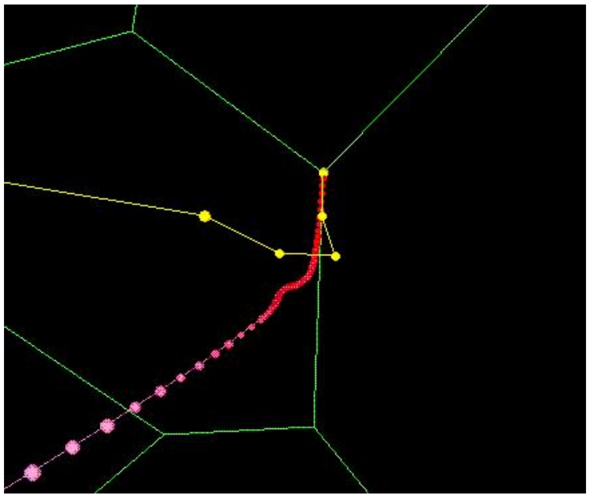
COURSE ON LMI OPTIMIZATION
WITH APPLICATIONS IN CONTROL
PART I.5

SOLVING LMIs

Denis Arzelier
www.laas.fr/∼arzelier
arzelier@laas.fr



January 2005

History

Convex programming

- Logarithmic barrier function [Frisch 1955)]
- Method of centers ([Huard 1967]

Interior-point (IP) methods for LP

- Ellipsoid algorithm [Khachiyan 1979]

polynomial bound on worst-case iteration count
- IP methods for LP [Karmarkar 1984]

improved complexity bound and efficiency - About 50% of commercial LP solvers

IP methods for SDP

- Self-concordant barrier functions [Nesterov, Nemirovski 1988], [Alizadeh 1991]
- IP methods for general convex programs (SDP and LMI)

Academic and commercial solvers (MATLAB)

Interior point methods (1)

For the optimization problem

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & f_0(x) \\ \text{s.t.} & f_i(x) \geq 0 \; i = 1, \cdots, m \end{array}$$

where the $f_i(x)$ are twice continuously differentiable convex functions

Sequential minimization techniques: Reduction of the initial problem into a sequence of unconstraint optimization problems
[Fiacco - Mc Cormick 68]

$$\boxed{\min_{x \in \mathbb{R}^n} f_0(x) + \mu \phi(x)}$$

where $\mu > 0$ is a parameter sequentially decreased to 0 and the term $\phi(x)$ is a barrier function
Barrier functions go to infinity as the boundary of the feasible set is approached

Interior point methods (2)
Descent methods

To solve an unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x)$$

we produce a minimizing sequence

$$x_{k+1} = x_k + t_k \Delta x_k$$

where $\Delta x_k \in \mathbb{R}^n$ is the step or search direction and $t^{(k)} \geq 0$ is the step size or step length

A descent method consists in finding a sequence $\{x_k\}$ such that

$$f(x^\star) \leq \cdots f(x_{k+1}) < f(x_k)$$

where $x^\star$ is the optimum

General descent method

```
0. $k = 0$; given starting point $x_k$
1. determine descent direction $\Delta x_k$
2. line search: choose step size $t_k > 0$
3. update: $k = k + 1$; $x_k = x_{k-1} + t_{k-1} \Delta x_{k-1}$
4. go to step 1 until a stopping criterion
   is satisfied
```

Interior point methods (3)
Newton's method

A particular choice of search direction is the Newton step

$$\boxed{\Delta x = -\nabla^2 f(x)^{-1} \nabla f(x)}$$

where
- $\nabla f(x)$ is the gradient
- $\nabla^2 f(x)$ is the Hessian

This step $y = \Delta x$ minimizes the second-order Taylor approximation

$$\widehat{f}(x+y) = f(x) + \nabla f(x)' y + y' \nabla^2 f(x) y / 2$$

and it is the steepest descent direction for the quadratic norm defined by the Hessian

Quadratic convergence near the optimum

Interior point methods (4)
Conic optimization

For the conic optimization problem

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & f_0(x) \\ \text{s.t.} & f_i(x) \preceq_{\mathcal{K}} 0 \; i = 1, \cdots, m \end{array}$$

suitable barrier functions are called self-concordant

Smooth convex 3-differentiable functions f with second derivative Lipschitz continuous w.r. to the local metric induced by the Hessian

$$\boxed{|f'''(x)| \leq 2 f''(x)^{3/2}}$$

- goes to infinity as the boundary of the cone is approached
- can be efficiently minimized by Newton's method
- Each convex cone $\mathcal{K}$ possesses a self-concordant barrier
- Such barriers are only computable for some special cones

