PENNON

A Generalized Augmented Lagrangian Method for Convex NLP and SDP

Michal Kočvara

Institute of Information Theory and Automation

Academy of Sciences of the Czech Republic

and

Czech Technical University

kocvara@utia.cas.cz

http://www.utia.cas.cz/kocvara

PBM Method for convex NLP

Ben-Tal, Zibulevsky, '92, '97

Combination of:
(exterior) **P**enalty meth., (interior) **B**arrier meth., Method of **M**ultipliers

Problem:

$$(CP) \qquad \min_{x\in\mathbb{R}^n} \{f(x) : g_i(x) \leq 0\,, \quad i = 1, \ldots, m\}$$

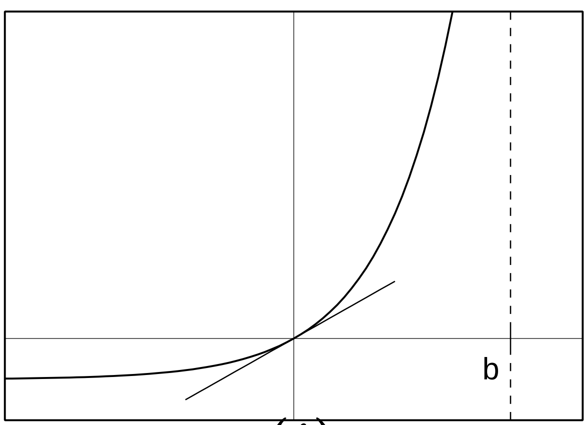
Assume:

1. f, g_i ($i = 1, \ldots, m$) convex
2. X^* nonempty and compact $\hfill (A1)$
3. $\exists \hat{x}$ so that $g_i(\hat{x}) < 0$ for all $i = 1, \ldots, m$ $\hfill (A2)$

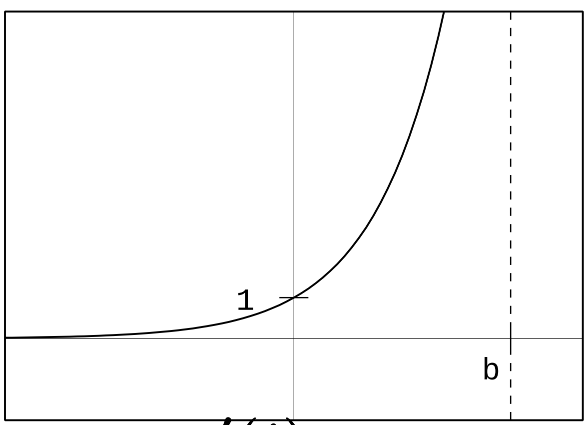
PBM Method for convex NLP

φ possibly smooth, $\mathbf{dom}\varphi$ possibly large

(φ_0)	φ strictly convex, strictly monotone increasing and C^2		
(φ_1)	$\mathbf{dom}\varphi = (-\infty, b)$ with $0 < b \leq \infty$		
(φ_2)	$\varphi(0) = 0\,,$	(φ_4)	$\lim_{t\to b} \varphi'(t) = \infty$
(φ_3)	$\varphi'(0) = 1\,,$	(φ_5)	$\lim_{t\to -\infty} \varphi'(t) = 0$



$\varphi(t)$



$\varphi'(t)$

PBM Method for convex NLP

Examples:

$$\varphi_1^r(t) = \begin{cases} c_1 \frac{1}{2} t^2 + c_2 t + c_3 & t \geq r \\ c_4 \log(t - c_5) + c_6 & t < r\,. \end{cases}$$

PBM Method for convex NLP

Examples:

$$\varphi_1^r(t) \quad = \quad \begin{cases} c_1 \frac{1}{2} t^2 + c_2 t + c_3 & t \geq r \\ c_4 \log(t - c_5) + c_6 & t < r \,. \end{cases}$$

$$\varphi_2^r(t) \quad = \quad \begin{cases} c_1 \frac{1}{2} t^2 + c_2 t + c_3 & t \geq r \,, \\ \frac{c_4}{t - c_5} + c_6 & t < r \,, \quad r \in \langle -1, 1 \rangle \end{cases}$$

Examples:

$$\varphi_1^r(t) = \begin{cases} c_1 \frac{1}{2} t^2 + c_2 t + c_3 & t \geq r \\ c_4 \log(t - c_5) + c_6 & t < r \,. \end{cases}$$

$$\varphi_2^r(t) = \begin{cases} c_1 \frac{1}{2} t^2 + c_2 t + c_3 & t \geq r \,, \\ \frac{c_4}{t - c_5} + c_6 & t < r \,, \quad r \in \langle -1, 1 \rangle \end{cases}$$

Properties:

- C^2, bounded second derivative

  $\Longrightarrow$ improved behaviour of Newton's method

- composition of barrier branch (logarithmic/reciprocal) and penalty branch (quadratic)

PBM algorithm for convex problems

With $p_i > 0$ for $i \in \{1, \ldots, m\}$, we have

$$g_i(x) \le 0 \iff p_i \varphi(g_i(x)/p_i) \le 0, \quad i = 1, \ldots, m$$

The corresponding *augmented Lagrangian*:

$$F(x, u, p) := f(x) + \sum_{i=1}^{m} u_i p_i \varphi(g_i(x)/p_i)$$

PBM algorithm:

$$\begin{aligned}
x^{k+1} &= \arg\min_{x \in \mathbb{R}^n} F(x, u^k, p^k) & & \\
u_i^{k+1} &= u_i^k \varphi'(g_i(x^{k+1})/p_i^k) & & i = 1, \ldots, m \\
p_i^{k+1} &= \pi\, p_i^k & & i = 1, \ldots, m
\end{aligned}$$

Properties of the PBM method

Theory:

- $\{u^k\}_k$ generated by PBM is the same as for a Proximal Point algorithm applied to the dual problem (→ convergence proof)
- any cluster point of $\{x^k\}_k$ is an optimal solution to (CP)
- $f(x^k) \to f^*$ without $p_k \to 0$

Properties of the PBM method

Theory:

- $\{u^k\}_k$ generated by PBM is the same as for a Proximal Point algorithm applied to the dual problem ($\longrightarrow$ convergence proof)
- any cluster point of $\{x^k\}_k$ is an optimal solution to (CP)
- $f(x^k) \longrightarrow f^*$ without $p_k \longrightarrow 0$

Praxis:

- fast convergence thanks to the barrier branch of φ
- particularly suitable for large sparse problems
- robust, typically 10–15 outer iterations and 40–80 Newton steps

PBM in semidefinite programming

Problem: $$\min_{x\in\mathbb{R}^n} \left\{ b^T x : \mathcal{A}(x) \preccurlyeq 0 \right\}$$

Question: How can the matrix constraint

$$\mathcal{A}(x) \preccurlyeq 0 \quad (\mathcal{A} : \mathbb{R}^n \longrightarrow \mathbb{S}_d \text{ convex})$$

be treated by PBM approach ?

Idea: Find an *augmented Lagrangian* as follows:

$$F(x, U, p) = f(x) + \langle U, \Phi_p\left(\mathcal{A}(x)\right)\rangle_{\mathbb{S}_d}$$

PBM in semidefinite programming

Problem: $$\min_{x \in \mathbb{R}^n} \{b^T x : \mathcal{A}(x) \preccurlyeq 0\}$$

Question: How can the matrix constraint

$$\mathcal{A}(x) \preccurlyeq 0 \quad (\mathcal{A} : \mathbb{R}^n \longrightarrow \mathbb{S}_d \text{ convex})$$

be treated by PBM approach ?

