Towards the solution of large-scale nonlinear SDP problems

Michal Kočvara and Michael Stingl

UTIA AV ČR Prague and University of Erlangen-Nürnberg

Background

Can we use iterative methods (CG, QMR) to the solution of Newton systems coming from penalty/barrier methods applied to NLP — instead of Cholesky factorization?

Yes, but. . .

Background

Can we use iterative methods (CG, QMR) to the solution of Newton systems coming from penalty/barrier methods applied to NLP — instead of Cholesky factorization?

Yes, but. . .

- Sparse Cholesky often very efficient (BUT: dense columns)
- Condition number of the system increases
- NO GENERAL PRECONDITIONER

Background

Can we use iterative methods (CG, QMR) to the solution of Newton systems coming from penalty/barrier methods applied to NLP — instead of Cholesky factorization?

Yes, but...

Can we use iterative methods (CG, QMR) to the solution of Newton systems coming from penalty/barrier methods applied to SDP — instead of Cholesky factorization?
Hmmm...

Nakata-Fujisawa-Kojima (1999)
Choi-Ye (2000)
Lin-Saigal (BIT, 2000)
Toh-Kojima (SIOPT, 2002)
Toh (SIOPT, 2003)

The NLP-SDP problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \ & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \qquad i = 1, \ldots, m_g \\ & \mathcal{A}(x) \preccurlyeq 0 \end{aligned}$$

(NLP-SDP)

$f,\ g_i : \mathbb{R}^n \to \mathbb{R}$ smooth

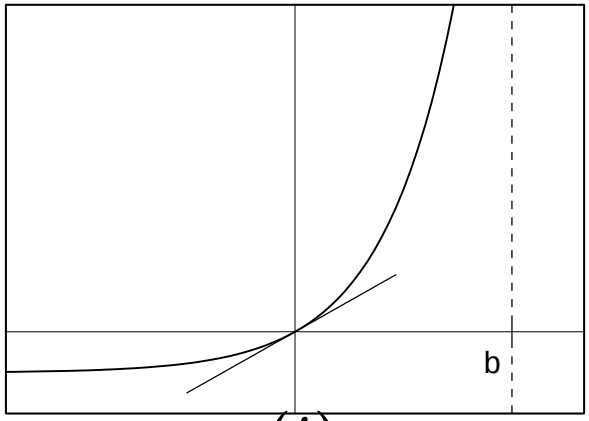
$\mathcal{A} : \mathbb{R}^n \to \mathbb{S}^{m_A}$ generally nonconvex

PENNON algorithm

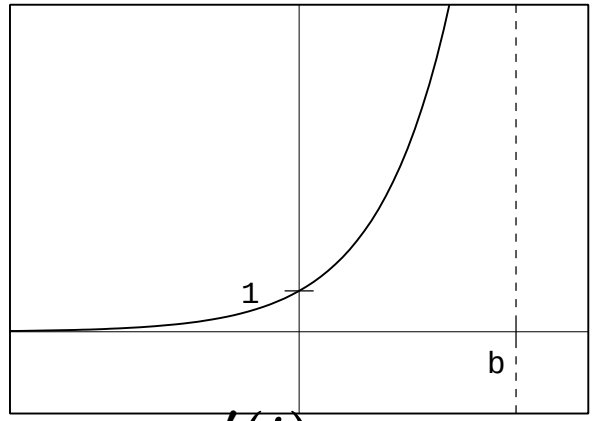
Based on the PBM method:

R. Polyak ’87, '92
Ben-Tal, Zibulevsky ’92, ’97
Breitfeld, Shanno ’94

“Penalty/barrier function:”



$\varphi(t)$



$\varphi'(t)$

PENNON algorithm

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

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

and

$$\mathcal{A}(x) \preccurlyeq 0 \iff \Phi_P(\mathcal{A}(x)) \preccurlyeq 0\,.$$

PENNON algorithm

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

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

and

$$\mathcal{A}(x) \preccurlyeq 0 \iff \Phi_P(\mathcal{A}(x)) \preccurlyeq 0\,.$$

The corresponding *augmented Lagrangian*:

$$F(x, u, U, p, P) = f(x) + \sum_{i=1}^{m_g} u_i p_i \varphi_g(g_i(x)/p_i) + \langle U, \Phi_P\left(\mathcal{A}(x)\right)\rangle_{\mathbb{S}_{m_A}}$$

PENNON algorithm

Augmented Lagrangian:

$$F(x,u,U,p,P) = f(x) + \sum_{i=1}^{m_g} u_i p_i \varphi_g(g_i(x)/p_i) + \langle U, \Phi_P(\mathcal{A}(x)) \rangle_{\mathbb{S}_{m_A}}$$

PENNON algorithm:

(i) Find $x^{k+1} =$ satisfying $\|\nabla_x F(x,u^k,U^k,p^k,P^k)\| \leq \varepsilon^k$

(ii) $u_i^{k+1} = u_i^k \varphi_g'(g_i(x^{k+1})/p_i^k), \quad i = 1,\ldots,m_g$

$U^{k+1} = D_{\mathcal{A}} \Phi_p(\mathcal{A}(x); U^k)$

(iii) $p_i^{k+1} < p_i^k, \; i = 1,\ldots,m_g$

$P^{k+1} < P^k$

PENNON algorithm

Augmented Lagrangian:

$$F(x,u,U,p,P) = f(x) + \sum_{i=1}^{m_g} u_i p_i \varphi_g(g_i(x)/p_i) + \langle U, \Phi_P(\mathcal{A}(x)) \rangle_{\mathbb{S}_{m_A}}$$

PENNON algorithm:

(i) Find $x^{k+1} =$ satisfying $\|\nabla_x F(x,u^k,U^k,p^k,P^k)\| \leq \varepsilon^k$

(ii) $u_i^{k+1} = u_i^k \varphi_g'(g_i(x^{k+1})/p_i^k), \quad i = 1,\ldots,m_g$

$U^{k+1} = D_{\mathcal{A}} \Phi_p(\mathcal{A}(x); U^k)$

(iii) $p_i^{k+1} < p_i^k, \; i = 1,\ldots,m_g$

$P^{k+1} < P^k$

Step (i): (modified) Newton's method (or TR)

Complexity issues

Complexity of Hessian computation - linear SDP:

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

Complexity of augm. Lagrangian evaluation - linear SDP:

- $O(m_A^3)$ for dense data matrices
- $O(m_A^2 \kappa)$ for sparse data matrices
  (κ ... max. number of nonzeros in L_i, $(A_i - I) = L_i L_i^T$,
  $i = 1, \ldots, n$)

Complexity of Cholesky algorithm - linear SDP:

- $O(n^3)$ for dense Hessians
- $O(n^{2.\cdots})$ for sparse Hessians

Nonlinear SDP: complexity

Structural optimization with stability constraint (nonconvex SDP)

$$\min_{\rho} W(\rho)$$

subject to

$$\begin{pmatrix} c & f^T \\ f & K(\rho) \end{pmatrix} \succeq 0$$

$$\rho_i \geq 0, \quad i = 1, \ldots, n$$

$$K(\rho) - \tilde{G}(\rho) \succeq 0$$

Nonlinear SDP: complexity

Structural optimization with stability constraint (nonconvex SDP)

$$\min_{\rho} W(\rho)$$

subject to

$$\begin{pmatrix} c & f^T \\ f & K(\rho) \end{pmatrix} \succeq 0$$

$$\rho_i \geq 0, \quad i = 1, \ldots, n$$

$$K(\rho) - \tilde{G}(\rho) \succeq 0$$

CPU: $O(k^2 * d^2 * n^3)$ for one Hessian evaluation

Nonlinear SDP: complexity

Structural optimization with stability constraint (nonconvex SDP)

$$\min_{\rho} W(\rho)$$

subject to

$$\begin{pmatrix} c & f^T \\ f & K(\rho) \end{pmatrix} \succeq 0$$

$$\rho_i \geq 0, \quad i = 1, \ldots, n$$

$$K(\rho) - \tilde{G}(\rho) \succeq 0$$

CPU: $O(k^2 * d^2 * n^3)$ for one Hessian evaluation

Pentium 4, 2.4GHz, ∼100 Newton steps:

400 elements . . . 8 h 45 min, 1000 elements . . . ∼130 hours

Nonlinear SDP: complexity

Structural optimization with stability constraint (nonconvex SDP)

$$\min_{\rho} W(\rho)$$

subject to

$$\begin{pmatrix} c & f^T \\ f & K(\rho) \end{pmatrix} \succeq 0$$

$$\rho_i \geq 0, \quad i = 1, \ldots, n$$

$$K(\rho) - \tilde{G}(\rho) \succeq 0$$

CPU: $O(k^2 * d^2 * n^3)$ for one Hessian evaluation

Pentium 4, 2.4GHz, ∼100 Newton steps:

400 elements . . . 8 h 45 min, 1000 elements . . . ∼130 hours

linear SDP (400 elements) . . . 6 min 20 sec

Motivation for CG

Use iterative solvers to improve:

Complexity of Cholesky algorithm - linear SDP:

- $O(n^3)$ for dense Hessians $\longrightarrow O(n^2)$
- $O(n^{2.\cdots})$ for sparse Hessians $\longrightarrow O(n^2)$

Motivation for CG

Use iterative solvers to improve:

Complexity of Cholesky algorithm - linear SDP:

- $O(n^3)$ for dense Hessians $\longrightarrow O(n^2)$
- $O(n^{2.\cdots})$ for sparse Hessians $\longrightarrow O(n^2)$

. . . too ambicious?

Motivation for CG

Use iterative solvers to improve:

Complexity of Cholesky algorithm - linear SDP:

- $O(n^3)$ for dense Hessians $\longrightarrow O(n^2)$
- $O(n^{2.\cdots})$ for sparse Hessians $\longrightarrow O(n^2)$

. . . too ambicious?

Complexity of Hessian assembling - nonlinear SDP:

- $O(n^3)$ for dense data matrices

Motivation for CG

Use iterative solvers to improve:

Complexity of Cholesky algorithm - linear SDP:

- $O(n^3)$ for dense Hessians $\longrightarrow O(n^2)$
- $O(n^{2.\cdots})$ for sparse Hessians $\longrightarrow O(n^2)$

. . . too ambicious?

Complexity of Hessian assembling - nonlinear SDP:

- $O(n^3)$ for dense data matrices

. . . using approximate Hessian-vector products

Iterative algorithms

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

$\ldots$
$\ldots$
$\boldsymbol{y = Hx}$
$\ldots$
$\ldots$

complexity $\boldsymbol{O(n^2)}$

Exact arithmetics: “convergence” in $\boldsymbol{n}$ steps

$\longrightarrow$ overall complexity $\boldsymbol{O(n^3)}$

Iterative algorithms

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

$\ldots$
$\ldots$
$\boldsymbol{y = Hx}$
$\ldots$
$\ldots$

complexity $\boldsymbol{O(n^2)}$

Exact arithmetics: "convergence" in $\boldsymbol{n}$ steps

$\longrightarrow$ overall complexity $\boldsymbol{O(n^3)}$

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

Iterative algorithms

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

$\ldots$
$\ldots$
$\boldsymbol{y = Hx}$
$\ldots$
$\ldots$

complexity $\boldsymbol{O(n^2)}$

Exact arithmetics: "convergence" in $\boldsymbol{n}$ steps

$\longrightarrow$ overall complexity $\boldsymbol{O(n^3)}$

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

Iterative algorithms

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

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

Exact arithmetics: "convergence" in $\boldsymbol{n}$ steps

$\longrightarrow$ overall complexity $\boldsymbol{O(n^3)}$

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 $\boldsymbol{Hd = -g}$, $\boldsymbol{H} \in \mathbb{S}^{\boldsymbol{n}}_{+}$

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

Exact arithmetics: "convergence" in $\boldsymbol{n}$ steps

$\longrightarrow$ overall complexity $\boldsymbol{O(n^3)}$

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 $\boldsymbol{M^{-1}Hd = M^{-1}g}$ with $\boldsymbol{M \approx H^{-1}}$

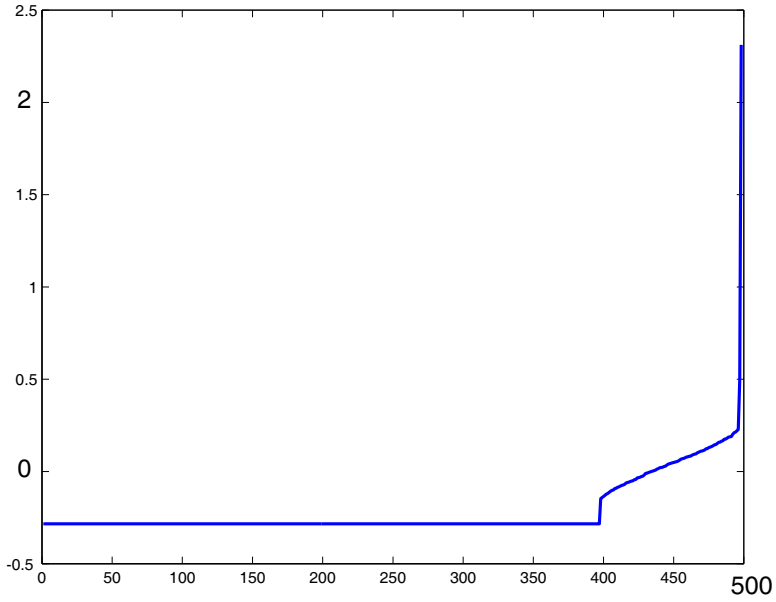
Conditioning of Hessian

Solve $Hd = -g$, H Hessian of

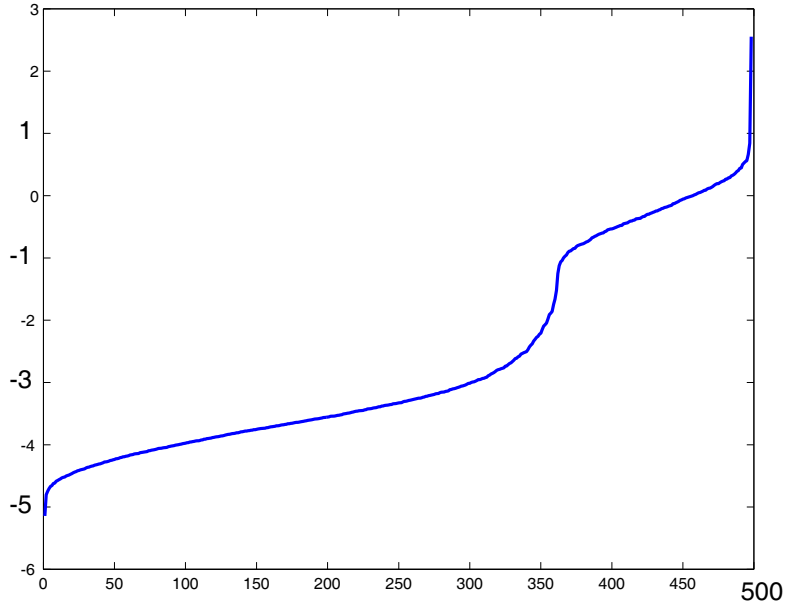
$$F(x, u, U, p, P) = f(x) + \langle U, \Phi_P(\mathcal{A}(x)) \rangle_{\mathbb{S}_{m_A}}$$