Barrier function for LP (1)

For LP and positive orthant $\mathbb{R}^n_+$, the logarithmic barrier function

$$\phi(y) = -\sum_{i=1}^{n} \log(y_i) = \log \prod_{i=1}^{n} y_i^{-1}$$

is convex in the interior $y \succ 0$ of the feasible set and is instrumental to design IP algorithms

$$\begin{array}{ll} \max\limits_{\mu \in \mathbb{R}^p} & b'y \\ \text{s.t.} & c_i - a_i y \succeq \mathbf{0},\ \ i = 1, \cdots, m,\ \ (y \in \mathcal{P}) \end{array}$$

$$\phi(y) = -\log \prod_{i=1}^{m} (c_i - a_i y) = -\sum_{i=1}^{m} \log(c_i - a_i y)$$

The optimum

$$y_c = \arg\left[\min_y \phi(y)\right]$$

is called the analytic center of the polytope

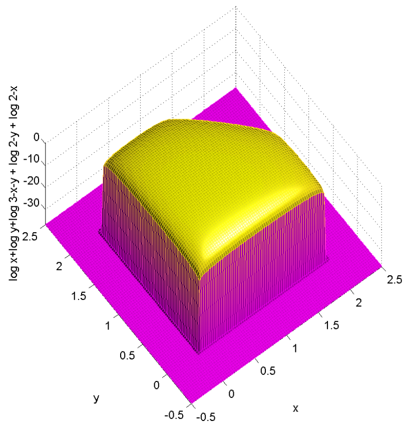
Barrier function for LP (2)
Example

$$J_1^* = \max_{x,y} \quad 2x + y$$

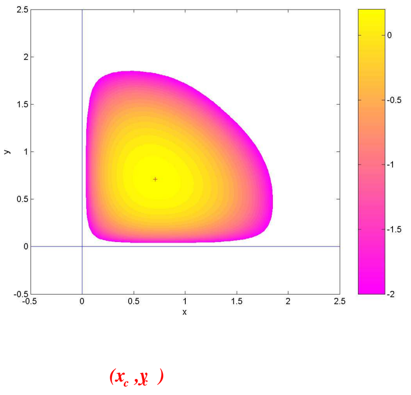
$$\text{s.t.} \quad x \geq 0 \quad y \geq 0 \quad x \leq 2$$

$$y \leq 2 \quad x + y \leq 3$$

$$\phi(x,y) = -\log(xy) - \log(2-x) - \log(2-y) - \log(3-x-y)$$



$$(x_c, y_c) = \left(\frac{6-\sqrt{6}}{5}, \frac{6-\sqrt{6}}{5}\right)$$



Barrier function for an LMI (1)

Given an LMI constraint $F(x) \succeq 0$

Self-concordant barriers are smooth convex 3-differentiable functions $\phi : \mathbb{S}^n_+ \to \mathbb{R}$ s.t. for $\overline{\phi}(\alpha) = \phi(X + \alpha H)$ for $X \succ 0$ and $H \in \mathbb{S}^n$

$$|\overline{\phi}'''(0)| \leq 2\overline{\phi}''(0)^{3/2}$$

Logarithmic barrier function

$$\phi(x) = -\log\det F(x) = \log\det F(x)^{-1}$$

This function is analytic, convex and self-concordant on $\{x : F(x) \succ 0\}$

The optimum

$$x_c = \arg\left[\min_x \phi(x)\right]$$

is called the analytic center of the LMI

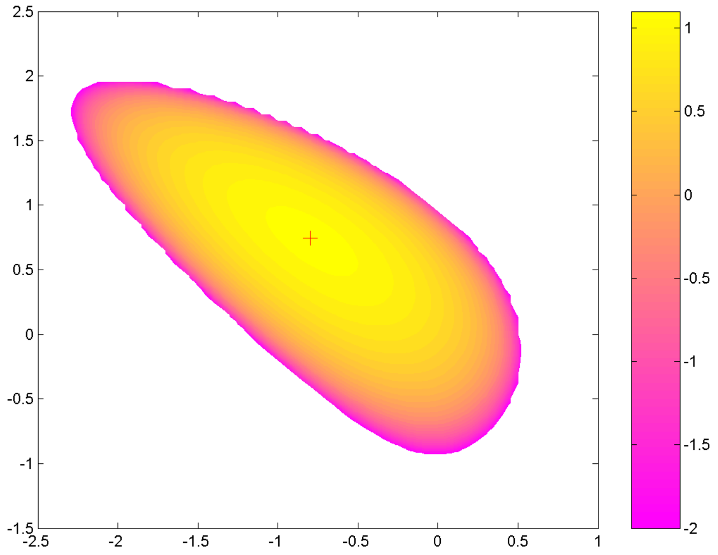
Barrier function for an LMI (2)
Example (1)

$$F(x_1,x_2) = \begin{bmatrix} 1-x_1 & x_1+x_2 & x_1 \\ x_1+x_2 & 2-x_2 & 0 \\ x_1 & 0 & 1+x_2 \end{bmatrix} \succeq 0$$