Idea: Find an *augmented Lagrangian* as follows:

$$F(x, U, p) = f(x) + \langle U, \Phi_p(\mathcal{A}(x)) \rangle_{\mathbb{S}_d}$$

Notation:

$\langle A, B \rangle_{\mathbb{S}_d}$	$:= \mathrm{tr}\left(A^T B\right)$ *inner product* on $\mathbb{S}_d$
$\mathbb{S}_{d_+}$	$= \{A \in \mathbb{S}_d \mid A$ *positive semidefinite*$\}$
$U \in \mathbb{S}_{d_+}$	*matrix multiplier (dual variable)*
Φ_p	*penalty function* on $\mathbb{S}_d$

Given:
scalar valued penalty function φ satisfying $(\varphi_0) - (\varphi_5)$
matrix $A = S^\top \Lambda S$, where $\Lambda = \mathrm{diag}\ (\lambda_1, \lambda_2, \ldots, \lambda_d)^\top$

Define

$$A \overset{\Phi_p}{\longmapsto} S^T \begin{pmatrix} p\varphi\left(\frac{\lambda_1}{p}\right) & 0 & \ldots & 0 \\ 0 & p\varphi\left(\frac{\lambda_2}{p}\right) & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \ldots & 0 & p\varphi\left(\frac{\lambda_d}{p}\right) \end{pmatrix} S$$

$\longrightarrow$ any positive eigenvalue of A is “penalized” by φ

PBM algorithm for semidefinite problems

We have

$$\mathcal{A}(x) \preccurlyeq 0 \Longleftrightarrow \Phi_p(\mathcal{A}(x)) \preccurlyeq 0$$

and the corresponding *augmented Lagrangian*:

$$F(x, U, p) := f(x) + \langle U, \Phi_p(\mathcal{A}(x)) \rangle_{\mathbb{S}_d}$$

PBM algorithm:

$$\begin{aligned}
(i) \qquad & x^{k+1} = \arg\min_{x \in \mathbb{R}^n} F(x, U^k, p^k) \\
(ii) \qquad & U^{k+1} = D_{\mathcal{A}} \Phi_p(\mathcal{A}(x); U^k) \\
(iii) \qquad & p^{k+1} < p^k
\end{aligned}$$

PBM algorithm for semidefinite problems

The first idea may not be the best one:

- The matrix function Φ_p corresponding to φ is convex but may be **nonmonotone** on $\mathbb{H}_d(r,\infty)$ (right branch) $\longrightarrow$

$$\langle U, \Phi_p\left(\mathcal{A}(x)\right)\rangle_{\mathbb{S}_d}$$

may be **nonconvex**.

PBM algorithm for semidefinite problems

The first idea may not be the best one:

- The matrix function Φ_p corresponding to φ is convex but may be **nonmonotone** on $\mathbb{H}_d(r, \infty)$ (right branch) $\longrightarrow$

$$\langle U, \Phi_p(\mathcal{A}(x)) \rangle_{\mathbb{S}_d}$$

may be **nonconvex**.

- Complexity of Hessian assembling $\longrightarrow O(d^4 + d^3 n + d^2 n^2)$
Even for a **very sparse** structure the complexity can be $O(d^4)$!

n . . . number of variables

d . . . size of matrix constraint

Hessian:

$$\left[\nabla_{xx}\left\langle U,\Phi_p\left(\mathcal{A}(x)\right)\right\rangle_{\mathbb{S}_d}\right]_{i,j} =$$

$$\sum_{k=1}^{d}\Big(s_k(x)^\top A_i\Big[S(x)\Big(\left[\triangle^2\varphi(\lambda_l(x),\lambda_m(x),\lambda_k(x))\right]_{l,m=1}^{n}$$

$$\circ\left[S(x)^\top U S(x)\right]\Big)S(x)^\top\Big]A_j s_k(x)\Big)$$

- S : decomposition matrix of $\mathcal{A}(x)$
- s_k : *k*-th column of S
- $\triangle^i$ divided difference of *i*-th order
- $\mathcal{A}^* : \mathbb{S}_d \longrightarrow \mathbb{R}^n$ conjugate operator to $\mathcal{A}$

Construction of the penalty function, second idea

Find a penalty function φ which allows "direct" computation of Φ, its gradient and Hessian.

Construction of the penalty function, second idea

Find a penalty function φ which allows "direct" computation of Φ, its gradient and Hessian.

Example: $(\mathcal{A}(x) = \sum x_i A_i)$

$$\varphi(x) = x^2 \qquad \Rightarrow \qquad \Phi(A) = A^2$$

Then

$$\frac{\partial}{\partial x_i} \Phi(\mathcal{A}(x)) = \mathcal{A}(x) A_i + A_i \mathcal{A}(x)$$

and

$$\frac{\partial^2}{\partial x_i \partial x_j} \Phi(\mathcal{A}(x)) = A_j A_i + A_i A_j$$

Construction of the penalty function

The reciprocal barrier function in SDP: $(\mathcal{A}(x) = \sum x_i A_i)$

$$\varphi := \frac{1}{t-1} - 1$$

The corresponding matrix function is

$$\Phi(A) = (A - I)^{-1} - I$$

and we can show that

$$\frac{\partial}{\partial x_i}\Phi(\mathcal{A}(x)) = (A - I)^{-1} A_i (A - I)^{-1}$$

and

$$\frac{\partial^2}{\partial x_i \partial x_j}\Phi(\mathcal{A}(x)) = (A - I)^{-1} A_i (A - I)^{-1} A_j (A - I)^{-1}$$

Complexity of Hessian assembling:

- $O(d^3 n + d^2 n^2)$ for dense matrices
- $O(n^2 K^2)$ for sparse matrices
  ($K \ldots$ max. number of nonzeros in $A_i,\ i = 1, \ldots, n$)
- Compare to $O(d^4 + d^3 n + d^2 n^2)$ in the general case

$$\min_{x \in \mathbb{R}^n} \left\{ b^T x : \mathcal{A}(x) \preccurlyeq 0 \right\} \qquad \mathcal{A} : \mathbb{R}^n \longrightarrow \mathbb{S}_d$$

Handling sparsity

... essential for code efficiency

$$\min_{x\in\mathbb{R}^n} \{b^T x : \mathcal{A}(x) \preccurlyeq 0\} \qquad \mathcal{A} = \sum x_i A_i$$

Three basic sparsity types:

- many (small) blocks $\longrightarrow$ sparse Hessian (multi-load truss/material)

Handling sparsity

. . . essential for code efficiency

$$\min_{x\in\mathbb{R}^n} \{b^T x : \mathcal{A}(x) \preccurlyeq 0\} \qquad \mathcal{A} = \sum x_i A_i$$

Three basic sparsity types:

- many (small) blocks $\longrightarrow$ sparse Hessian (multi-load truss/material)
- few (large) blocks

Handling sparsity

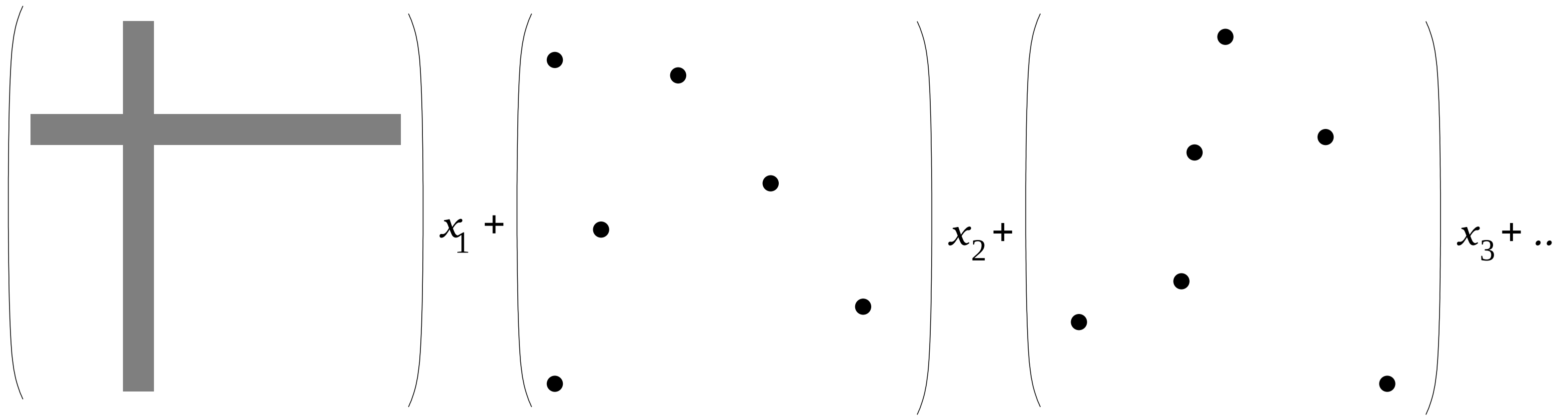
. . . essential for code efficiency

$$\min_{x\in\mathbb{R}^n} \{b^T x : \mathcal{A}(x) \preccurlyeq 0\} \qquad \mathcal{A} = \sum x_i A_i$$

Three basic sparsity types:

- many (small) blocks $\longrightarrow$ sparse Hessian (multi-load truss/material)
- few (large) blocks
  - $\mathcal{A}$ dense, A_i sparse (most of SDPLIB examples)

Handling sparsity



full version as inefficient as general sparse version

Recently, 3 matrix-matrix multiplication routines:

- full–full
- full–sparse
- sparse–sparse

Handling sparsity

essential for code efficiency

$$\min_{x \in \mathbb{R}^n} \{b^T x : \mathcal{A}(x) \preccurlyeq 0\} \qquad \mathcal{A} = \sum x_i A_i$$

Three basic sparsity types:

- many (small) blocks $\longrightarrow$ sparse Hessian (multi-load truss/material)
- few (large) blocks
  - $\mathcal{A}$ dense, A_i sparse (most of SDPLIB examples)
  - $\mathcal{A}$ sparse (truss design with buckling/vibration, maxG, . . .)