Condition number depends on P

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

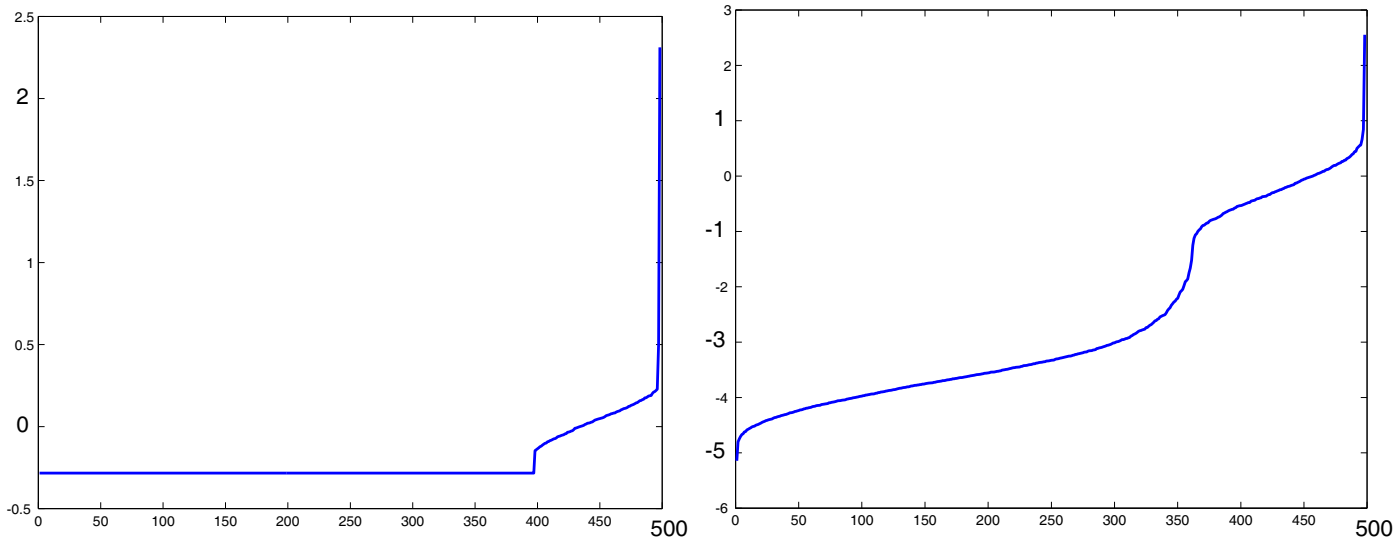


$\kappa_0 = 394$

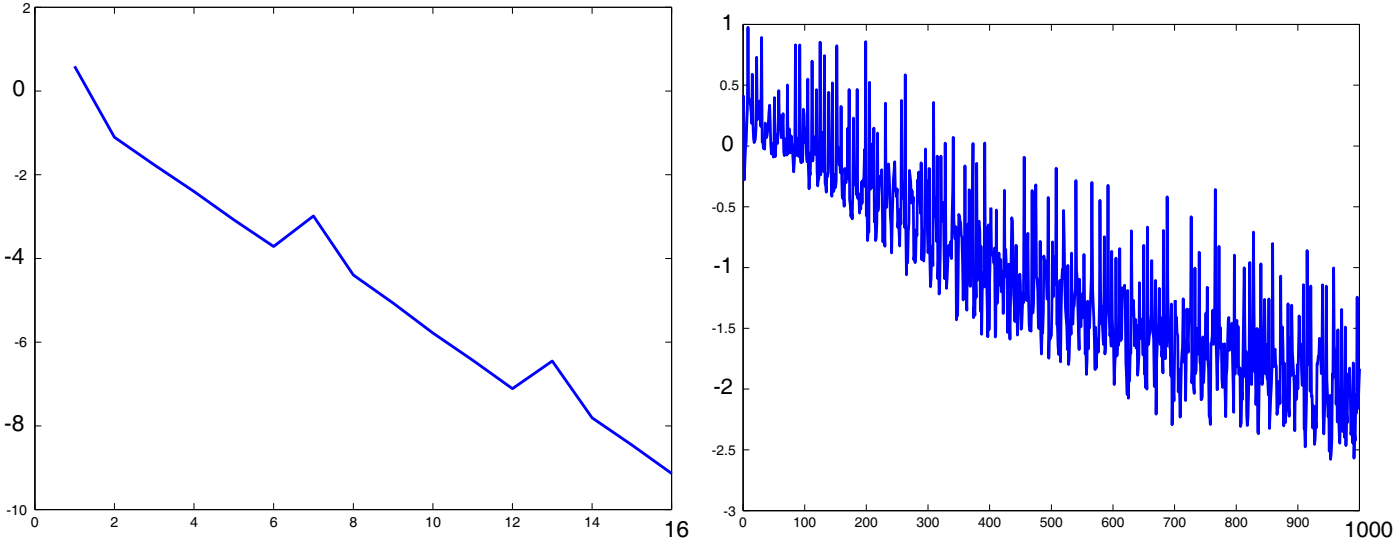


$\kappa_{\text{opt}} = 4.9 \cdot 10^7$

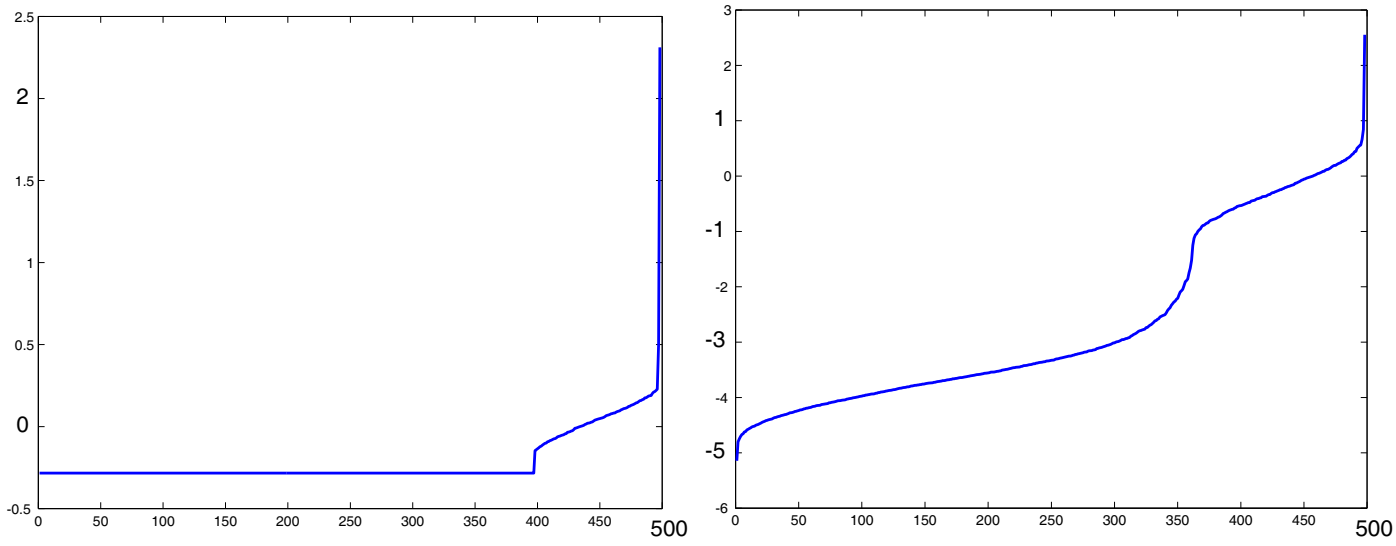
