. Henrion and J. B. Lasserre. Gloptipoly: Global optimization over polynomials with Matlab and SeDuMi. *ACM Transactions on Mathematical Software*, 29(2):165–194, 2003.

[6] M. Kočvara and M. Stingl. PENBMI. Available at `http://www.penopt.com`.

[7] M. Kočvara and M. Stingl. PENNON: a code for convex nonlinear and semidefinite programming. *Optimization Methods and Software*, 18(3):317–333, 2003.

[8] M. Krstić and P. Kokotović. Lean backstepping design for a jet engine compressor model. In *Proceedings of the 4th IEEE Conference on Control Applications*, pages 1047–1052, Albany, New York, 1995.

[9] M. Kvasnica, P. Grieder, M. Baotic, and M. Morari. *Multi-Parametric Toolbox (MPT)*. Automatic Control Laboratory, ETHZ, Zürich, 2003.

[10] J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

[11] L.Vandenberghe, R.V. Balakrishnan, R. Wallin, and A. Hansson. On the implementation of primal-dual interior-point methods for semidefinite programming problems derived from the kyp lemma. In *Proceedings of the IEEE Conference on Decision and Control*, volume 5, pages 4658–4663, Maui, HI, USA, December 9-12 2003. IEEE.

[12] J. M. Maciejowski. *Predictive Control with constraints*. Prentice Hall, 2002.

[13] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM Studies in Applied Mathematics. SIAM, Philadelphia, Pennsylvania, 1993.

[14] P. A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming Ser. B*, 96(2):293–320, 2003.

[15] S. Prajna, A. Papachristodoulou, and P. A. Parrilo. SOSTOOLS: a general purpose sum of squares programming solver. In *Proceedings of the 41th Conference on Decision & Control*, Las Vegas, USA, 2002.

[16] J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11-12(1-4):625–653, 1999.

[17] K. C. Toh, M. J. Todd, and R. H. Tütüncü. SDPT3 - a Matlab software package for semidefinite programming, version 2.1. *Optimization Methods and Software*, 11-12(1-4):545–581, 1999.

[18] O. Toker and H. Özbay. On NP-hardness of solving bilinear matrix inequalities and simultaneous stabilization with static output feedback. In *Proceedings of the American Control Conference*, Seattle, Washington, USA, 1995.

[19] L. Vandenberghe, S. Boyd, and S.-P. Wu. Determinant maximization with linear matrix inequality constraints. *SIAM Journal on Matrix Analysis and Applications*, 19(2):499–533, 1998.

[20] R. Wallin and A. Hansson. KYPD: a solver for semidefinite programs derived from the Kalman-Yakubovich-Popov lemma. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[21] S.-P. Wu, L. Vandenberghe, and S. Boyd. *MAXDET-Software for Determinant Maximization Problems-User's Guide*. Information Systems Laboratory, Electrical Engineering Department, Stanford University, 1996.