$$(A - I)^{-1} A_i (A - I)^{-1} A_j (A - I)^{-1}$$

Handling sparsity

Fast inverse computation of sparse matrices

$$Z = M^{-1}N$$

Explicite inverse of M: $O(n^3)$
Assume M is sparse and Cholesky factor L of M is sparse

$$Z_i = (L^{-1})^T L^{-1} N_i, \; i = 1, \ldots, n$$

Complexity: n times $nK \rightarrow O(n^2 K)$

Numerical results

New code called PENNON

Comparison with
DSDP by Benson and Ye
SDPT3 by Toh, Todd and Tütüncü
SeDuMi by Jos Sturm

SDPLIB problems: `http://www.nmt.edu/~sdplib/`

Numerical results

problem	variables	matrix	DSDP	SDPT3	PENNON
arch8	174	335	4	7	6
control7	136	45	114	48	82
control11	1596	165	1236	288	974
gpp500-4	501	500	28	39	21
mcp500-1	500	500	2	18	7
qap10	1021	101	19	8	16
ss30	132	426	10	18	20
theta6	4375	300	551	287	797
equalG11	801	801	139	156	102
equalG51	1001	1001	351	350	391
maxG11	800	800	6	54	25
maxG32	2000	2000	72	650	259

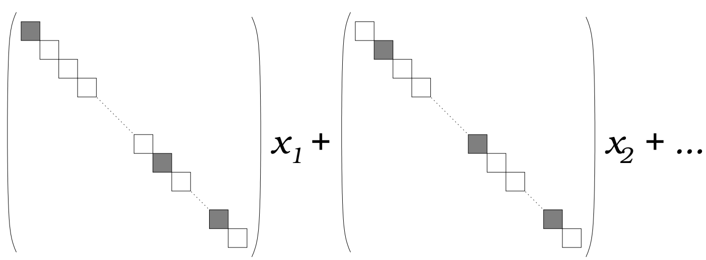
Multiple-load Free Material Optimization

After reformulation, discretization, further reformulation:

$$\min_{\alpha\in\mathbb{R},x\in(\mathbb{R}^n)^L}\left\{\alpha-\sum_{\ell=1}^{L}(c^\ell)^T x^\ell \mid \mathcal{A}_i(\alpha,x)\succeq 0\,,\, i=1,\ldots,m\right\}$$

Many (~ 5000) small (11–19) matrices.
Large dimension ($nL \sim 20\,000$)
In a standard form:

$$\min_{x\in(\mathbb{R}^n)^L}\left\{a^T x \mid \sum_{i=1}^{nL} x_i B_i \succeq 0\right\}$$



Examples from Mechanics

problem	no. of var.	size of matrix	DSDP	SDPT3	SeDuMi	PENNON
mater3	1439	3588	146	35	20	6
mater4	4807	12498	6269	295	97	29
mater5	10143	26820	36000	m	202	78
mater6	20463	56311	m	m	533	233

Truss design with free vibration control

Lowest eigenfrequency of the optimal structure should be bigger than a prescribed value

$$\min_{t,u} \sum t_i$$

$$\text{s.t.} \quad A(t)u = f$$

$$|\sigma| \leq \sigma_\ell \qquad (g(u) \leq c)$$

$$t \in T$$

min. eigenfrequency $\geq$ a given value

Formulated as SDP problem:

$$\min_t \sum t_i$$
$$\text{subject to} \quad A(t) - \lambda M(t) \succeq 0$$
$$\begin{pmatrix} c & f^T \\ f & A(t) \end{pmatrix} \succeq 0$$
$$t_i \geq 0, \quad i = 1, \ldots, n$$

where

$$A(t) = \sum t_i A_i \qquad A_i = \frac{E_i}{\ell_i^2} \gamma_i \gamma_i^T$$

$$M(t) = \sum t_i M_i \qquad M_i = c * \text{diag}(\gamma_i \gamma_i^T)$$

`truss` *test problems*

- `trto:` problems from single-load truss topology design. Normally formulated as LP, here reformulated as SDP for testing purposes.
- `vibra:` single load truss topology problems with a vibration constraint. The constraint guarantees that the minimal self-vibration frequency of the optimal structure is bigger than a given value.
- `buck:` single load truss topology problems with linearized global buckling constraint. Originally a nonlinear matrix inequality, the constraint should guarantee that the optimal structure is mechanically stable (does not buckle).

All problems characterized by sparsity of the matrix operator $\mathcal{A}$.

truss *test problems*

problem	n	m	DSDP	SDPT3	PENNON
trto3	544	321+544	11	19	17
trto4	1200	673+1200	134	124	106
trto5	3280	1761+3280	3125	1422	1484
buck3	544	641+544	44	43	39
buck4	1200	1345+1200	340	241	221
buck5	3280	3521+3280	10727	2766	3006
vibra3	544	641+544	52	45	34
vibra4	1200	1345+1200	596	294	191
vibra5	3280	3521+3280	25290	3601	2724

Benchmark tests by Hans Mittelmann:
`http://plato.la.asu.edu/bench.html`

Implemented on the NEOS server:
`http://www-neos.anl.gov`

Homepage:
`http://www2.am.uni-erlangen.de/∼kocvara/pennon/`
`http://www.penopt.com/`

Available with MATLAB interface through TOMLAB

`http://www.tomlab.biz`

When PCG helps (SDP) ?

Linear SDP, dense Hessian $\qquad A = \sum_{i=1}^{n} A_i, \; A_i \in \mathbb{R}^{m \times m}$

Complexity of Hessian evaluation

- $O(m_A^3 n + m_A^2 n^2)$ for dense matrices
- $O(m_A^2 n + K^2 n^2)$ for sparse matrices
  ($K \ldots$ max. number of nonzeros in $A_i, \; i = 1, \ldots, n$)

Complexity of Cholesky algorithm - linear SDP

- $O(n^3)$ $\qquad$ ($\ldots$from PCG we expect $O(n^2)$)

Problems with large n and small m:

CG better than Cholesky (expected)

Iterative algorithms

Conjugate Gradient method for $Hd = -g$, $H \in \mathbb{S}^n_+$

$\dots$
$\dots$
$y = Hx$ complexity $O(n^2)$
$\dots$
$\dots$

Exact arithmetics: “convergence” in n steps

$\longrightarrow$ overall complexity $O(n^3)$

Iterative algorithms

Conjugate Gradient method for $Hd = -g$, $H \in \mathbb{S}^n_+$

$\ldots$
$\ldots$
$y = Hx$ complexity $O(n^2)$
$\ldots$
$\ldots$

Exact arithmetics: “convergence” in n steps

$\longrightarrow$ overall complexity $O(n^3)$

Praxis: may be much worse (ill-conditioned problems)

Iterative algorithms

Conjugate Gradient method for $Hd = -g$, $H \in \mathbb{S}^n_+$

$\ldots$
$\ldots$
$y = Hx$ complexity $O(n^2)$
$\ldots$
$\ldots$

Exact arithmetics: "convergence" in n steps

$\longrightarrow$ overall complexity $O(n^3)$

Praxis: may be much worse (ill-conditioned problems)
may be much better $\longrightarrow$ *preconditioning*

Iterative algorithms

Conjugate Gradient method for $Hd = -g$, $H \in \mathbb{S}^n_+$

$\ldots$
$\ldots$
$y = Hx$ complexity $O(n^2)$
$\ldots$
$\ldots$

Exact arithmetics: "convergence" in n steps

$\longrightarrow$ overall complexity $O(n^3)$

Praxis: may be much worse (ill-conditioned problems)
may be much better $\longrightarrow$ *preconditioning*

Convergence theory: number of iterations depends on

- condition number
- distribution of eigenvalues

Iterative algorithms

Conjugate Gradient method for $Hd = -g$, $H \in \mathbb{S}^n_+$

$\dots$
$\dots$
$y = Hx$ complexity $O(n^2)$
$\dots$
$\dots$

Exact arithmetics: "convergence" in n steps

$\longrightarrow$ overall complexity $O(n^3)$

Praxis: may be much worse (ill-conditioned problems)
may be much better $\longrightarrow$ *preconditioning*