Computation of analytic center:

$$\nabla_{x_1} \log\det F(x) = 2 + 3x_2 + 6x_1 + x_2^2 = 0$$

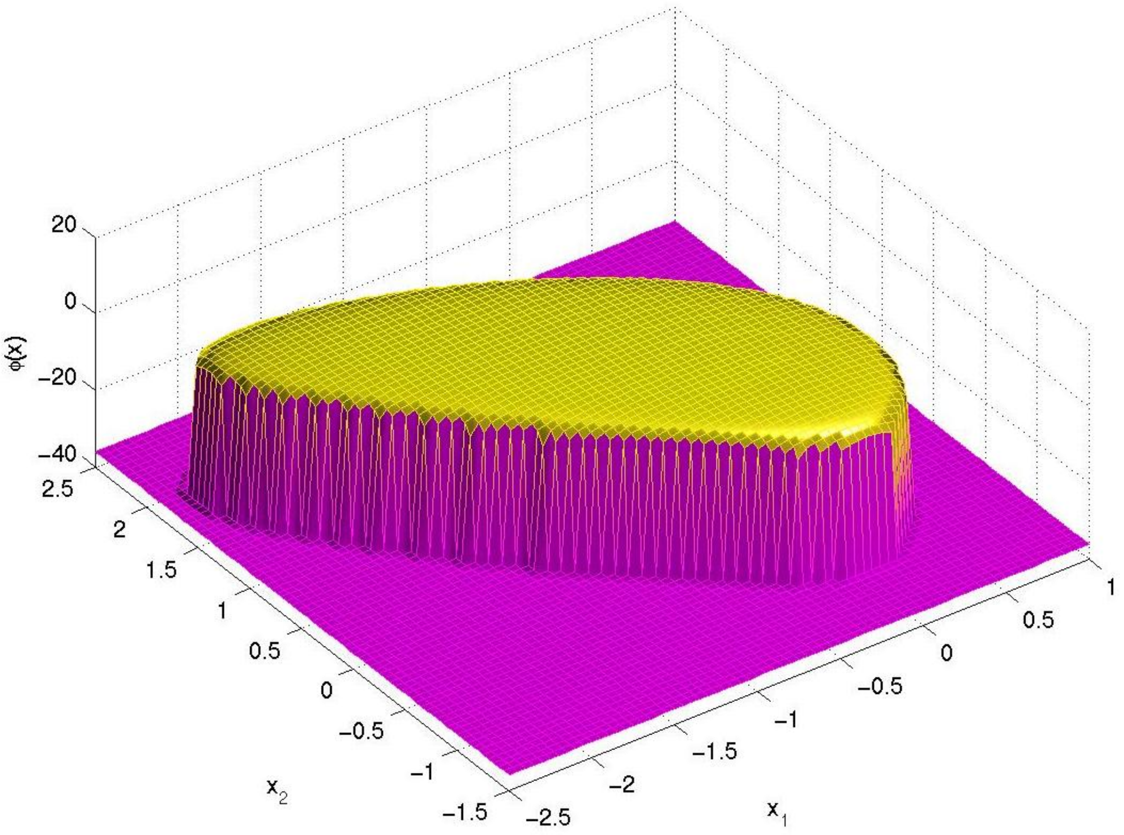
$$\nabla_{x_2} \log\det F(x) = 1 - 3x_1 - 4x_2 - 3x_2^2 - 2x_1x_2 = 0$$



$$x_{1c} = -0.7989 \qquad x_{2c} = 0.7458$$

Barrier function for an LMI (3)
Example (2)

The barrier function $\phi(x)$ is flat in the interior of the feasible set and sharply increases toward the boundary



IP methods for SDP (1)

Primal / dual SDP

$$\begin{array}{ll} \min_{Z} & -\text{trace}(F_0 Z) \\ \text{s.t.} & -\text{trace}(F_i Z) = c_i \\ & Z \succeq 0 \end{array}$$

$$\begin{array}{ll} \min_{x,\,Y} & c'x \\ \text{s.t.} & Y + F_0 + \sum_{i=1}^{m} x_i F_i = \mathbf{0} \\ & Y \succeq 0 \end{array}$$

