Fast Algorithms for SDPs derived from the Kalman-Yakubovich-Popov Lemma

Venkataramanan (Ragu) Balakrishnan
School of ECE, Purdue University

8 September 2003
European Union RTN Summer School on Multi-Agent Control
Hamilton Institute

Fast Algorithms for SDPs derived from the Kalman-Yakubovich-Popov Lemma

Venkataramanan (Ragu) Balakrishnan
School of ECE, Purdue University

8 September 2003
European Union RTN Summer School on Multi-Agent Control
Hamilton Institute

Joint work with Lieven Vandenberghe, UCLA
Anders Hansson and Ragnar Wallin, Linkoping University

Outline

- A brief introduction to Semidefinite Programming (SDP)

Outline

- A brief introduction to Semidefinite Programming (SDP)
- Focus: LMIs from the Kalman-Yakubovich-Popov Lemma

Outline

- A brief introduction to Semidefinite Programming (SDP)
- Focus: LMIs from the Kalman-Yakubovich-Popov Lemma
- Fast algorithms for SDPs from KYP Lemma

Semidefinite Programming (SDP)

Convex optimization of the form:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & F_0 + x_1 F_1 + \cdots + x_p F_p \succeq 0 \end{array}$$

$F_0, F_1, \ldots, F_p$ are given symmetric matrices, c is a vector, x is the vector of optimization variables

Semidefinite Programming (SDP)

Convex optimization of the form:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & F_0 + x_1 F_1 + \cdots + x_p F_p \succeq 0 \end{array}$$

$F_0, F_1, \ldots, F_p$ are given symmetric matrices, c is a vector, x is the vector of optimization variables

- $F(x) = F_0 + x_1 F_1 + \cdots + x_p F_p \succeq 0$ called an “LMI”
- $F \succeq 0$ means F is positive semidefinite, that is $u^T F u \succeq 0$ for all vectors u
- LMIs are nonlinear, but *convex* constraints:
  If $F(x) \succeq 0$ and $F(y) \succeq 0$, then

$$F(\lambda x + (1-\lambda)y) = \lambda F(x) + (1-\lambda)F(y) \succeq 0 \text{ for all } \lambda \in [0,1]$$

SDP vs. LP

SDP: minimize $c^T x$

subject to $F_0 + x_1 F_1 + \cdots + x_p F_p \succeq 0$

$F_0, F_1, \ldots, F_p$ are given symmetric matrices, c is a vector, x is the vector of optimization variables

LP: minimize $c^T x$

subject to $a_i^T x \leq b_i, i = 1, \ldots, N$

- Same linear objective
- Linear matrix inequality constraint instead of linear scalar inequalities

More on LMIs

- Matrices as variables:
  Example: Lyapunov inequality

$$A^T P + PA \prec 0$$

  A is given, $P = P^T$ is the variable

  Can write it as an LMI in the entries of P

  Better to leave LMIs in a condensed form

  - ⋆ saves notation
  - ⋆ leads to more efficient computation

More on LMIs

- Matrices as variables
- Multiple LMIs $F^{(1)}(x) \succeq 0, \ldots, F^{(N)}(x) \succeq 0$ same as single LMI

$$\textbf{diag}(F^{(1)}(x), \ldots, F^{(N)}(x)) \succeq 0$$

LMI examples

Many standard constraints can be written as LMIs

- Linear constraints $Ax + b > 0$ (componentwise)

  Can be rewritten as an LMI using diagonal matrices

LMI examples

Many standard constraints can be written as LMIs

- Linear constraints
- Quadratic constraints:
  Inequality $(Ax+b)^T(Ax+b)+c^Tx+d<0$ is equivalent to the LMI

$$\begin{bmatrix} I & Ax+b \\ (Ax+b)^T & -(c^Tx+d) \end{bmatrix} \succ 0$$

LMI examples

Many standard constraints can be written as LMIs

- Linear constraints
- Quadratic constraints
- Trace constraints:
  Inequality $P = P^T$, $A^T P + PA \prec 0$, **Tr**$P \leq 1$ is an LMI