Convergence theory: number of iterations depends on

- condition number
- distribution of eigenvalues

Preconditioning: solve $M^{-1}Hd = M^{-1}g$ with $M \approx H^{-1}$

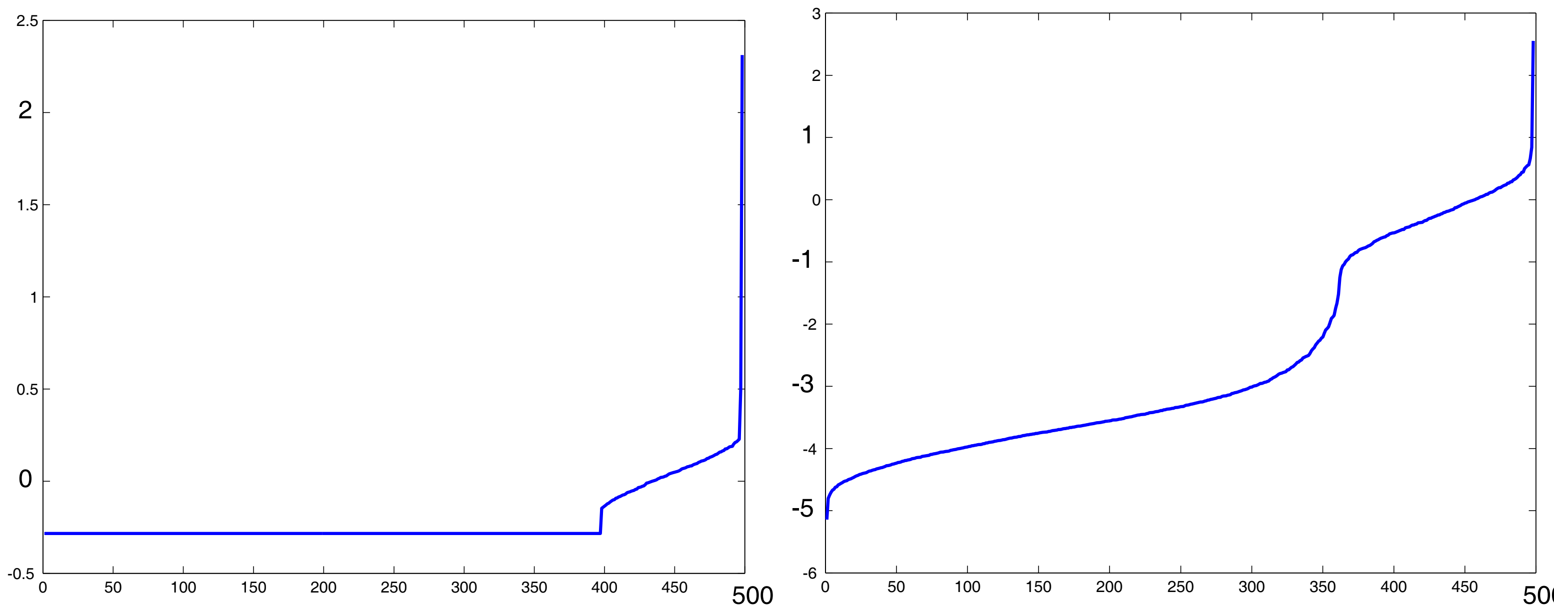
Conditioning of Hessian

Solve $Hd = -g$, H Hessian of

$$F(x,u,U,p,P) = f(x) + \langle U, \Phi_P(\mathcal{A}(x))\rangle_{\mathbb{S}_{m_A}}$$

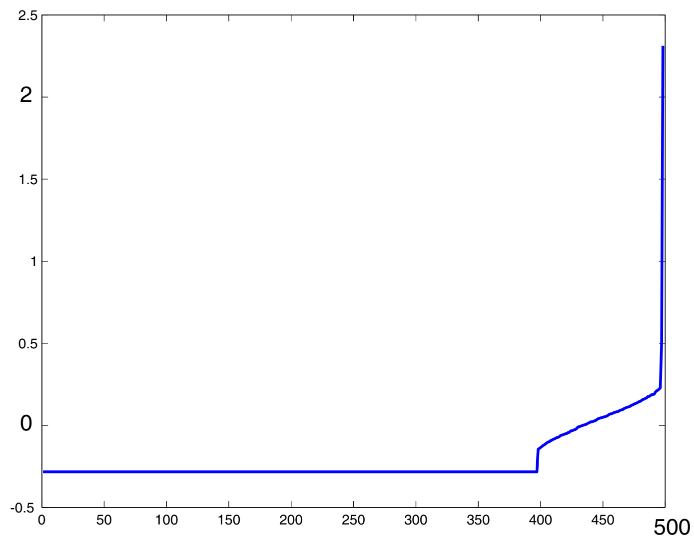
Condition number depends on P

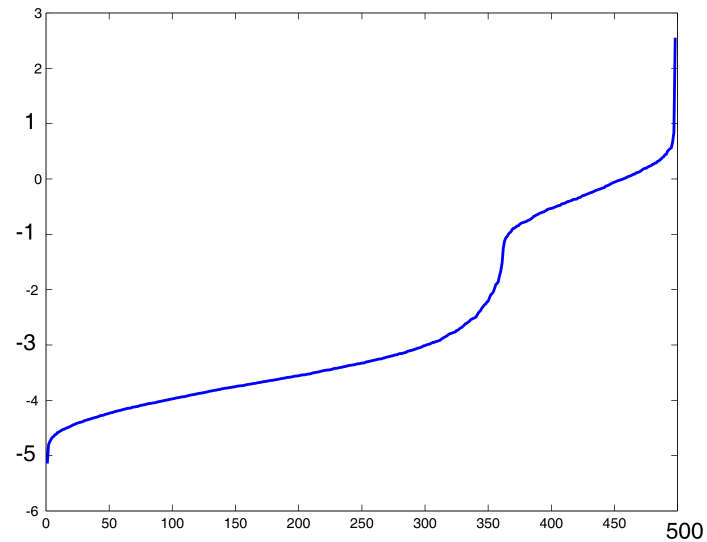
Example: problem Theta2 from SDPLIB ($n = 498$)



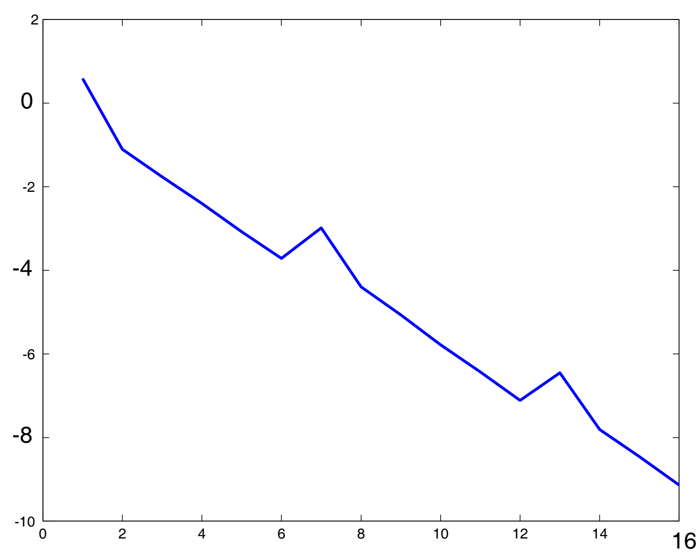
$\kappa_0 = 394$ $\kappa_{\text{opt}} = 4.9 \cdot 10^7$

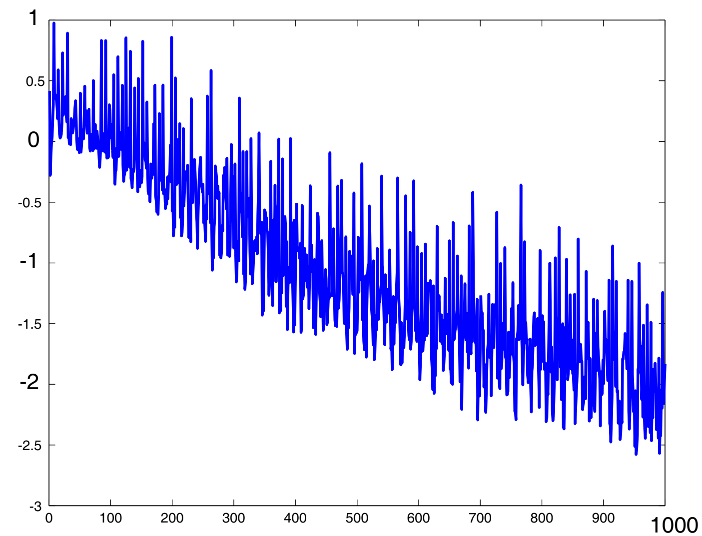
Theta2 from SDPLIB ($n = 498$)



