# SDP solvers

Available under the Matlab environment

Primal-dual path-following predictor-corrector algorithms:
- SeDuMi (Sturm)
- SDPT3 (Toh, Tütüncü, Todd)
- CSDP (Borchers)
- SDPA (Kojima and colleagues)

parallel version available

Primal-dual potential reduction:
- MAXDET (Wu, Vandenberghe, Boyd)

explicit max det terms in objective function

Dual-scaling path-following algorithms:
- DSDP (Benson, Ye, Zhang)

exploits structure for combinatorics

Barrier method and augmented Lagrangian:
- PENSDP (Kočvara, Stingl)

## Matrices as variables

Generally, in control problems we do not encounter the LMI in canonical or semidefinite form but rather with matrix variables

Lyapunov's inequality

$$A^T P + PA < 0 \quad P = P^T > 0$$

can be written in canonical form

$$F(x) = F_0 + \sum_{i=1}^{m} F_i x_i > 0$$

with the notations

$$F_0 = 0 \quad F_i = -A^T B_i - B_i A$$

where $B_i$, $i = 1, \ldots, n(n+1)/2$ are matrix bases for symmetric matrices of size $n$

Most software packages for solving LMIs however work with canonical or semidefinite forms, so that a (sometimes time-consuming) pre-processing step is required

# LMI solvers

Available under the Matlab environment

Projective method: project iterate on ellipsoid within PSD cone = least squares problem

• LMI Control Toolbox (Gahinet, Nemirovski) exploits structure with rank-one linear algebra warm-start + generalized eigenvalues originally developed for INRIA's Scilab

LMI interface to SDP solvers

• LMITOOL (Nikoukah, Delebecque, El Ghaoui) for both Scilab and Matlab
• SeDuMi Interface (Peaucelle)
• YALMIP (Löfberg)

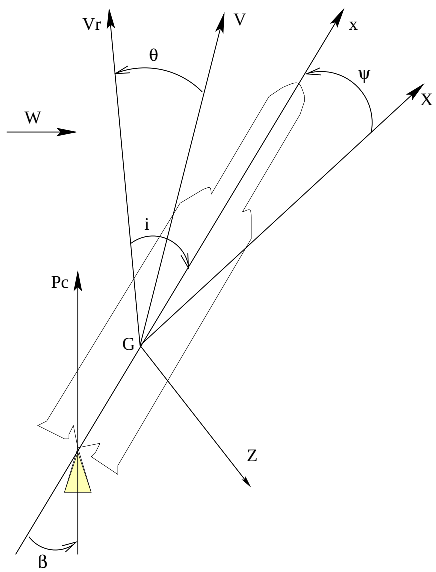
See Helmberg's page on SDP

`www-user.tu-chemnitz.de/~helmberg/semidef.html`

and Mittelmann's page on optimization software with benchmarks

`plato.la.asu.edu/guide.html`

# Numerical example




Control of an aerospace launcher

Linearized model of a rigid launcher

$$\ddot{\psi}(t) = A_6\left(\psi(t) + \frac{\dot{z}(t) - W(t)}{V}\right) + K_1\beta(t)$$

$$\ddot{z}(t) = a_1\psi(t) + a_2\left(\dot{z}(t) - W(t)\right) + a_3\beta(t)$$

$$i(t) = \psi(t) + \frac{\dot{z}(t) - W(t)}{V}$$

Uncertainty: aerodynamic and thruster efficiency

$$\underline{A}_6 \le A_6 \le \overline{A}_6 \quad \underline{K}_1 \le K_1 \le \overline{K}_1$$

# Numerical example (2)

State-space model:

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 \\ A_6 & 0 & \dfrac{A_6}{V} \\ a_1 & 0 & a_2 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ -\dfrac{A_6}{V} \\ -a_2 \end{bmatrix} W + \begin{bmatrix} 0 \\ K_1 \\ a_3 \end{bmatrix} u(t)$$

$$z(t) = i(t) = \begin{bmatrix} 1 & 0 & \dfrac{1}{V} \end{bmatrix} x(t) - \frac{1}{V} W$$