LMI examples

Many standard constraints can be written as LMIs

- Linear constraints
- Quadratic constraints
- Trace constraints
- Norm constraints:
  Inequality $\sigma_{\max}(A) < 1$ is equivalent to LMI

$$\begin{bmatrix} I & A \\ A^T & I \end{bmatrix} \succ 0$$

LMI examples

Many standard constraints can be written as LMIs

- Linear constraints
- Quadratic constraints
- Trace constraints
- Norm constraints
- ... mixtures of these constraints and many more

SDP applications

- Systems and control (quite well-known)

SDP applications

- Systems and control (quite well-known)
- Circuit design

SDP applications

- Systems and control (quite well-known)
- Circuit design
- Nonconvex optimization

SDP applications

- Systems and control (quite well-known)
- Circuit design
- Nonconvex optimization
- ... many others

Outline

- A brief introduction to Semidefinite Programming (SDP)
- **Focus: LMIs from the Kalman-Yakubovich-Popov Lemma**
- Fast algorithms for SDPs from KYP Lemma

Kalman-Yakubovich-Popov lemma

Frequency-domain inequality, rational in frequency ω, and affine in a design vector x, expressed as

$$\left[\begin{array}{c}(j\omega I - A)^{-1}B \\ I\end{array}\right]^* \left(\sum_{i=1}^{p} x_i M_i - N\right) \left[\begin{array}{c}(j\omega I - A)^{-1}B \\ I\end{array}\right] \succeq 0$$

Kalman-Yakubovich-Popov lemma

If (A, B) is controllable, then

$$\left[\begin{array}{c}(j\omega I - A)^{-1}B \\ I\end{array}\right]^* (\sum_{i=1}^{p} x_i M_i - N) \left[\begin{array}{c}(j\omega I - A)^{-1}B \\ I\end{array}\right] \succeq 0$$

hold for all $\omega \in \mathbf{R}$

Kalman-Yakubovich-Popov lemma

If (A, B) is controllable, then

$$\begin{bmatrix} (j\omega I - A)^{-1}B \\ I \end{bmatrix}^* (\sum_{i=1}^{p} x_i M_i - N) \begin{bmatrix} (j\omega I - A)^{-1}B \\ I \end{bmatrix} \succeq 0$$

hold for all $\omega \in \mathbf{R}$

$$\Longleftrightarrow$$

$$\begin{bmatrix} AP + PA & PB \\ B^T P & 0 \end{bmatrix} + \sum_{i=1}^{p} x_i M_i - N \succeq 0$$

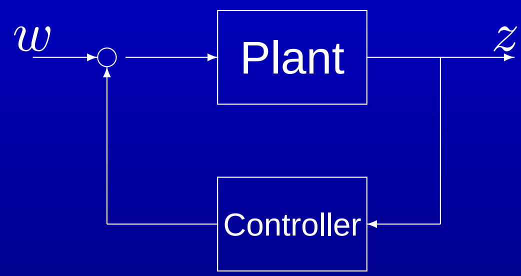
is feasible (an LMI with variables P, x)

KYP Lemma consequences

- Semi-infinite frequency domain inequality is exactly equivalent to LMI (no sampling)
- P serves as an auxiliary variable
- Of enormous importance for systems, control, and signal processing

KYP LMI applications

- **Linear system analysis and design:**



  - ⋆ Problem: Design LTI controller for LTI plant
  - ⋆ Constraints specified as frequency domain inequalities on TF from w to z
  - ⋆ Youla parametrization used to express TF from w to z

$$T(j\omega, x) = T_1(j\omega) + T_2(j\omega)\left(\sum_{i=1}^{p} x_i Q_i(j\omega)\right) T_3(j\omega),$$

  - ⋆ KYP Lemma used to obtain LMIs in variable x

KYP LMI applications

- **Linear system analysis and design**
- **Digital filter design:**
  - ⋆ An FIR or more general filter design problem: Find x such that

$$H(e^{j\theta}, x) = \sum_{i=0}^{p-1} x_i H_i(e^{j\theta})$$

    satisfies frequency-domain constraints (i.e., for all $\theta \in [0, 2\pi]$)
  - ⋆ KYP Lemma used to obtain LMIs in variable x

KYP LMI applications

- **Linear system analysis and design**
- **Digital filter design**
- **Robust control analysis:**
  - ⋆ Stability of interconnected systems via passivity or small-gain analysis
  - ⋆ Techniques that take advantage of uncertainty structure/nature
  - ⋆ Performance analysis via Lyapunov functions