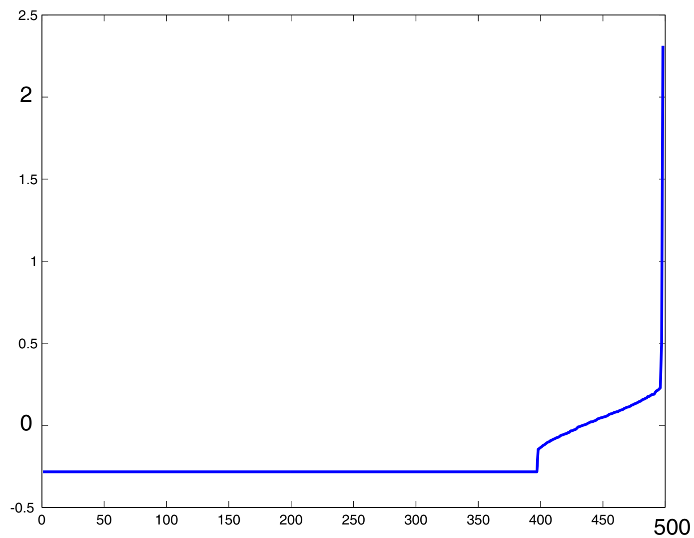


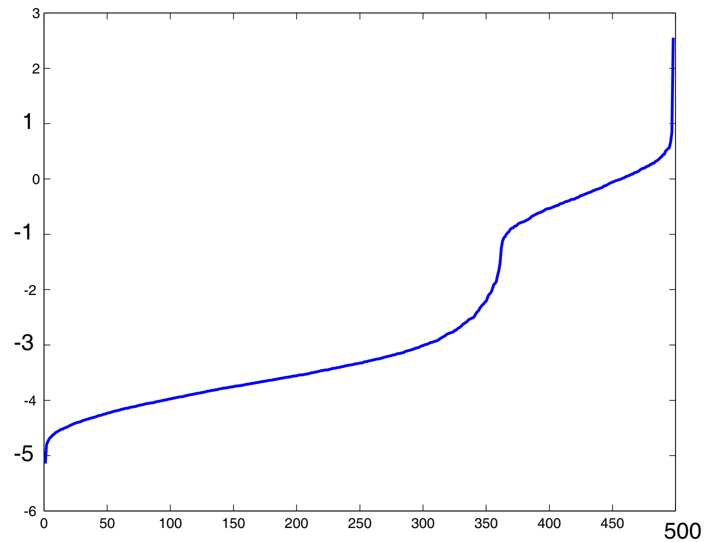
Behaviour of CG: testing $\|\boldsymbol{H}\boldsymbol{d} + \boldsymbol{g}\| / \|\boldsymbol{g}\|$



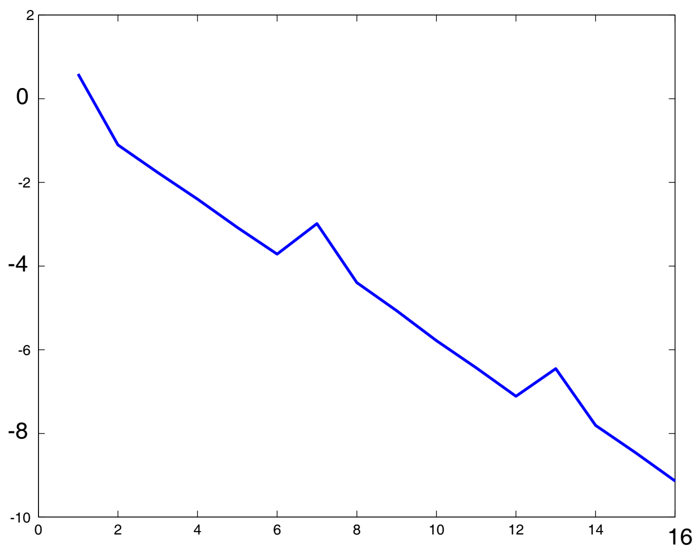


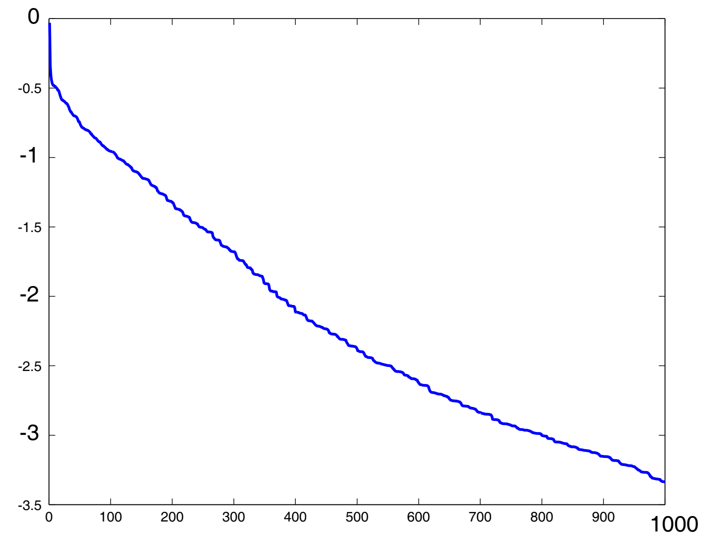
Theta2 from SDPLIB ($n = 498$)



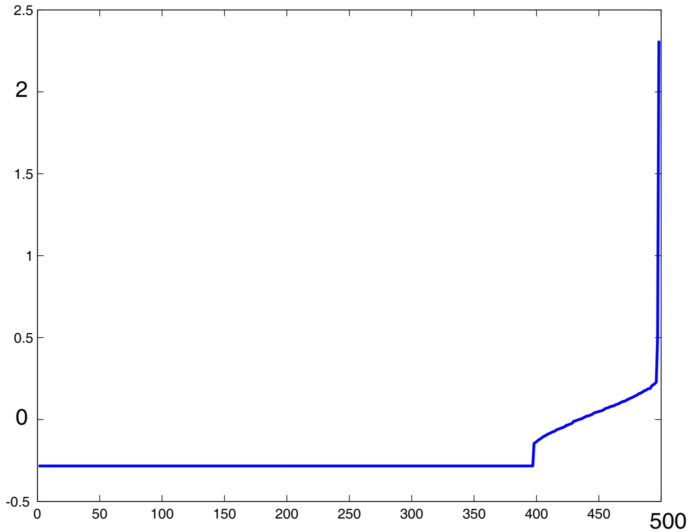


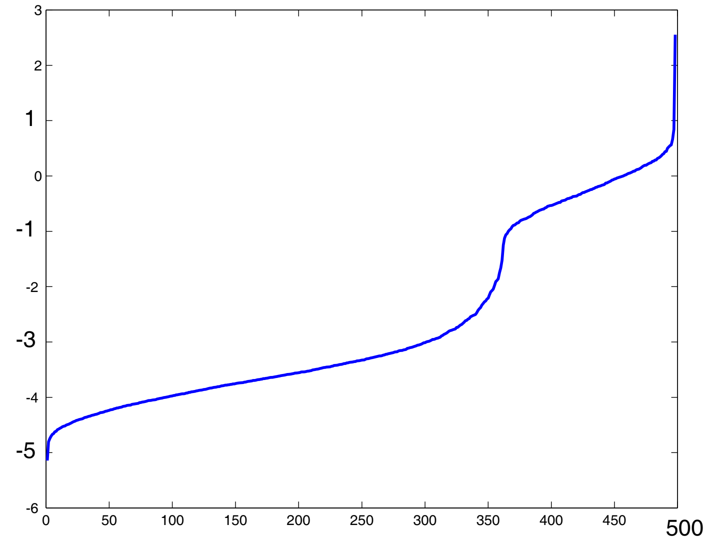
Behaviour of QMR: testing $\|\boldsymbol{H}\boldsymbol{d} + \boldsymbol{g}\| / \|\boldsymbol{g}\|$



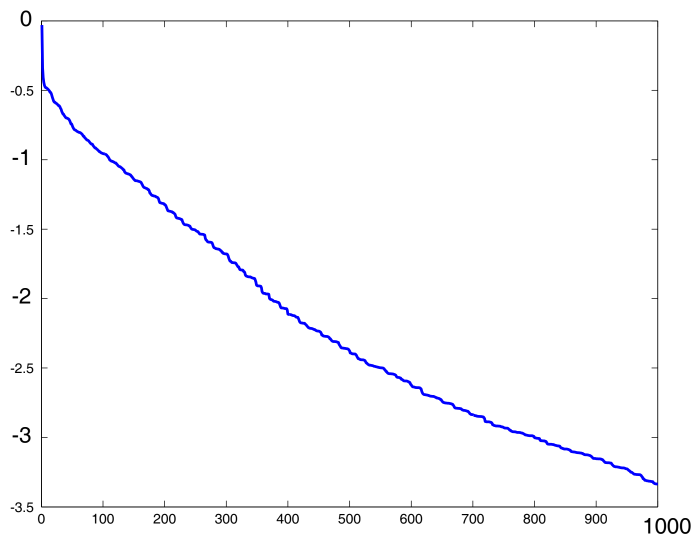


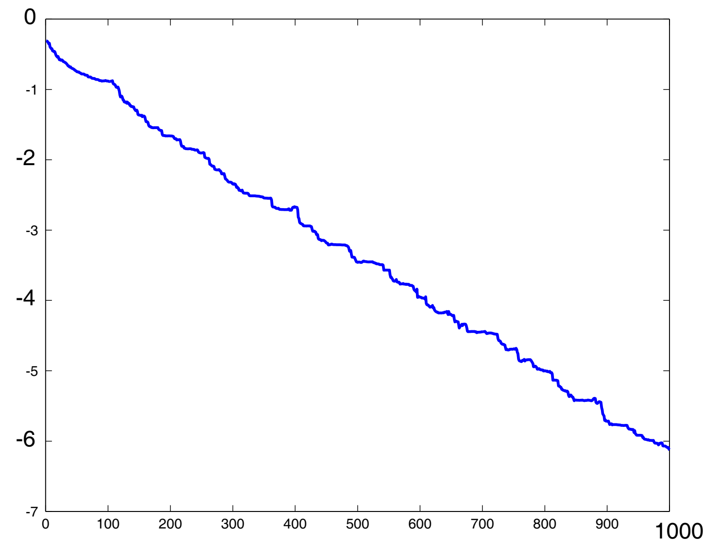
Theta2 from SDPLIB ($n = 498$)