Remember KKT optimality conditions

$$\begin{array}{rcll} F_0 + \sum_{i=1}^{m} x_i F_i + Y & = & \mathbf{0} & Y \succeq \mathbf{0} \\ \forall\ i \ \text{trace}\, F_i Z + c_i & = & \mathbf{0} & Z \succeq \mathbf{0} \\ Z^\star F(x^\star) = Z^\star Y^\star & = & \mathbf{0} & \end{array}$$

IP methods for SDP (2)
The central path

Perturbed KKT optimality conditions = Centrality conditions

$$\begin{array}{rcll} F_0 + \sum_{i=1}^{m} x_i F_i + Y & = & \mathbf{0} & Y \succeq \mathbf{0} \\ \forall\ i \ \text{trace}\, F_i Z + c_i & = & \mathbf{0} & Z \succeq \mathbf{0} \\ ZY & = & \mu \mathbf{1} & \end{array}$$

where $\mu > 0$ is the centering parameter or barrier parameter

For any $\mu > 0$, centrality conditions have a unique solution $Z(\mu), x(\mu), Y(\mu)$ which can be seen as the parametric representation of an analytic curve: The central path

The central path exists if the primal and dual are strictly feasible and converges to the analytic center when $\mu \to 0$

IP methods for SDP (3)
Primal methods

$$\begin{array}{ll} \min\limits_{Z} & -\text{trace}(F_0 Z) - \mu \log\det Z \\ \text{s.t.} & \text{trace}(F_i Z) = -c_i \end{array}$$

where parameter μ is sequentially decreased to zero

Follow the primal central path approximately: Primal path-following methods

The function $f_p^{\mu}(Z)$

$$f_p^{\mu}(Z) = -\frac{1}{\mu}\text{trace}(F_0 Z) - \log\det\, Z$$

is the primal barrier function and the primal central path corresponds to the minimizers $Z(\mu)$ of $f_p^{\mu}(Z)$

- The projected Newton direction ΔZ
- Updating of the centering parameter μ

IP methods for SDP (4)
Dual methods (1)

$$\begin{aligned} \min_{x,Y} \quad & c'x - \mu \log\det Y \\ \text{s.t.} \quad & Y + F_0 + \sum_{i=1}^{m} x_i F_i = \mathbf{0} \end{aligned}$$

where parameter μ is sequentially decreased to zero

The function $f_d^{\mu}(x, Y)$

$$f_d^{\mu}(x, Y) = \frac{1}{\mu} c'x - \log\det Y$$

is the dual barrier function and the dual central path corresponds to the minimizers $(x(\mu), Y(\mu))$ of $f_d^{\mu}(x, Y)$

$Y_k \succeq 0$ ensured via Newton process:

- Large decreases of μ require damped Newton steps
- Small updates allow full (deep) Newton steps

Dual methods (2)
Newton step for LMI

The centering problem is

$$\min \phi(x) = \frac{1}{\mu} c' x - \log\det(-F(x))$$

and at each iteration Newton step Δx satisfies the linear system of equations (LSE)

$$\boxed{H \Delta x = -g}$$

where gradient g and Hessian H are given by

$$\begin{aligned} H_{ij} &= \text{trace}\, F(x)^{-1} F_i F(x)^{-1} F_j \\ g_i &= c_i / \mu - \text{trace}\, F(x)^{-1} F_i \end{aligned}$$

LSE typically solved via Cholesky factorization or QR decomposition (near the optimum)

Nota: Expressions for derivatives of $\phi(x) = -\log\det F(x)$
Gradient:

$$\begin{aligned} (\nabla \phi(x))_i &= -\text{trace}\, F(x)^{-1} F_i \\ &= -\text{trace}\, F(x)^{-1/2} F_i F(x)^{-1/2} \end{aligned}$$

Hessian:

$$\begin{aligned} (\nabla^2 \phi(x))_{ij} &= \text{trace}\, F(x)^{-1} F_i F(x)^{-1} F_j \\ &= \mu \text{trace} \left(F(x)^{-1/2} F_i F(x)^{-1/2}\right) \left(F(x)^{-1/2} F_j F(x)^{-1/2} \right. \end{aligned}$$

Complexity of dual methods

For the n-by-n LMI $F(x) \preceq 0$ with m variables
the flops count of IP methods for SDP is as follows:

For each iteration:
(a) $\mathcal{O}(n^2m)$ to form $F(x)$
(b) $\mathcal{O}(n^3m)$ to form $F(x)^{-1}F_iF(x)^{-1}F_j$
(c) $\mathcal{O}(n^2m^2)$ to form $F(x)^{-1}F_i$
(d) $\mathcal{O}(m^3)$ to solve Newton LSE with Cholesky

Dominating terms are (b) and (c) so the complexity for solving one Newton step is:

$$\boxed{\mathcal{O}(n^3m + n^2m^2)}$$

..but structure can be exploited in these steps !