KYP SDP

Focus on:

$$\begin{array}{ll} \text{minimize} & c^T x + \mathbf{Tr}(CP) \\ \text{subject to} & \begin{bmatrix} A^T P + PA & PB \\ B^T P & 0 \end{bmatrix} + \sum_{i=1}^{p} x_i M_i \succeq N \end{array}$$

where $c \in \mathbf{R}^p, C \in \mathbf{S}^n, A \in \mathbf{R}^{n\times n}, B \in \mathbf{R}^{n\times m}, M_i \in \mathbf{S}^{n+m}, N \in \mathbf{S}^{n+m}$

KYP SDP

Focus on:

$$\begin{array}{ll} \text{minimize} & c^T x + \mathbf{Tr}(CP) \\ \text{subject to} & \begin{bmatrix} A^T P + PA & PB \\ B^T P & 0 \end{bmatrix} + \sum_{i=1}^p x_i M_i \succeq N \end{array}$$

where $c \in \mathbf{R}^p, C \in \mathbf{S}^n, A \in \mathbf{R}^{n \times n}, B \in \mathbf{R}^{n \times m}, M_i \in \mathbf{S}^{n+m}, N \in \mathbf{S}^{n+m}$

(Extension to multiple LMIs in multiple variables straightforward)

$$\begin{array}{ll} \text{minimize} & c^T x + \sum_{k=1}^K \mathbf{Tr}(C_k P_k) \\ \text{subject to} & \begin{bmatrix} A_k^T P_k + P_k A_k & P_k B_k \\ B_k^T P_k & 0 \end{bmatrix} + \sum_{i=1}^p x_i M_{ki} \succeq N_k, \quad k = 1, \ldots, K. \end{array}$$

Numerical solution of SDPs

All SDPs are convex optimization problems:

- Generic algorithms will work in polynomial-time
- Matlab “LMI Control Toolbox” available
- Moderate size problems solved quite easily

But...

Numerical solution of SDPs

All SDPs are convex optimization problems:

- Generic algorithms will work in polynomial-time
- Matlab "LMI Control Toolbox" available
- Moderate size problems solved quite easily

But...

KYP SDPs tend to be of very large scale

Large problem sizes due to:

- underlying problems themselves
- auxiliary variable P

Rest of the talk on **efficient solution of KYP SDPs using convex duality**

Convex duality

Rewrite SDP as

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & F_0 + x_1 F_1 + \cdots + x_p F_p - S = 0 \\ & S \succeq 0 \end{array}$$

Convex duality

Primal SDP

minimize $c^T x$
subject to $F_0 + x_1 F_1 + \cdots + x_p F_p - S = 0$
$S \succeq 0$

Dual SDP

maximize $-\mathbf{Tr} F_0 Z$
subject to $Z \succeq 0$
$\mathbf{Tr} F_i Z = c_i, \ i = 1, \ldots, m$

Convex duality

Primal SDP

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & F_0 + x_1 F_1 + \cdots + x_p F_p - S = 0 \\ & S \succeq 0 \end{array}$$

Dual SDP

$$\begin{array}{ll} \text{maximize} & -\mathbf{Tr} F_0 Z \\ \text{subject to} & Z \succeq 0 \\ & \mathbf{Tr} F_i Z = c_i, \; i = 1, \ldots, m \end{array}$$

- If Z is dual feasible, then $-\mathbf{Tr} F_0 Z \leq p^*$
- If x is primal feasible, then $c^T x \geq d^*$
- Under mild conditions, $p^* = d^*$
- At optimum, $S_{\rm opt} Z_{\rm opt} = F(x_{\rm opt}) Z_{\rm opt} = 0$

Primal-dual algorithms

Solve primal and dual problem together:

$$\begin{array}{ll} \text{minimize} & c^T x + \mathbf{Tr} F_0 Z \\ \text{subject to} & F_0 + x_1 F_1 + \cdots + x_p F_p - S = 0 \\ & S \succeq 0, \ Z \succeq 0 \\ & \mathbf{Tr} F_i Z = c_i, \ i = 1, \ldots, m \end{array}$$