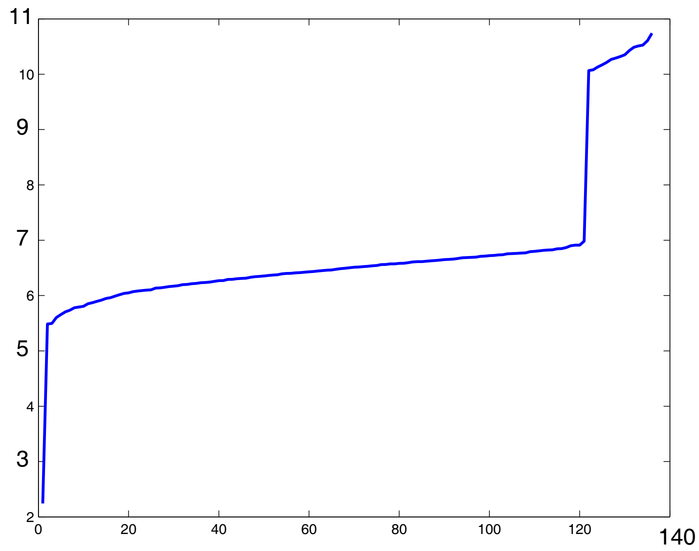


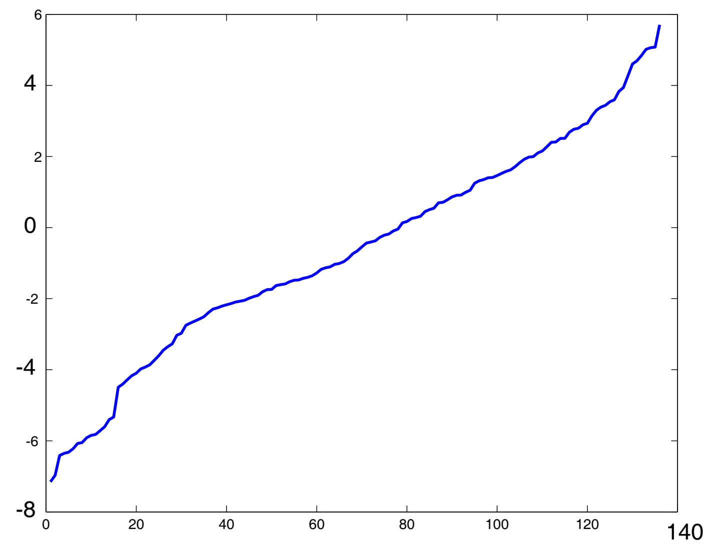
QMR: effect of preconditioning (for small $\boldsymbol{P}$)





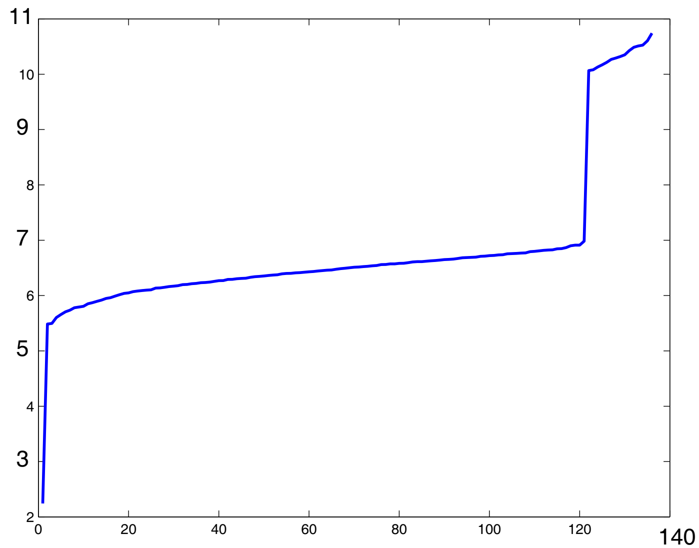
Control3 from SDPLIB ($n = 136$)

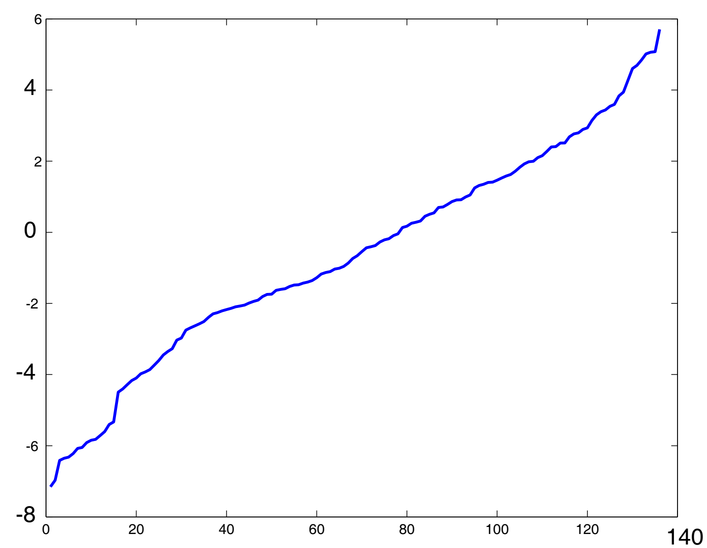




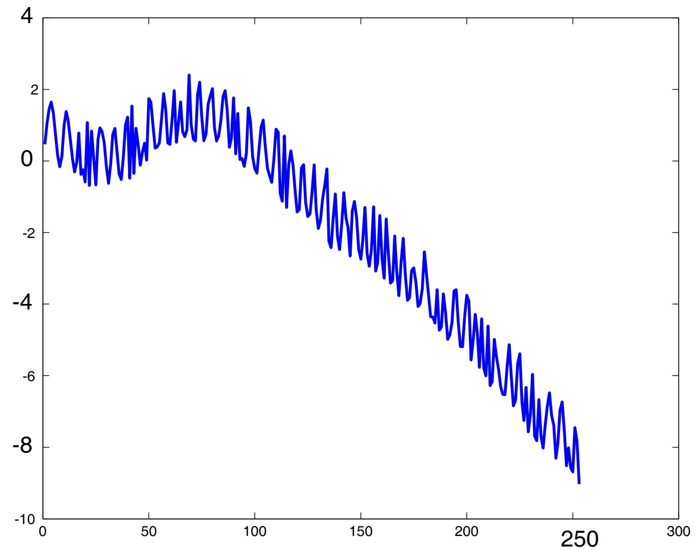
$$\boldsymbol{\kappa_0 = 3.1 \cdot 10^8} \qquad \boldsymbol{\kappa_{\rm opt} = 7.3 \cdot 10^{12}}$$

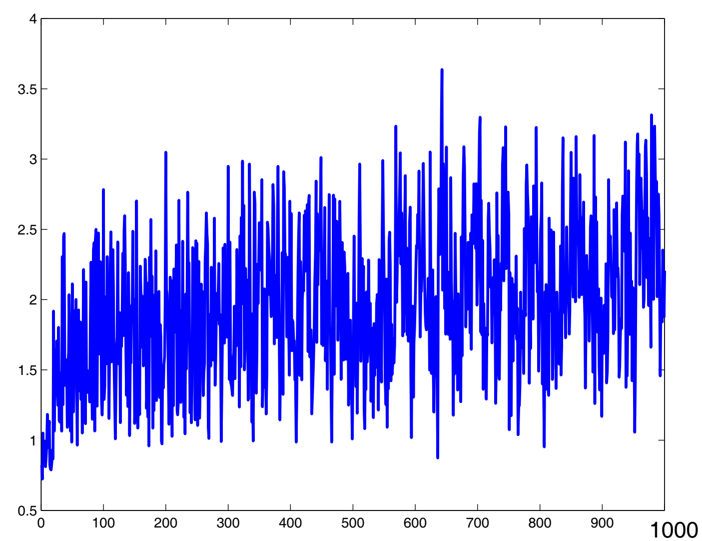
Control3 from SDPLIB ($n = 136$)