Robust state-feedback synthesis: $u_k = K x_k$

$$\begin{bmatrix} x_{k+1} \\ z_k \end{bmatrix} = M \begin{bmatrix} x_k \\ w_k \\ u_k \end{bmatrix} = \begin{bmatrix} A & B_1 & B \\ C_1 & D_1 & D_{1u} \end{bmatrix} \begin{bmatrix} x_k \\ w_k \\ u_k \end{bmatrix}$$

$$M \in \text{co} \left\{ M^{[1]}, \cdots, M^{[N]} \right\}$$

Impulse-to-peak norm minimization:

$$\begin{array}{ll} \min\limits_{\mathbf{K} \in \mathcal{K}} & \boldsymbol{\gamma_{i2p}} \\ \text{under} & ||\mathbf{\Sigma} \star \mathbf{K}||_{i2p}^2 \leq \boldsymbol{\gamma_{i2p}} \end{array}$$

Nota:
$||\mathbf{\Sigma} \star \mathbf{K}||_{i2p} = ||z||_\infty$ when $w$ is an impulse

# Numerical example (3)

Convex relaxation via LMIs:

$$\gamma_G^* = \min_{G,\ X^{[i]},\ \gamma_G} \quad \gamma$$

$$\begin{bmatrix} -P^{[i]} & A^{[i]}G + B_1^{[i]}S \\ \star & X^{[i]} - G - G' \end{bmatrix} \prec \mathbf{0}$$

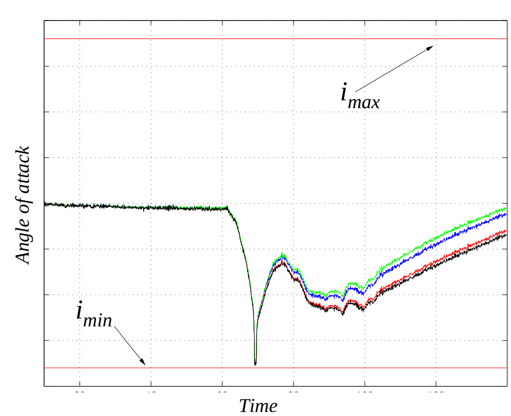
$$\begin{bmatrix} -X^{[i]} & B_1^{[i]} \\ \star & -\mathbf{1} \end{bmatrix} \prec \mathbf{0}$$

$$\begin{bmatrix} -\gamma\mathbf{1} & C_1^{[i]}X^{[i]} + D_{1u}^{[i]}S \\ \star & X^{[i]} - G - G' \end{bmatrix} \prec \mathbf{0}$$

$$\begin{bmatrix} -\gamma\mathbf{1} & D_1^{[i]} \\ \star & -\mathbf{1} \end{bmatrix} \prec \mathbf{0}$$

Stabilizing state-feedback:

$$K_G = SG^{-1} \quad ||z||_\infty < \sqrt{\gamma_G^*}$$



# Numerical example (4)

```
>> yalmip('clear');
>> for i=1:N
     Xv{i}=sdpvar(n,n,'symmetric','real');
end
>> Gv=sdpvar(n,n,'full','real');
>> Sv=sdpvar(m,n,'full','real');
>> gv=sdpvar(1,1,'full','real');
>> L=set;
>> for i=1:N
     L=L+set([-Xv{i} sys.A{i}*Gv+sys.B{i}*Sv;...
     Gv'*sys.A{i}'+Sv'*sys.B{i}' Xv{i}-Gv-Gv']<0,'Lyapunov');

     L=L+set(sys.B1{i}*sys.B1{i}'-Xv{i}<0,'B');

     L=L+set([-gv*eye(r) sys.C1{i}*Gv+sys.D1u{i}*Sv;...
     Gv'*sys.C1{i}'+Sv'*sys.D1u{i}'  Xv{i}-Gv-Gv']<0,'C');

     L=L+set(sys.D1{i}*sys.D1{i}'-gv*eye(r)<0,'D');end
>> sol=solvesdp(L,[],gv,ops);
>> for i=1:N
      X{i}=double(Xv{i});
end
>> G=double(Gv); S=double(Sv);
```