Primal-dual algorithms

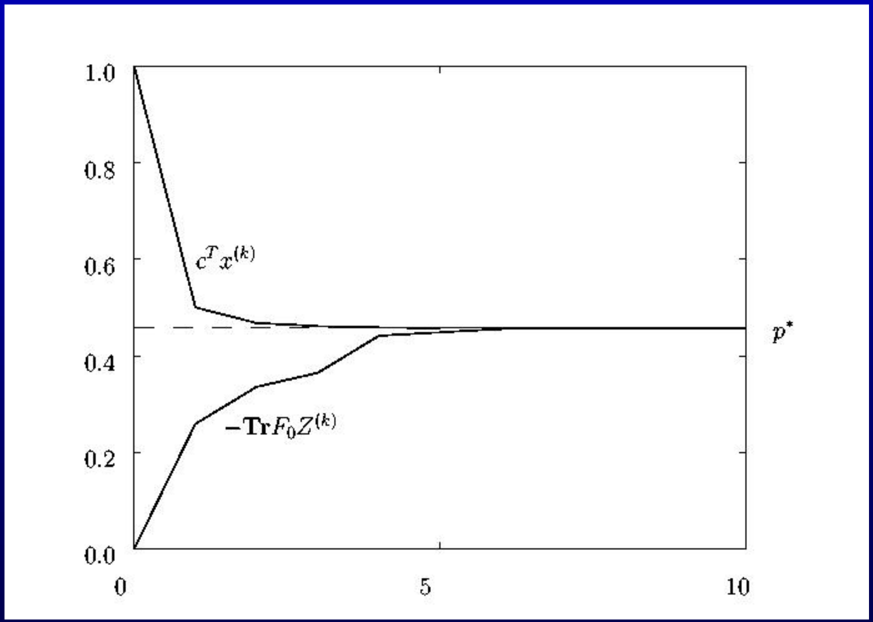
Solve primal and dual problem together:

$$
\begin{array}{ll}
\text{minimize} & c^T x + \mathbf{Tr} F_0 Z \quad (= \mathbf{Tr} SZ) \\
\text{subject to} & F_0 + x_1 F_1 + \cdots + x_p F_p - S = 0 \\
& S \succeq 0, \ Z \succeq 0 \\
& \mathbf{Tr} F_i Z = c_i, \ i = 1, \ldots, m
\end{array}
$$

(Optimal value is zero!)

Why primal-dual algorithms?

- At every iteration, we have upper and lower bounds, thus guaranteed accuracy



- Early termination possible
- Other advantages at algorithmic level

Primal-dual algorithm outline

For simplicity, suppose we have a feasible point, i.e., x, Z and S s.t.

$$F_0 + x_1 F_1 + \cdots + x_p F_p - S = 0$$
$$S \succeq 0, Z \succeq 0$$
$$\mathbf{Tr} F_i Z = c_i, \; i = 1, \ldots, m$$

(More general case, with infeasible starting points, essentially the same)

Primal-dual algorithm outline

For simplicity, suppose we have a feasible point, i.e., x, Z and S s.t.

$$F_0 + x_1 F_1 + \cdots + x_p F_p - S = 0$$
$$S \succeq 0, Z \succeq 0$$
$$\mathbf{Tr} F_i Z = c_i, \; i = 1, \ldots, m$$

At each iteration:

- Compute product SZ. If it is “small”, stop
- Otherwise, take steps ΔS, ΔZ, and Δx such that

$$\left.\begin{array}{l} \Delta x_1 F_1 + \cdots + \Delta x_p F_p - \Delta S = 0 \\ \mathbf{Tr} F_i \Delta Z = 0, \; i = 1, \ldots, m \\ S + \Delta S \succeq 0, \; Z + \Delta Z \succeq 0 \end{array}\right\} \quad \text{(maintain feasibility)}$$

Primal-dual algorithm outline

For simplicity, suppose we have a feasible point, i.e., x, Z and S s.t.

$$\begin{array}{c} F_0 + x_1 F_1 + \cdots + x_p F_p - S = 0 \\ S \succeq 0, Z \succeq 0 \\ \mathbf{Tr} F_i Z = c_i, \; i = 1, \ldots, m \end{array}$$

At each iteration:

- Compute product SZ. If it is "small", stop
- Otherwise, take steps ΔS, ΔZ, and Δx such that

$$\left.\begin{array}{l} \Delta x_1 F_1 + \cdots + \Delta x_p F_p - \Delta S = 0 \\ \mathbf{Tr} F_i \Delta Z = 0, \; i = 1, \ldots, m \\ S + \Delta S \succeq 0, \; Z + \Delta Z \succeq 0 \end{array}\right\} \quad \text{(maintain feasibility)}$$

$(S + \Delta S)(Z + \Delta Z)$ is made "smaller" (address objective)

Solving search equations

1. $\Delta x_1 F_1 + \cdots + \Delta x_p F_p - \Delta S = 0$
2. **Tr**$F_i \Delta Z = 0, \; i = 1, \ldots, m$
3. $(S + \Delta S)(Z + \Delta Z)$ is made "smaller"
4. $S + \Delta S \succeq 0, \; Z + \Delta Z \succeq 0$

Solving search equations

1. $\Delta x_1 F_1 + \cdots + \Delta x_p F_p - \Delta S = 0$
2. **Tr**$F_i \Delta Z = 0, \; i = 1, \ldots, m$
3. $(S + \Delta S)(Z + \Delta Z)$ is made “smaller”
4. $S + \Delta S \succeq 0, \; Z + \Delta Z \succeq 0$

(1), (2) linear equations

Solving search equations

1. $\Delta x_1 F_1 + \cdots + \Delta x_p F_p - \Delta S = 0$
2. **Tr**$F_i \Delta Z = 0, \; i = 1, \ldots, m$
3. $(S + \Delta S)(Z + \Delta Z)$ is made “smaller”
4. $S + \Delta S \succeq 0, \; Z + \Delta Z \succeq 0$

(1), (2) linear equations

(3) accomplished via Newton step, another linear equation

Solving search equations

1. $\Delta x_1 F_1 + \cdots + \Delta x_p F_p - \Delta S = 0$
2. **Tr**$F_i \Delta Z = 0,\ i = 1, \ldots, m$
3. $(S + \Delta S)(Z + \Delta Z)$ is made "smaller"
4. $S + \Delta S \succeq 0,\ Z + \Delta Z \succeq 0$

(1), (2) linear equations

(3) accomplished via Newton step, another linear equation

Solution strategy:

- First, eliminate ΔS from the linear equations
- Next eliminate ΔZ
- Solve a dense linear system in variable Δx
- Reconstruct ΔZ and ΔS
- $S + \Delta S \succeq 0,\ Z + \Delta Z \succeq 0$ ensured using line search

Outline

- A brief introduction to Semidefinite Programming (SDP)
- Focus: LMIs from the Kalman-Yakubovich-Popov Lemma
- **Fast algorithms for SDPs from KYP Lemma**

General-purpose implementation for KYP SDPs

$$\begin{array}{ll} \text{minimize} & c^T x + \mathbf{Tr}(CP) \\ \text{subject to} & \begin{bmatrix} A^T P + PA & PB \\ B^T P & 0 \end{bmatrix} + \sum_{i=1}^p x_i M_i \succeq N \end{array}$$

- $A \in \mathbf{R}^{n \times n}$, $B \in \mathbf{R}^{n \times 1}$
- (A, B) controllable
- $p + n(n+1)/2$ variables

Primal and dual KYP SDPs

Primal SDP

$$\begin{array}{ll} \text{minimize} & c^T x + \mathbf{Tr}(CP) \\ \text{subject to} & \left[\begin{array}{cc} A^T P + PA & PB \\ B^T P & 0 \end{array}\right] + \sum_{i=1}^{p} x_i M_i \succeq N \end{array}$$

Dual SDP

$$\begin{array}{ll} \text{maximize} & -\mathbf{Tr}(NZ) \\ \text{subject to} & AZ_{11} + Z_{11}A^T + \tilde{z}B^T + B\tilde{z}^T = C \\ & \mathbf{Tr} M_i Z = c_i \\ & Z = \left[\begin{array}{cc} Z_{11} & \tilde{z} \\ \tilde{z}^T & 2z_{n+1} \end{array}\right] \succeq 0 \end{array}$$

Primal and dual KYP SDPs

Primal SDP

$$\begin{array}{ll} \text{minimize} & c^T x + \mathbf{Tr}(CP) \\ \text{subject to} & \begin{bmatrix} A^T P + PA & PB \\ B^T P & 0 \end{bmatrix} + \sum_{i=1}^p x_i M_i \succeq N \end{array}$$