Theta2 from SDPLIB ($n = 498$)



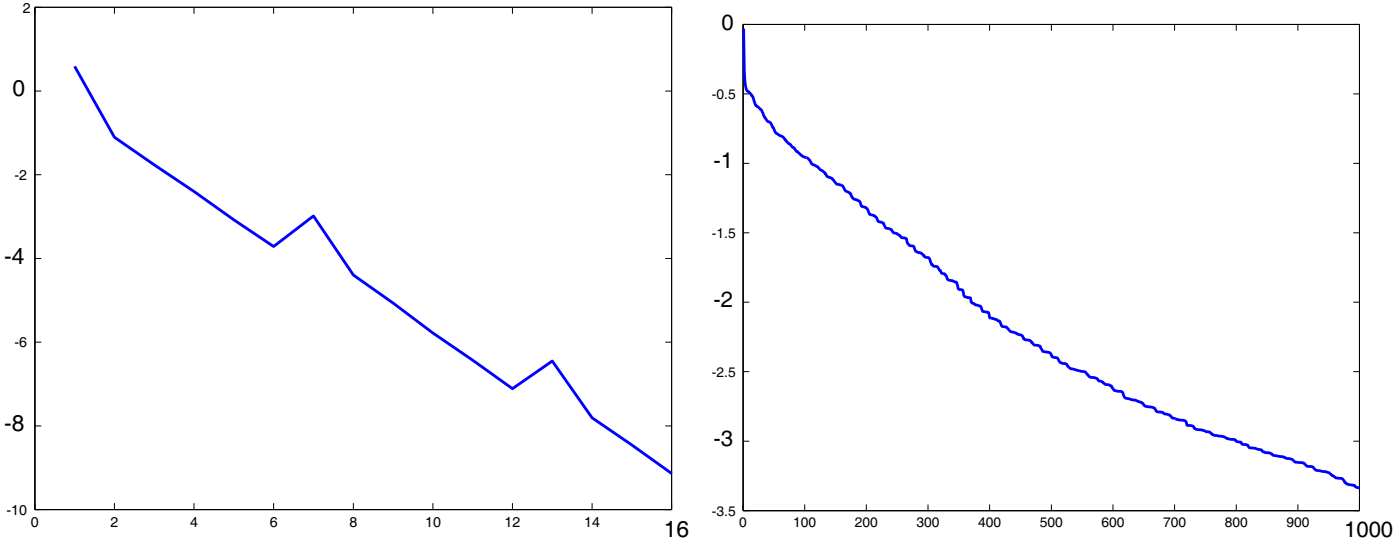
Behaviour of CG: testing $\|\boldsymbol{H}\boldsymbol{d} + \boldsymbol{g}\|/\|\boldsymbol{g}\|$



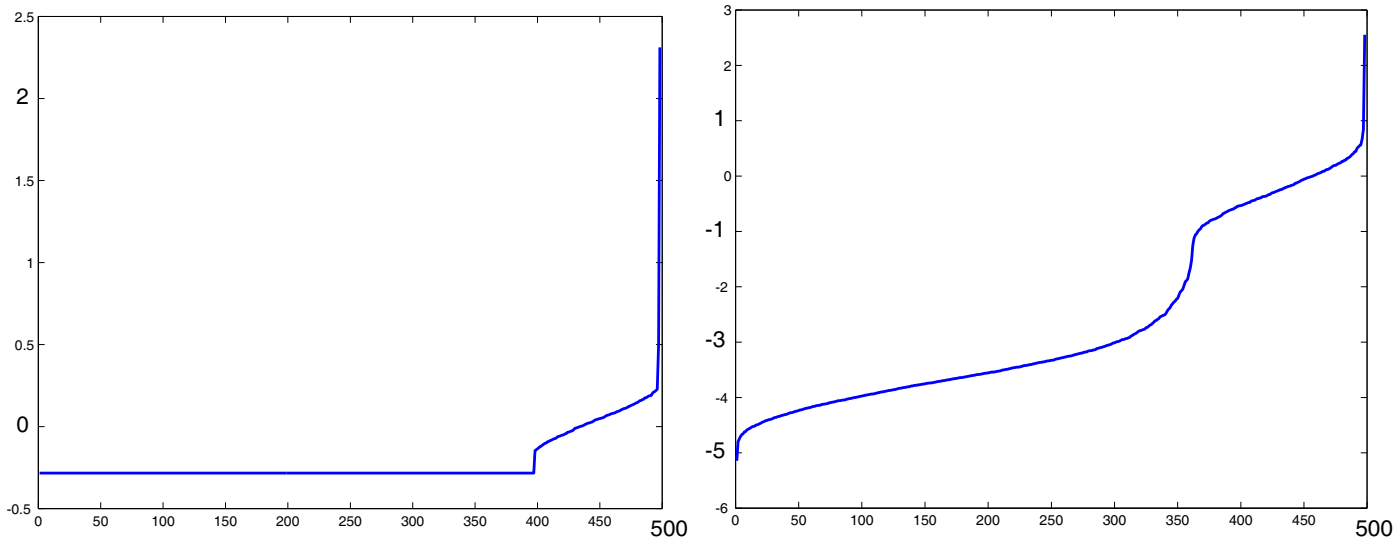
Theta2 from SDPLIB ($n = 498$)