Number of iterations with Newton's method:

$$\boxed{\mathcal{O}(\sqrt{n}\log\varepsilon^{-1})}$$

where ε is the desired accuracy

In general, it is assumed that $m = \mathcal{O}(n^2)$ otherwise redundant constraints can be removed, so the global worst-case complexity for a dense SDP is

$$\boxed{\mathcal{O}(n^{6.5}\log\varepsilon^{-1})}$$

Much less in practice !

IP methods for SDP (4)
Primal-dual methods (1)

$$\begin{array}{ll} \min\limits_{x,Y,Z} & \text{trace}\, YZ - \mu \log\det YZ \\ \text{s.t.} & -\text{trace}\, F_i Z = c_i \\ & Y + F_0 + \sum_{i=1}^{m} x_i F_i = \mathbf{0} \end{array}$$

Minimizers $(x(\mu), Y(\mu), Z(\mu))$ satisfy optimality conditions

$$\begin{array}{rcl} \text{trace}\, F_i Z &=& -c_i \\ \sum_{i=1}^{m} x_i F_i + Y &=& -F_0 \\ YZ &=& \mu I \\ Y, Z &\succeq& \mathbf{0} \end{array}$$

The duality gap:

$$-\text{trace}(F_0 Z) - c'x = \text{trace}(YZ) \geq 0$$

is minimized along the central path

IP methods for SDP (5)
Primal-dual methods (2)

For primal-dual IP methods, primal and dual directions ΔZ, Δx and ΔY must satisfy non-linear and over determined system of conditions

$$\begin{aligned}
\text{trace}(F_i \Delta Z) &= 0 \\
\sum_{i=1}^{m} \Delta x_i F_i + \Delta Y &= 0 \\
(Z + \Delta Z)(Y + \Delta Y) &= \mu I \\
Z + \Delta Z &\succeq \mathbf{0} \\
\Delta Z &= \Delta Z' \\
Y + \Delta Y &\succeq \mathbf{0}
\end{aligned}$$

These centrality conditions are solved *approximately* for a given $\mu > 0$, after which μ is reduced and the process is repeated

Key point is in linearizing and symmetrizing the latter equation

IP methods for SDP (6)
Primal-dual methods (3)

The non linear equation in the centrality conditions is replaced by

$$H_P(\Delta Z Y + Z \Delta Y) = \mu \mathbf{1} - H_P(ZY)$$

where H_P is the linear transformation

$$H_P(M) = \frac{1}{2}\left[PMP^{-1} + P^{-1'}M'P'\right]$$

for any matrix M and the scaling matrix P gives the symmetrization strategy.

Following the choice of P, long list of primal-dual search directions, (AHO, HRVW, KSH, M, NT...), the most known of which is Nesterov-Todd's

Algorithms differ in how the symmetrized equations are solved and how μ is updated (long step methods, dynamic updates of for predictor-corrector methods)

Other IP methods for SDP (1)
Affine-scaling methods

Solve the non linear system of optimality conditions via some iterative scheme
A family of directions is formed as solutions of the system

$$\begin{aligned}
\forall\ i = 1, \cdots, m \quad \text{trace}(F_i \Delta Z) &= 0 \\
\sum_{i=1}^{m} \Delta x_i F_i + \Delta Y &= 0 \\
H_P(\Delta Z Y + Z \Delta Y) &= -H_P(ZY) \\
Z + \Delta Z &\succeq \mathbf{0} \\
\Delta Z &= \Delta Z' \\
Y + \Delta Y &\succeq \mathbf{0}
\end{aligned}$$

- Primal affine-scaling algorithms are extension to SDP of Karmarkar's work for LP: No polynomial complexity and search direction may converge to non optimal point
- Primal-dual affine-scaling algorithms: Minimize the duality gap over some prescribed ellipsoid in the primal dual space

Other IP methods for SDP (2)
Primal-dual potential reduction methods

Defining the Tanabe-Todd-Ye potential function

$$\Phi(Z, Y) = (n + \nu\sqrt{n}) \log \text{trace} ZY - \log\det ZY - \log n$$

where $\nu \geq 1$

Polynomial complexity of primal-dual IP algorithms is ensured by the decay of Φ by at least a fixed constant at each iteration

- Compute feasible descent directions $(\Delta Z, \Delta Y)$ for Φ at (Z, Y) strictly feasible