Dual SDP

$$\begin{array}{ll} \text{maximize} & -\mathbf{Tr}(NZ) \\ \text{subject to} & AZ_{11} + Z_{11}A^T + \tilde{z}B^T + B\tilde{z}^T = C \\ & \mathbf{Tr} M_i Z = c_i \\ & Z = \begin{bmatrix} Z_{11} & \tilde{z} \\ \tilde{z}^T & 2z_{n+1} \end{bmatrix} \succeq 0 \end{array}$$

(For future reference $z = [\tilde{z}^T, z_{n+1}]^T$)

Search equations for KYP SDPs

$$
\begin{aligned}
W\Delta Z W + \left[\begin{array}{cc} A^T\Delta P + \Delta P A & \Delta P B \\ B^T \Delta P & 0 \end{array}\right] + \sum_{i=1}^{p} \Delta x_i M_i &= D \\
A\Delta Z_{11} + \Delta Z_{11} A^T + \Delta \tilde{z} B^T + B \Delta \tilde{z}^T &= 0 \\
\mathbf{Tr}\, M_i \Delta Z &= 0
\end{aligned}
$$

$W \succ 0$; values of W, D change at each iteration

Search equations for KYP SDPs

$$
\begin{aligned}
W\Delta Z W + \left[\begin{array}{cc} A^T\Delta P + \Delta P A & \Delta P B \\ B^T \Delta P & 0 \end{array}\right] + \sum_{i=1}^{p} \Delta x_i M_i &= D \\
A\Delta Z_{11} + \Delta Z_{11} A^T + \Delta \tilde{z} B^T + B \Delta \tilde{z}^T &= 0 \\
\mathbf{Tr}\, M_i \Delta Z &= 0
\end{aligned}
$$

$W \succ 0$; values of W, D change at each iteration

For convenience:

$$
\mathcal{K}(P) = \left[\begin{array}{cc} A^T P + PA & PB \\ B^T P & 0 \end{array}\right], \qquad \mathcal{M}(x) = \textstyle\sum_{i=1}^{p} x_i M_i
$$

Search equations for KYP SDPs

$$\begin{aligned} W\Delta Z W + \left[\begin{array}{cc} A^T\Delta P + \Delta P A & \Delta P B \\ B^T \Delta P & 0 \end{array}\right] + \sum_{i=1}^{p} \Delta x_i M_i &= D \\ A\Delta Z_{11} + \Delta Z_{11} A^T + \Delta\tilde{z} B^T + B\Delta\tilde{z}^T &= 0 \\ \mathbf{Tr} M_i \Delta Z &= 0 \end{aligned}$$

$W \succ 0$; values of W, D change at each iteration

For convenience:

$$\mathcal{K}(P) = \left[\begin{array}{cc} A^T P + PA & PB \\ B^T P & 0 \end{array}\right], \qquad \mathcal{M}(x) = \sum_{i=1}^{p} x_i M_i$$

Then,

$$\mathcal{K}^{\mathrm{adj}}(\Delta Z) = A\Delta Z_{11} + \Delta Z_{11} A^T + \Delta\tilde{z} B^T + B\Delta\tilde{z}^T, \quad \mathcal{M}^{\mathrm{adj}}(\Delta Z) = \{\mathbf{Tr} M_i \Delta Z\}$$

Standard method of solving the search equations

$$
\begin{aligned}
W\Delta Z W + \mathcal{K}(\Delta P) + \mathcal{M}(\Delta x) &= D \\
\mathcal{K}^{\mathrm{adj}}(\Delta Z) &= 0 \\
\mathcal{M}^{\mathrm{adj}}(\Delta Z) &= 0
\end{aligned}
$$

Standard method of solving the search equations

$$\begin{aligned} W\Delta Z W + \mathcal{K}(\Delta P) + \mathcal{M}(\Delta x) &= D \\ \mathcal{K}^{\mathrm{adj}}(\Delta Z) &= 0 \\ \mathcal{M}^{\mathrm{adj}}(\Delta Z) &= 0 \end{aligned}$$