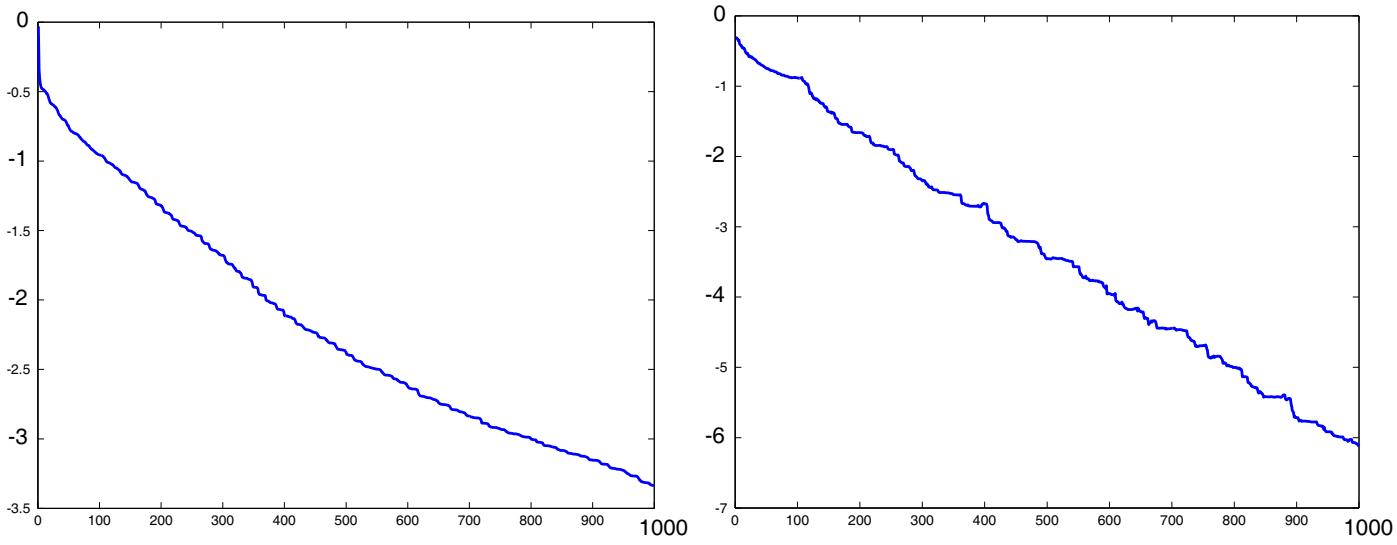
Behaviour of QMR: testing $\|\boldsymbol{H}\boldsymbol{d} + \boldsymbol{g}\|/\|\boldsymbol{g}\|$



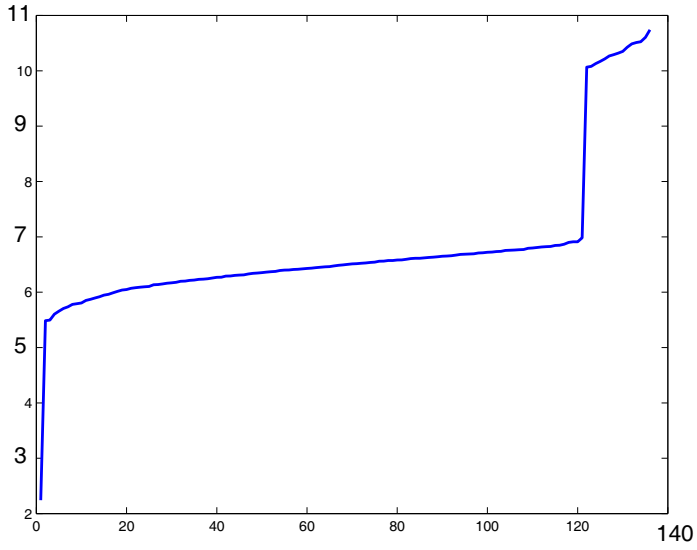
Theta2 from SDPLIB ($n = 498$)

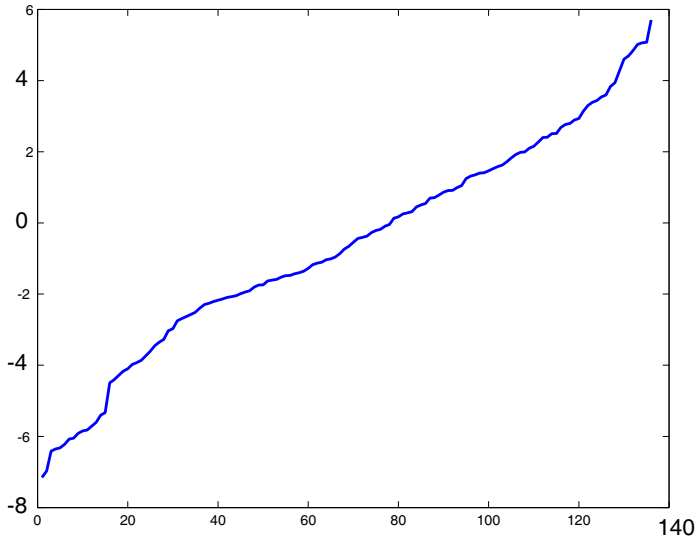


QMR: effect of preconditioning (for small P)



Control3 from SDPLIB ($n = 136$)