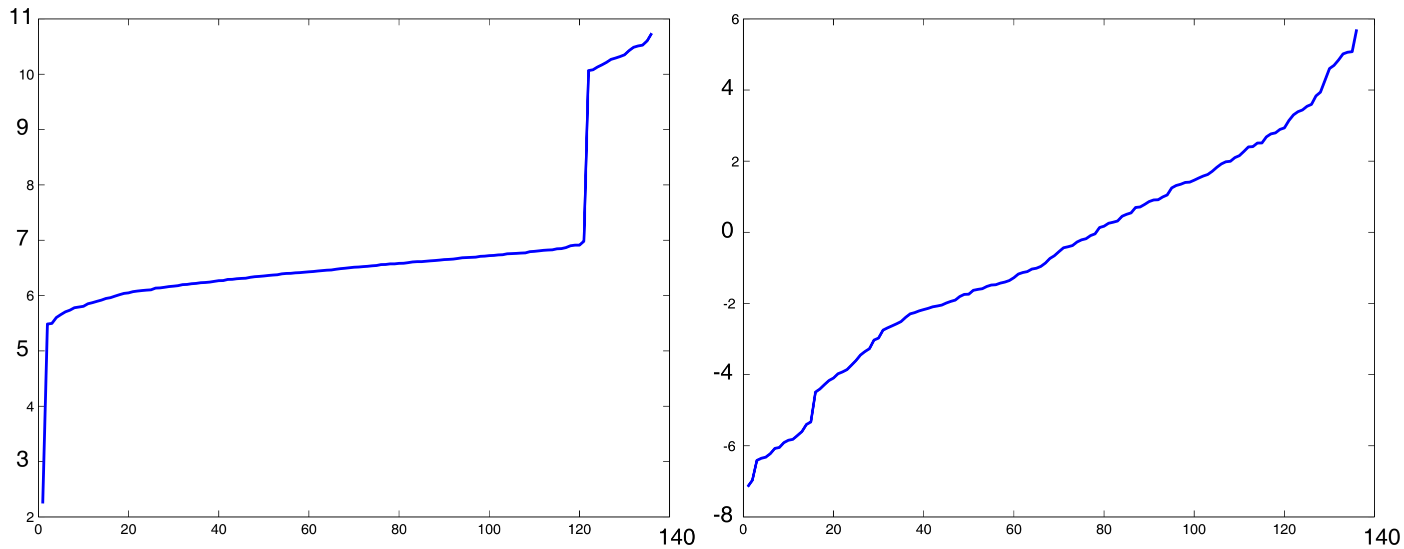


Behaviour of CG: testing $\|\boldsymbol{H}\boldsymbol{d} + \boldsymbol{g}\| / \|\boldsymbol{g}\|$

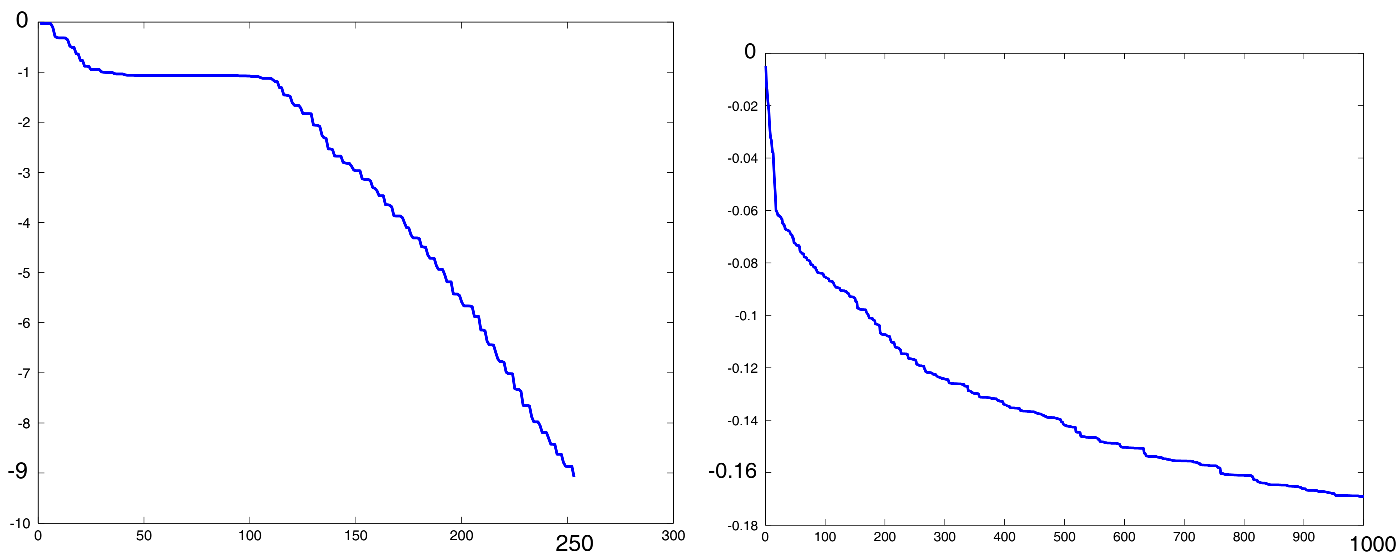




Control3 from SDPLIB ($n = 136$)



Behaviour of QMR: testing $\|\boldsymbol{Hd} + \boldsymbol{g}\| / \|\boldsymbol{g}\|$



Preconditioners

Should be:

- efficient (obvious but often difficult to reach)
- simple (low complexity)
- only use Hessian-vector product (NOT Hessian elements)

Preconditioners

Should be:

- efficient (obvious but often difficult to reach)
- simple (low complexity)
- only use Hessian-vector product (NOT Hessian elements)

- Diagonal
- Symmetric Gauss-Seidel
- L-BFGS (Morales-Nocedal, SIOPT 2000)
- A-inv (approximate inverse) (Benzi-Collum-Tuma, SISC 2000)

Preconditioners

Monteiro, O'Neil, Nemirovski (2004): Adaptive Ellipsoid Preconditioner

"Improves the CG performance on extremely ill-conditioned systems."

preconditioner:

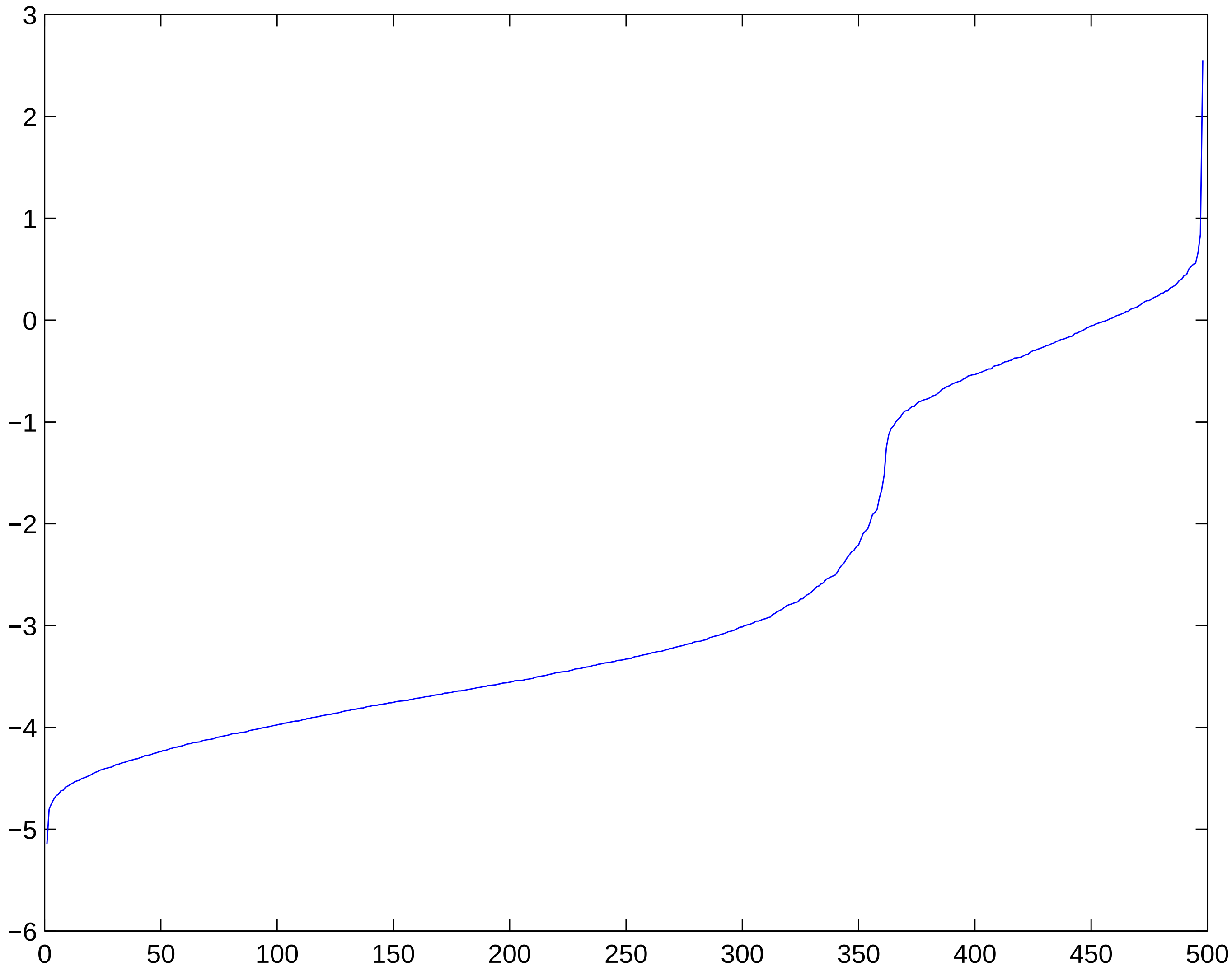
$$M = C_k C_k^T, \quad C_{k+1} \leftarrow \alpha C_k + \beta C_k p_k p_k^T, \quad C_1 = \gamma I$$