General-purpose solvers eliminate ΔZ from first equation:

$$\begin{aligned} \mathcal{K}^{\mathrm{adj}}(W^{-1}(\mathcal{K}(\Delta P) + \mathcal{M}(\Delta x))W^{-1}) &= \mathcal{K}^{\mathrm{adj}}(W^{-1}DW^{-1}) \\ \mathcal{M}^{\mathrm{adj}}(W^{-1}(\mathcal{K}(\Delta P) + \mathcal{M}(\Delta x))W^{-1}) &= \mathcal{M}^{\mathrm{adj}}(W^{-1}DW^{-1}) \end{aligned}$$

A dense set of linear equations in ΔP, Δx

Cost: At least $O(n^6)$

Alternative method: Step 1

$$\begin{aligned} W\Delta Z W + \mathcal{K}(\Delta P) + \mathcal{M}(\Delta x) &= D \\ \mathcal{K}^{\mathrm{adj}}(\Delta Z) &= 0 \\ \mathcal{M}^{\mathrm{adj}}(\Delta Z) &= 0 \end{aligned}$$

Alternative method: Step 1

$$\begin{aligned} W\Delta Z W + \mathcal{K}(\Delta P) + \mathcal{M}(\Delta x) &= D \\ A\Delta Z_{11} + \Delta Z_{11}A^T + \Delta\tilde{z}B^T + B\Delta\tilde{z}^T &= 0 \\ \mathcal{M}^{\mathrm{adj}}(\Delta Z) &= 0 \end{aligned}$$

Use second equation to express ΔZ_{11} in terms of $\Delta\tilde{z}$:

$$\Delta Z_{11} = \sum_{i=1}^n \Delta z_i X_i, \qquad \text{where } AX_i + X_iA^T + Be_i^T + e_iB^T = 0$$

Thus $\quad \Delta Z = \mathcal{B}(\Delta z) = \begin{bmatrix} \sum_{i=1}^n \Delta z_i X_i & \Delta\tilde{z} \\ \Delta\tilde{z}^T & 2\Delta z_{n+1} \end{bmatrix}$

Alternative method: Step 1

$$\begin{aligned} W\Delta Z W + \mathcal{K}(\Delta P) + \mathcal{M}(\Delta x) &= D \\ A\Delta Z_{11} + \Delta Z_{11}A^T + \Delta\tilde{z}B^T + B\Delta\tilde{z}^T &= 0 \\ \mathcal{M}^{\mathrm{adj}}(\Delta Z) &= 0 \end{aligned}$$

Use second equation to express ΔZ_{11} in terms of $\Delta\tilde{z}$:

$$\Delta Z_{11} = \sum_{i=1}^n \Delta z_i X_i, \quad \text{where } AX_i + X_iA^T + Be_i^T + e_iB^T = 0$$

Thus $\Delta Z = \mathcal{B}(\Delta z) = \begin{bmatrix} \sum_{i=1}^n \Delta z_i X_i & \Delta\tilde{z} \\ \Delta\tilde{z}^T & 2\Delta z_{n+1} \end{bmatrix}$

Substituting in first and third equations gives

$$\begin{aligned} W\mathcal{B}(\Delta z)W + \mathcal{K}(\Delta P) + \mathcal{M}(\Delta x) &= D \\ \mathcal{M}^{\mathrm{adj}}(\mathcal{B}(\Delta z)) &= 0 \end{aligned}$$

Alternative method: Step 2

$$\begin{aligned} W\mathcal{B}(\Delta z)W + \mathcal{K}(\Delta P) + \mathcal{M}(\Delta x) &= D \\ \mathcal{M}^{\mathrm{adj}}(\mathcal{B}(\Delta z)) &= 0 \end{aligned}$$

Note that $G = \mathcal{K}(\Delta P)$ for some $\Delta P \iff \mathcal{B}^{\mathrm{adj}}(G) = 0$

Alternative method: Step 2

$$\begin{aligned} W\mathcal{B}(\Delta z)W + \mathcal{K}(\Delta P) + \mathcal{M}(\Delta x) &= D \\ \mathcal{M}^{\mathrm{adj}}(\mathcal{B}(\Delta z)) &= 0 \end{aligned}$$

Note that $G = \mathcal{K}(\Delta P)$ for some $\Delta P \iff \mathcal{B}^{\mathrm{adj}}(G) = 0$