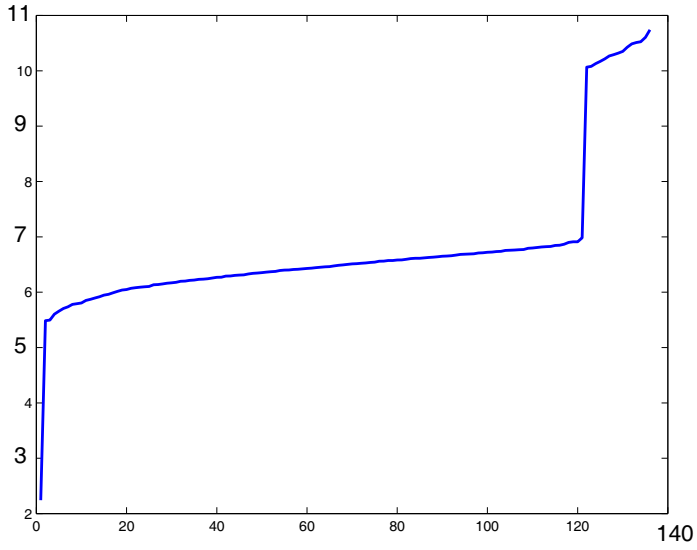


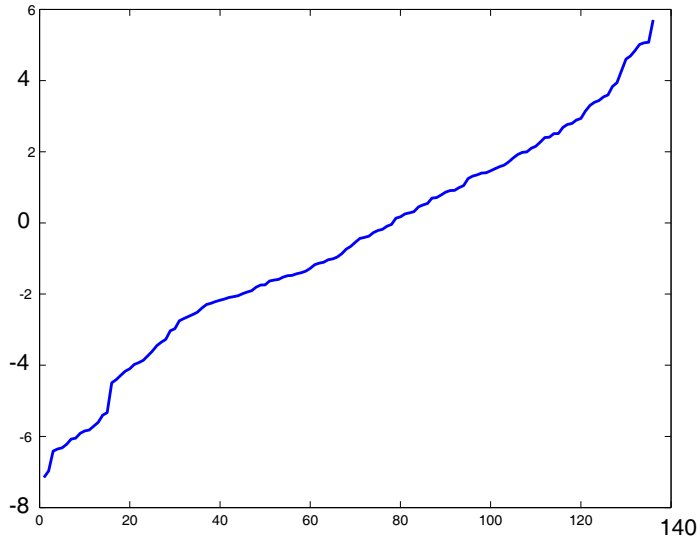


$$\kappa_0 = 3.1 \cdot 10^8$$

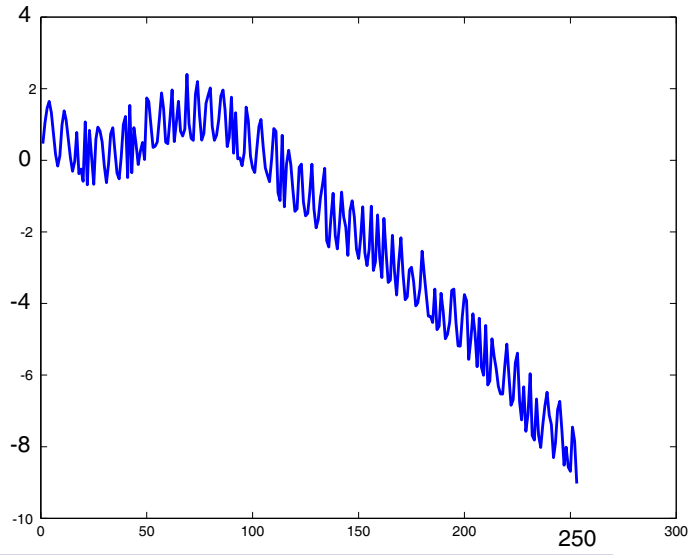
$$\kappa_{\text{opt}} = 7.3 \cdot 10^{12}$$

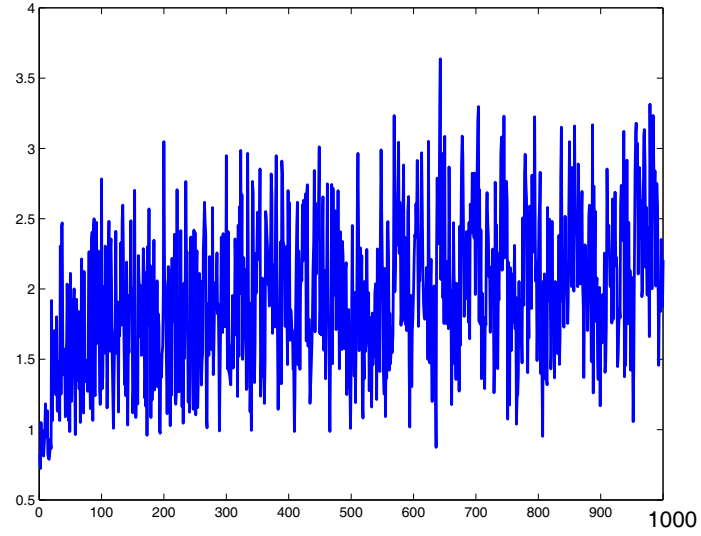
Control3 from SDPLIB ($n = 136$)



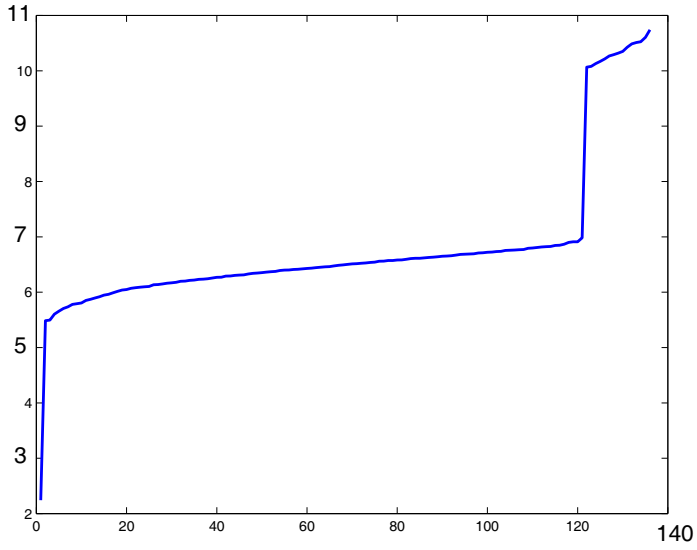


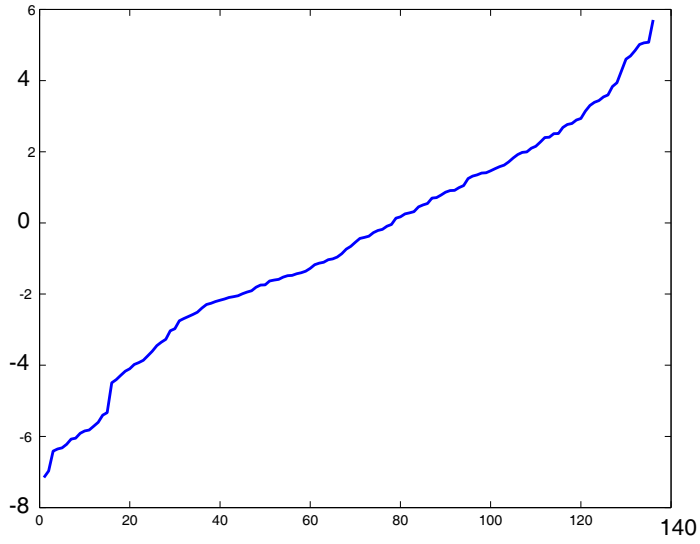
Behaviour of CG: testing $\|\boldsymbol{H}\boldsymbol{d} + \boldsymbol{g}\|/\|\boldsymbol{g}\|$



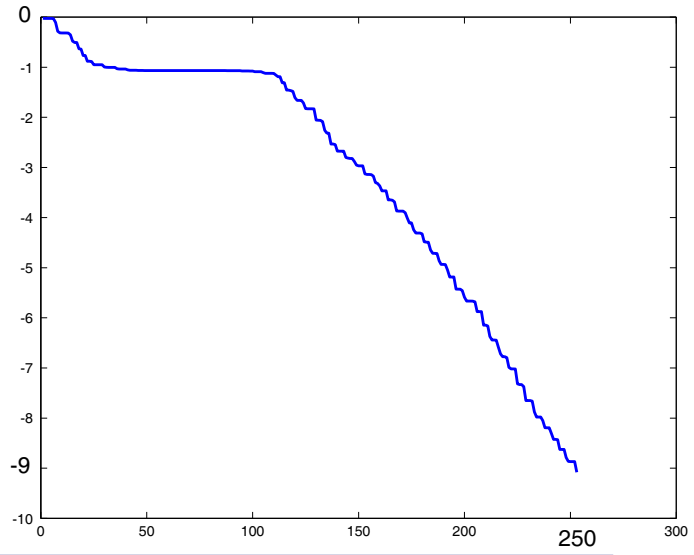


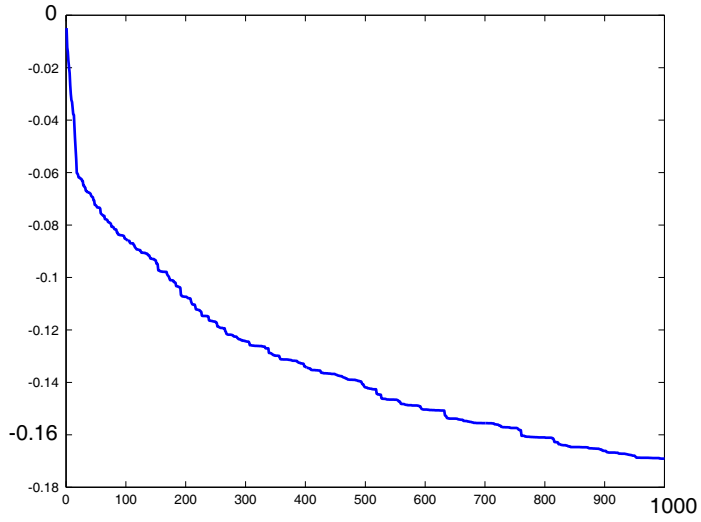
Control3 from SDPLIB ($n = 136$)