α, β, p_k . . . by matrix-vector products

VERY preliminary results (MATLAB implementation)

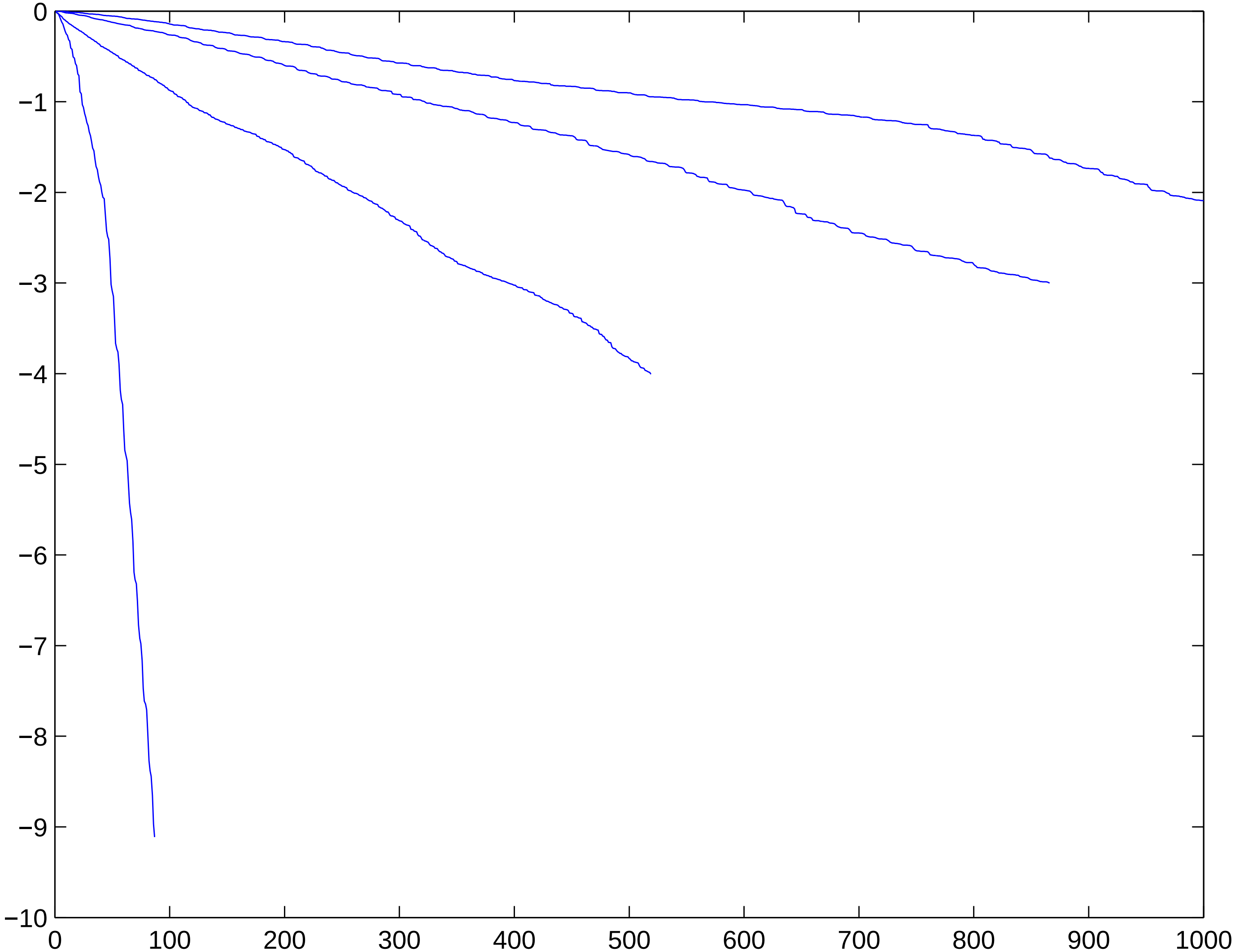
Preconditioners — example

Example: problem Theta2 from SDPLIB ($n = 498$)



Preconditioners — example

Example: problem Theta2 from SDPLIB ($n = 498$)



CG

CG-diag

CG-SGS

CG-MON

.

Hessian free methods

Use finite difference formula for Hessian-vector products:

$$\nabla^2 F(x_k)v \approx \frac{\nabla F(x_k + hv) - \nabla F(x_k)}{h}$$

with $h = (1 + \|x_k\|_2 \sqrt{\varepsilon})$

Complexity: Hessian-vector product = gradient evaluation
need for Hessian-vector-product type preconditioner

Limited accuracy (4–5 digits)

Test results: linear SDP, dense Hessian

Stopping criterium for PENNON

Exact Hessian: $\mathbf{10^{-7}}$ (7–8 digits in objective function)
Approximate Hessian: $\mathbf{10^{-4}}$ (4–5 digits in objective function)

Stopping criterium for CG/QMR ???

$$\boldsymbol{Hd = -g}, \text{ stop when } \|\boldsymbol{Hd + g}\|/\|\boldsymbol{g}\| \leq \boldsymbol{\epsilon}$$

Test results: linear SDP, dense Hessian

Stopping criterium for PENNON

Exact Hessian: 10^{-7} (7–8 digits in objective function)
Approximate Hessian: 10^{-4} (4–5 digits in objective function)

Stopping criterium for CG/QMR ???

$$Hd = -g, \text{ stop when } \|Hd + g\|/\|g\| \le \epsilon$$

Experiments: $\epsilon = 10^{-2}$ sufficient.
$\longrightarrow$ often very low (average) number of CG iterations

Complexity: $n^3 \to kn^2, k \approx 4 - 8$

Practice: effect not that strong, due to other complexity issues

Problems with large n and small m

Library of examples with large n and small m
(courtesy of Kim Toh)

CG-exact much better than Cholesky
CG-approx much better than CG-exact

problem	n	m	PENSDP	PENSDP (APCG)		SDPLR	
			CPU	CPU	CG	CPU	iter
ham_7_5_6	1793	128	126	4	52	1	113
ham_9_8	2305	512	423	210	66	46	222
ham_8_3_4	16129	256	81274	104	52	21	195
ham_9_5_6	53761	512		1984	71	71	102
theta42	200	5986	4722	51	269	393	11548
theta6	4375	300	2327	108	308	1221	20781
theta62	13390	300	68374	196	240	1749	16784
theta8	7905	400	11947	263	311	1854	15257
theta82	23872	400	m	650	267	4650	20653
theta83	39862	400	m	1715	277	7301	23017
theta10	12470	500	57516	492	278	4636	18814
theta102	37467	500	m	1948	340	12275	29083
theta103	62516	500	m	6149	421	17687	29483
theta104	87845	500	m	8400	269		
theta12	17979	600	t	843	240	8081	21338
keller4	5101	171	3264	52	432	244	8586
sanr200-0.7	6033	200	6664	52	278	405	12139

problem	n	m	PENSDP (APCG)		RENDL
theta83	39862	400	460	345	440
theta103	62516	500	1440	491	850
theta123	90020	600	5286	1062	1530