- Plane search: Find

$$(\alpha, \beta) = \arg \min_{\substack{0 \leq \alpha \leq \alpha_{max} \\ 0 \leq \beta \leq \beta_{max}}} \Phi(Z + \alpha\Delta Z, Y + \beta\Delta Y)$$

- First methods to be extended from LP to SDP ([Nesterov-Nemirovsky], [Alizadeh])

IP methods in general

Generally for LP, QP or SDP primal-dual methods outperform primal or dual methods
General characteristics of IP methods:

- Efficiency: About 5 to 50 iterations, almost independent of input data (problem), each iteration is a least-squares problem (well established linear algebra)
- Theory: Worst-case analysis of IP methods yields polynomial computational time
- Structure: Tailored SDP solvers can exploit problem structure

For more information see the Linear, Cone and SDP section at

`www.optimization-online.org`

and the Optimization and Control section at

`fr.arXiv.org/archive/math`

Other algorithms for SDP (1)
Bundle methods

Tackle directly the non differentiable convex optimization problem

$$\min_{x \in \mathbb{R}^m} \quad \lambda_{\max}(F(x)) + c'x$$

Use convex analysis tools (subdifferential, subgradients, cutting planes...)
A subgradient of the function $g(x) = \lambda_{max}(F(x)) + c'x$ at x is a vector $z \in \mathbb{R}^m$ s.t.

$$z \in \partial g(x) = \left\{ z \in \mathbb{R}^m \mid g(x_1) \geq g(x) + z'(x_1 - x) \ \forall \ x_1 \right\}$$

and it is given by

$$z = [\text{trace}(F_i v v')]_{i=1}^m \qquad F(x)v = \lambda v$$

Generalization of descent methods: ϵ-subgradients, spectral bundle methods [Helmberg-Rendl 2000], second order scheme [Oustry 2000]
- Good global convergence properties
- No iterations bound known

Other algorithms for SDP (2)
Penalty/augmented Lagrangian methods (1)

Use similar ideas, but cannot be considered as an interior-point method

When applied to LMI problem

$$\min c'x \text{ s.t. } F(x) = F_0 + \sum_i x_i F_i \preceq 0$$

- penalty method - some eigenvalues of $F(x)$ can be positive
- barrier method - no eigenvalue of $F(x)$ can be positive (use of $\log(-F(x))$)

Augmented Lagrangian

$$\boxed{L(x, Z, p) = c'x + \text{trace}\, Z\Phi(x, p)}$$

with dual variable Z and suitable penalty function, for example

$$\boxed{\Phi(x, p) = p^2(-F(x) + pI)^{-1} - pI}$$

with penalty parameter p

Penalty/augmented Lagrangian methods (2)

General algorithm

1. find x_{k+1} such that $\|\nabla_x L(x, Z_k, p_k)\| \leq \epsilon_k$
2. update dual variables: $Z_{k+1} = f(x_{k+1}, Z_k)$
3. update penalty parameter: $p_{k+1} < p_k$
4. go to step 1 until a stopping criterion is satisfied

Can be considered as a primal-dual method, but dual variables are obtained in closed-form at step 2

Complexity for the n-by-n LMI $F(x) \preceq 0$ with m variables depends mostly on Newton step 1 in $O(n^3m + n^2m^2)$, same as IP methods

Can be improved to $O(m^2K^2)$ where K is the max number of non-zero terms in the F_i

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

$$A'P + PA < 0 \quad P = P' > 0$$

can be written in canonical form

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

with the notations

$$F_0 = 0 \quad F_i = A'B_i + B_i A$$

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

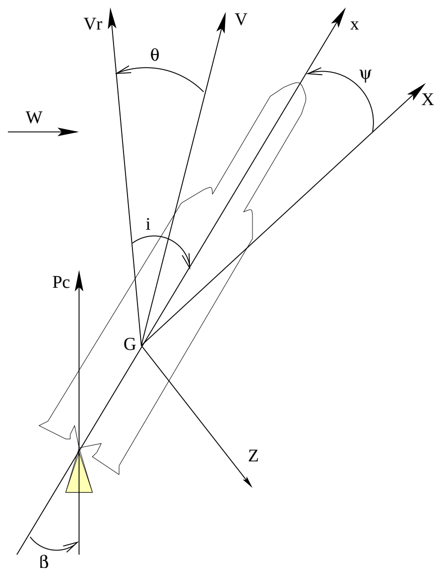
Available under the Matlab environment

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

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

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:

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

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

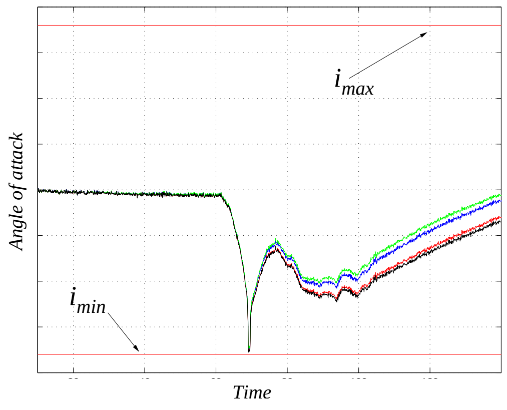
$$\begin{bmatrix} -X^{[i]} & B_1^{[i]} \\ \star & -\mathbf{1} \end{bmatrix} \prec \mathbf{0}$$

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

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

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);
```

LMI relaxation software

GloptiPoly is written as an open-source, general purpose and user-friendly Matlab software

Optionally, problem definition made easier with Matlab Symbolic Math Toolbox, gateway to Maple kernel

Gloptipoly solves small to medium non-convex global optimization problems with multivariate real-valued polynomial objective functions and constraints

Software and documentation available at

`www.laas.fr/∼henrion/software/gloptipoly`

Metholodogy

GloptiPoly builds and solves a hierarchy of successive convex linear matrix inequality (LMI) relaxations of increasing size, whose optima are guaranteed to converge asymptotically to the global optimum

Relaxations are build from LMI formulation of sum-of-squares (SOS) decomposition of multivariate polynomials (see last chapter)

In practice convergence is ensured fast, typically at 2nd or 3rd LMI relaxation

Features

General features of GloptiPoly:

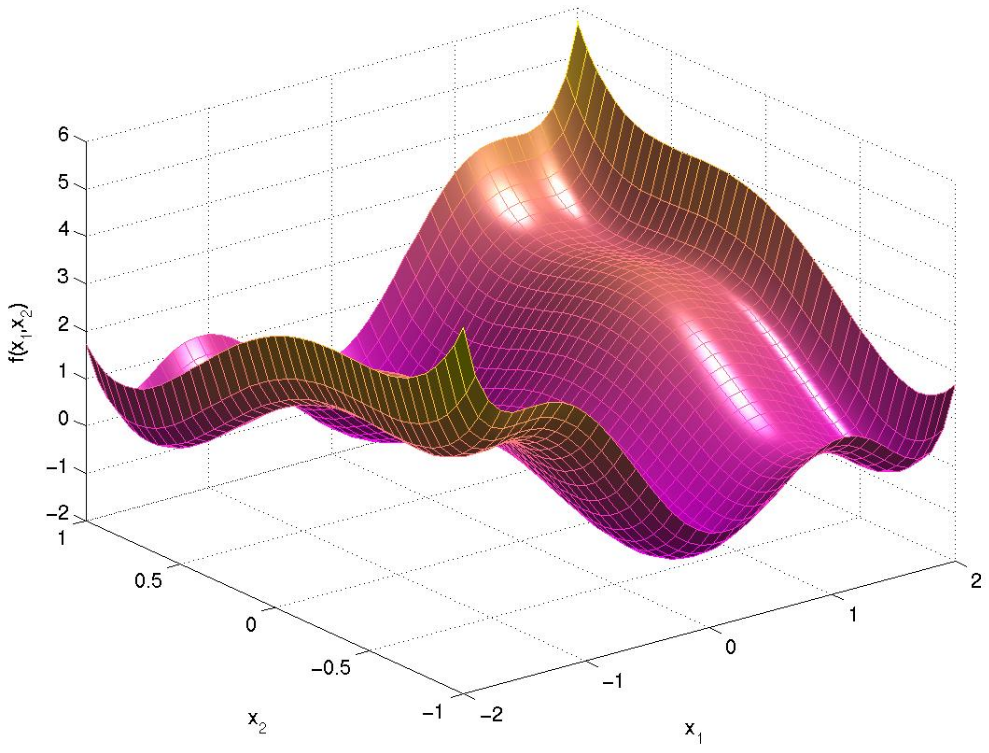
- Certificate of global optimality (rank checks)
- Automatic extraction of globally optimal solutions (multiple eigenvectors)
- 0-1 or ± 1 integer constraints on some of the decision variables (combinatorial optimization problems)
- Generation of input and output data in SeDuMi's format