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





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

$$M = \mathbf{diag}\,(H)$$

simple, not (considered) very efficient

Preconditioners

Should be:

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

Symmetric Gauss-Seidel

$$M = L^T D^{-1} L \quad \text{where} \quad H = D - L - L^T$$

relatively efficient, a bit too costly

Preconditioners

Should be:

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

L-BFGS (Morales-Nocedal, SIOPT 2000)
–start with CG (no precond.)
–use CG iterations as *correction pairs* $\longrightarrow$ build M using L-BFGS
–next Newton step $\longrightarrow$ use M as preconditioner
–from CG iterations build new M

relatively inexpensive (16–32 correction pairs)

mixed success

Preconditioners

Should be:

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

A-inv (approximate inverse) (Benzi-Collum-Tuma, SISC 2000)

$$M = ZD^{-1}Z^T \approx H^{-1}$$

Z sparse approximation of Cholesky factor L^{-1}
computed directly from H by incomplete $H-$orthogonalization
small elements dropped to prevent (introduce) sparsity
uses only Hessian-vector products

relatively expensive, dependent on (sensitive to) the dropping parameter

efficient (often)

Test results: linear SDP, dense Hessian

Stopping criterium for PENNON

$$10^{-7} \rightarrow 10^{-4} \quad \text{(4–5 digits in objective function)}$$

??? Stopping criterium for CG/QMR ???

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

Test results: linear SDP, dense Hessian

Stopping criterium for PENNON

$$10^{-7} \rightarrow 10^{-4} \quad \text{(4–5 digits in objective function)}$$

??? Stopping criterium for CG/QMR ???

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

Experiments: $\epsilon = 10^{-2}$ sufficient !

$\rightarrow$ often very low (average) number of CG iterations

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

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

	time	Nwt	time	Nwt	time	Nwt	time	Nwt	time	Nwt
	pensdp		PCG(BFGS)		PCG(0)		PCG(SGS)		QMR(AINV)	
control4	12	101	19	83	22	79	24	101	19	90
control5	35	86	45	65	63	69	78	95	61	87
mcp250-1	5	48	3	36	4	36	4	36	6	36
mcp500-1	25	53	16	38	16	38	17	38	22	38
theta3	58	52	20	57	19	52	19	46	22	47
theta4	307	57	64	62	59	51	72	54	75	51
theta5	761	62	166	69	151	58	576	62	234	56
theta6	2910	58	294	53	332	52	476	61	440	56
equalG11	653	39	756	41	744	41	838	48	608	41
maxG11	91	59	51	37	54	38	55	38	59	44
maxG32	997	57	531	40	561	44	589	45	526	43
maxG51	852	73	680	69	643	59	652	60	543	54
qpG11	163	37	183	41	169	41	156	41	153	41
thetaG11	839	71	446	104	563	119	***		399	91

			time	time/iter	time	time/iter	time	time/iter	time	time/iter
	n	m	pensdp		PCG(BFGS)		PCG(0)		QMR(AINV)	
control4	231	60	12	0.12	19	0.23	22	0.28	19	0.21
control5	351	75	35	0.41	45	0.69	63	0.91	61	0.70
mcp250-1	250	250	5	0.10	3	0.08	4	0.11	6	0.17
mcp500-1	500	500	25	0.47	16	0.42	16	0.42	22	0.58
theta3	1106	150	58	1.12	20	0.35	19	0.37	22	0.47
theta4	1949	200	307	5.39	64	1.03	59	1.16	75	1.47
theta5	3038	250	761	12.27	166	2.41	151	2.60	234	4.18
theta6	4375	300	2910	50.17	294	5.55	332	6.38	440	7.86
equalG11	801	801	653	16.74	756	18.44	744	18.15	608	14.83
maxG11	800	800	91	1.54	51	1.38	54	1.42	59	1.34
maxG32	2000	2000	997	17.49	531	13.28	561	12.75	526	12.23
maxG51	1000	1000	852	11.67	680	9.86	643	10.90	543	10.06
qpG11	800	1600	163	4.41	183	4.46	169	4.12	153	3.73
thetaG11	2401	801	839	11.82	446	4.29	563	4.73	399	4.38

CG iter / Nwt. iter

	n	m	BFGS	none	SGS	AINV
control4	231	60	145	269	130	166
control5	351	75	187	420	196	255
mcp250-1	250	250	4	6	5	4
mcp500-1	500	500	4	6	5	4
theta3	1106	150	4	4	3	6
theta4	1949	200	5	7	4	7
theta5	3038	250	5	4	21	12
theta6	4375	300	4	6	4	6
equalG11	801	801	5	8	5	6
maxG11	800	800	6	11	4	8
maxG32	2000	2000	6	12	4	10
maxG51	1000	1000	5	8	4	7
qpG11	800	1600	5	7	4	6
thetaG11	2401	801	5	6		5

			time	time/iter	time	time/iter	cg/iter
	n	m	pensdp		p_QMR(AINV)		
buck2	144	237	5	0.06	4	0.06	7
buck3	544	1185	247	1.24	239	1.28	33
buck4	1200	2545	757	5.82	815	5.91	54
buck5	3280	6801	20291	77.15	16273	65.62	80
trto2	144	235	3	0.03	3	0.04	7
trto3	544	865	50	0.53	57	0.51	20
trto4	1200	1873	475	3.30	517	3.08	30
trto5	3280	5040	11939	63.84	3168	23.64	20
vibra2	144	237	6	0.06	5	0.06	8
vibra3	544	1185	223	1.04	149	1.04	17
vibra4	1200	2545	632	5.50	914	6.30	17
vibra5	3280	6801	10848	63.44	18439	54.88	77
shmup2	200	1281	211	2.48	199	2.80	5
shmup3#	420	1641	465	4.51	368	4.60	6
shmup4#	800	4961	3001	27.28	2455	27.58	7
spmup5#	1800	11141	4745	139.56	4564	138.30	3

Test results: large sparse NLP

First experiments for convex QP

CG (much) better when

$$\boldsymbol{H} \text{ sparse}, \quad \boldsymbol{L} \text{ much denser} \quad (\boldsymbol{H} = \boldsymbol{L}^T \boldsymbol{L})$$

CUTE problems `huestis, hues-mod, cvxqp*`

problem	var	constr	pennon		CG(SGS)		
			time(s)	Nwt iter	time(s)	Nwt iter	CG iter
cvxqp1_l	10000	5000	357	38	305	60	74354
cvxqp2_l	10000	2500	61	27	33	34	6694
cvxqp3_l	10000	7500	310	25	66	44	16440
huestis	10000	2	35	743	1	27	74
hues-mod	10000	2	35	605	2	25	68
qship04s	1291	241	2	43	48	58	22821
cont-200	40397	39601	48	34	2009	43	107721
lukvle3	50002	2	7	17	13	45	54
lukvli16_l	249997	187479	73	39	71	40	60