Use to eliminate ΔP:

$$\begin{aligned} \mathcal{B}^{\mathrm{adj}}(W\mathcal{B}(\Delta z)W) + \mathcal{B}^{\mathrm{adj}}(\mathcal{M}(\Delta x)) &= \mathcal{B}^{\mathrm{adj}}(D) \\ \mathcal{M}^{\mathrm{adj}}(\mathcal{B}(\Delta z)) &= 0 \end{aligned}$$

$n + p + 1$ linear equations in $n + p + 1$ variables Δz, Δx

Alternative method: Summary

Reduced search equations of the form

$$\begin{bmatrix} P_{11} & P_{12} \\ P_{12}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta x \end{bmatrix} = \begin{bmatrix} q_1 \\ 0 \end{bmatrix}$$

- Cost of solving is $O(n^3)$ operations (if we assume $p = O(n)$)
- From Δz, Δx, can find ΔZ, ΔP in $O(n^3)$ operations
- Need to precompute X_is ($O(n^4)$)
- P_{12} is independent of current iterates and can be pre-computed, in $O(n^4)$
- Constructing P_{11} requires constructing terms such as **Tr**$(X_i W_{11} X_j W_{11})$ and $W_{11} X_i W_{12}$ (also $O(n^4)$)
- **Overall cost dominated by** $O(n^4)$

Numerical example

$n = p$	KYP IPM prep. time	KYP IPM time/iter.	SeDuMi (primal) time/iter.
25	0.1	0.07	0.1
50	1.2	0.3	7.4
100	21.7	3.3	324.7
200	438.3	31.6	

- CPU time in seconds on 2.4GHz PIV with 1GB of memory
- KYP-IPM: Matlab implementation of alternative method
- SeDuMi (primal): SeDuMi version 1.05 applied to primal problem
- Prep. time is time to compute matrices X_i
- #iterations in both methods is comparable (7–15)

Further reduction in computation

Use factorization of A to compute terms such as **Tr**$(X_i W_{11} X_j W_{11})$ without computing X_i, i.e., without explicitly solving

$$AX_i + X_i A^T + Be_i^T + e_i B^T = 0, \quad i = 1, \ldots, n$$

- Advantages: no need to store matrices X_i, faster construction of reduced search equations
- Possible factorizations: eigenvalue decomposition, companion form, . . .
- For example, if A has distinct eigenvalues $A = V\,\textbf{diag}(\lambda)V^{-1}$, easy to write down search equations in $O(n^3)$, in terms of V and λ

Existence of distinct stable eigenvalues

- By assumption, (A, B) is controllable; hence can arbitrarily assign eigenvalues of $A + BK$ by choosing K
- Choose $T = \begin{bmatrix} I & 0 \\ K & I \end{bmatrix}$, and replace LMI by equivalent LMI

$$T^T \left(\begin{bmatrix} A^T P + PA & PB \\ B^T P & 0 \end{bmatrix} + \sum_{i=1}^{N} x_i M_i \right) T \succeq T^T N T$$

$$\begin{bmatrix} (A + BK)^T P + P(A + BK) & PB \\ B^T P & 0 \end{bmatrix} + \sum_{i=1}^{N} x_i (T^T M_i T) \succeq T^T N T$$

Conclusion: Can assume without loss of generality that A is stable with distinct eigenvalues

Numerical example

Five randomly generated problems with $p = 50$, $n = 100, \ldots, 500$

	KYP IPM (fast)		KYP IPM		SeDuMi (primal)	
n	prep. time	time/iter	prep. time	time/iter	prep. time	time/iter
100	1.3	1.2	21.7	3.3	–	324.7
200	10.1	8.9	438.3	31.6		
300	32.4	27.3				
400	72.2	62.0				
500	140.4	119.4				

- KYP-IPM (fast) uses eigenvalue decomposition of A to construct reduced search equations
- Preprocessing time and time/iteration grow as $O(n^3)$

Conclusions

SDPs derived from the KYP-lemma

- A useful class of SDPs, widely encountered in systems, control and signal processing
- Difficult to solve using general-purpose software
- Generic solvers take $O(n^6)$ computation

Fast solution using interior-point methods

- Custom implementation based on fast solution of search equations (cost $O(n^4)$ or $O(n^3)$)