- Generation of moment matrices associated with LMI relaxations (rank checks)
- User-defined scaling of decision variables (to improve numerical behavior)
- Exploits sparsity of polynomial data

Major update of GloptiPoly (GloptiPoly 3.0) planned (hopefully !) for summer 2005

Benchmark examples
Continuous problems

Mostly from Floudas/Pardalos 1999 handbook

About 80 % of pbs solved with LMI relaxation of small order (typically 2 or 3) in less than 3 seconds on a PC Pentium IV at 1.6 MHz with 512 Mb RAM

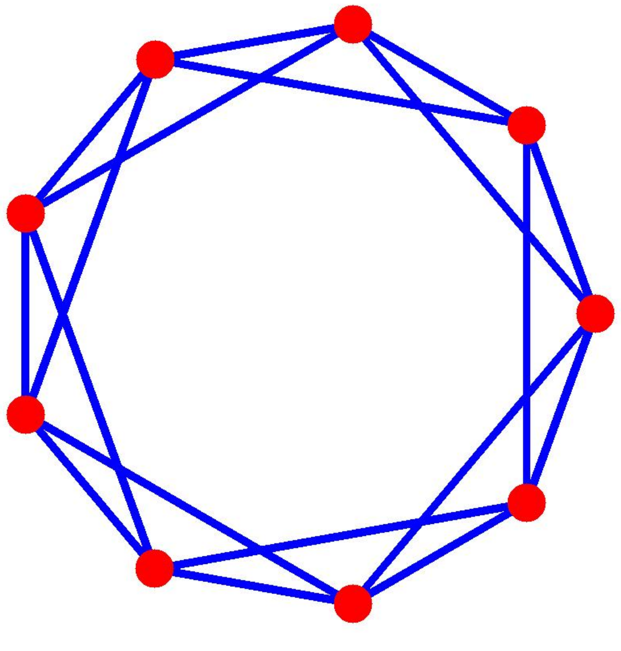


Six-hump camel back function

Benchmark examples
Discrete problems

From Floudas/Pardalos handbook and also Anjos' Ph.D (Univ Waterloo)

By perturbing criterion (destroys symmetry) global convergence ensured on 80 % of pbs in less than 4 seconds



MAXCUT on antiweb AW_9^2 graph

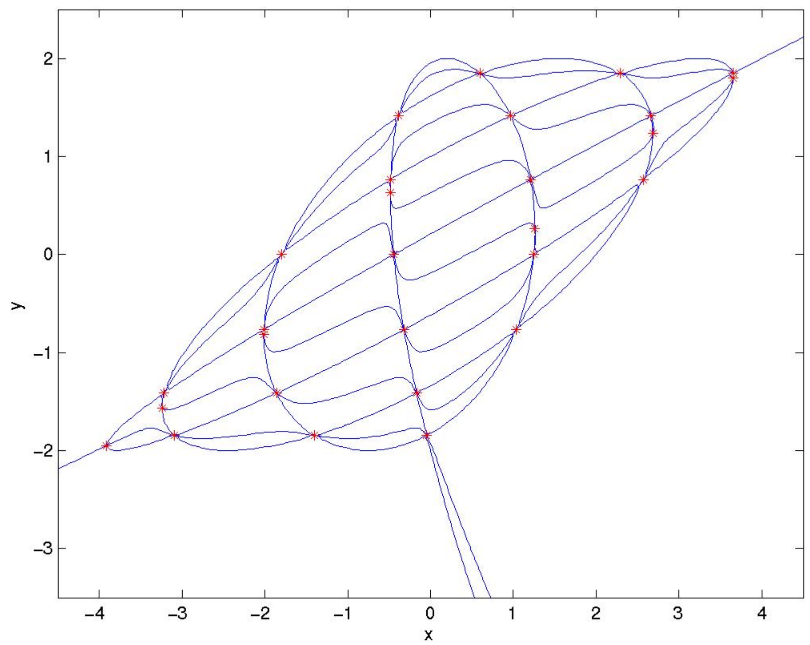
Benchmark examples
Polynomial systems of equations

From Verschelde's and Posso databasis
Real coefficients & coeffs only

Out of 59 systems:

- 61 % solved in t $<$ 10 secs
- 20 % solved in 10 $<$ t $<$ 100 secs
- 10 % solved in t $\geq$ 100 secs
- 9 % out of memory

No criterion optimized
No enumeration of all solutions



Intersections of seventh and eighth degree polynomial curves

GloptiPoly: summary

GloptiPoly is a general-purpose software with a user-friendly interface

Pedagogical flavor, black-box approach,
no expert tuning required to cope with very distinct applied maths and engineering pbs

Not a competitor to highly specialized codes for solving polynomial systems of equations or large combinatorial optimization pbs

Numerical conditioning (Chebyshev basis) deserves further study

See also the SOSTOOLS software

`www.cds.caltech.edu/sostools`