Test results: large sparse NLP

First experiments for convex QP

CG (much) better when

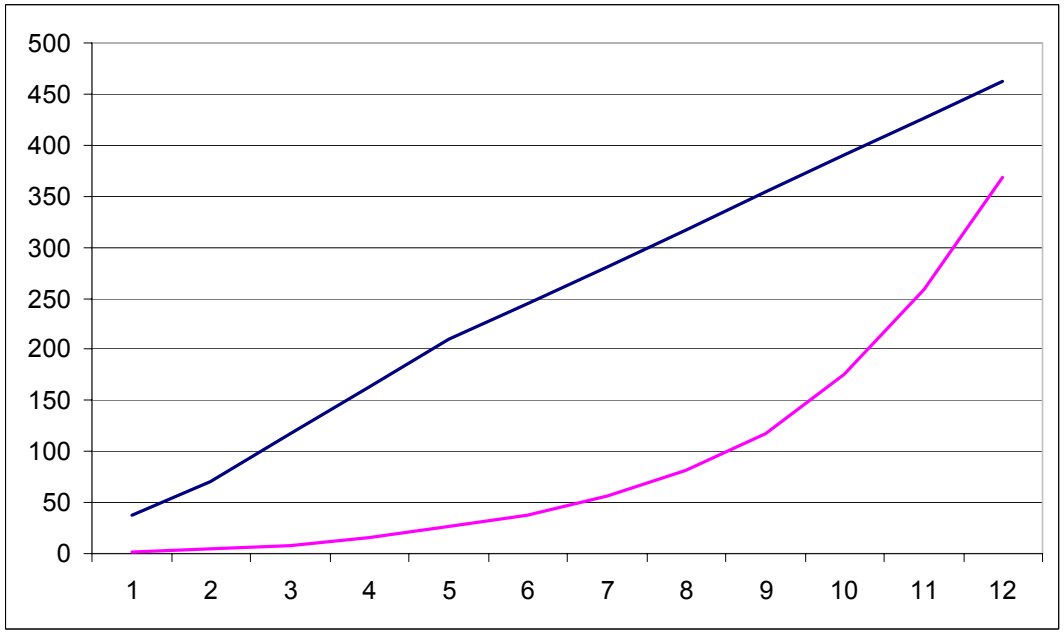
$$\boldsymbol{H} \text{ sparse}, \quad \boldsymbol{L} \text{ much denser} \quad (\boldsymbol{H} = \boldsymbol{L}^T \boldsymbol{L})$$

CUTE problems `huestis, hues-mod, cvxqp*`

Other problems (when sparse Cholesky routine is efficient):

good start, then slower and slower

Typical example: `cvxqp1-l`

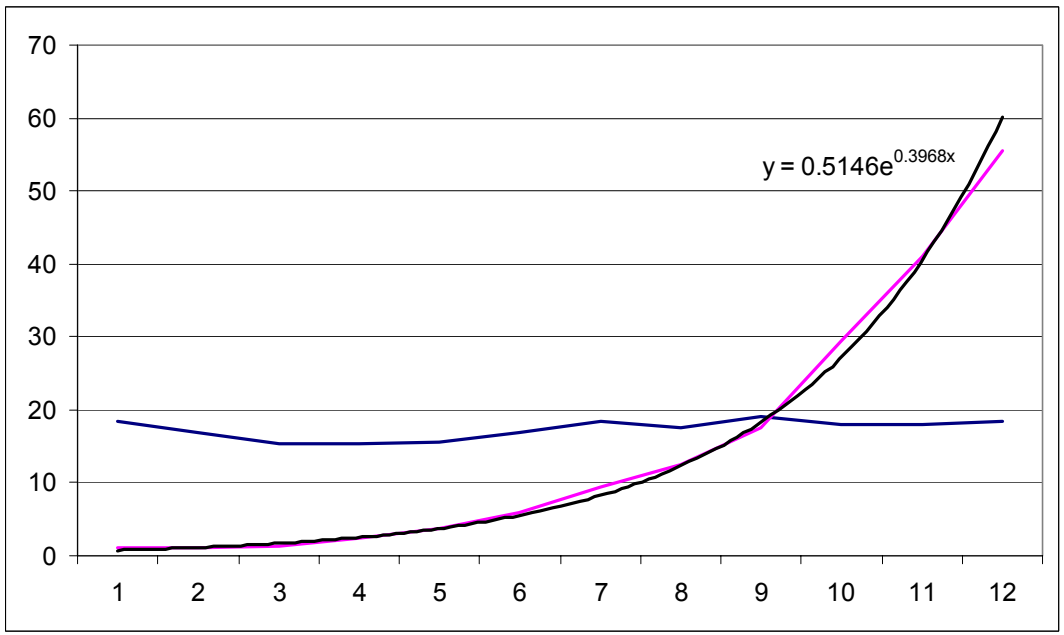


cvxqp1_l

cumullative total time

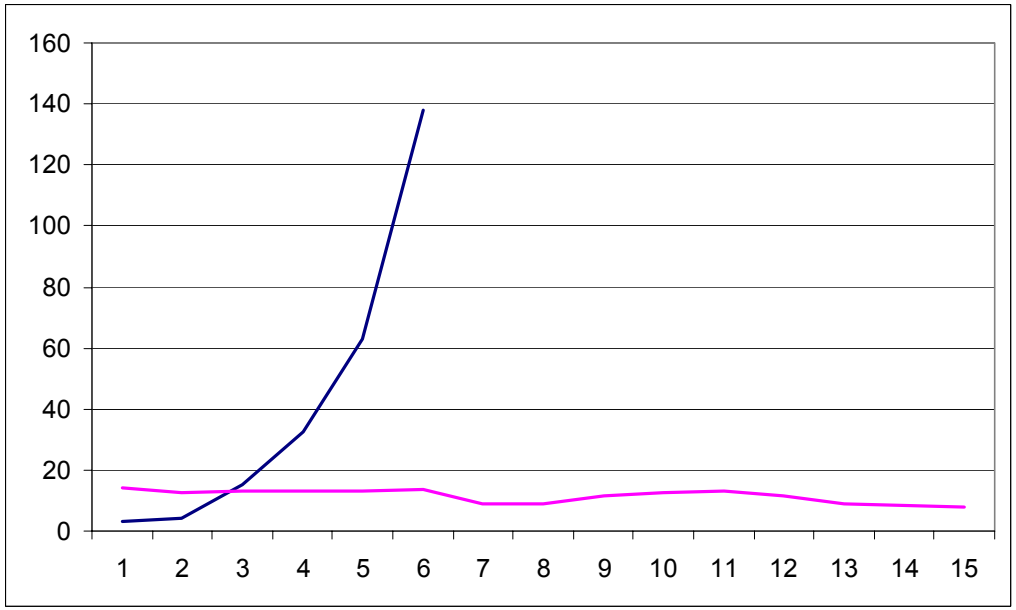
Cholesky

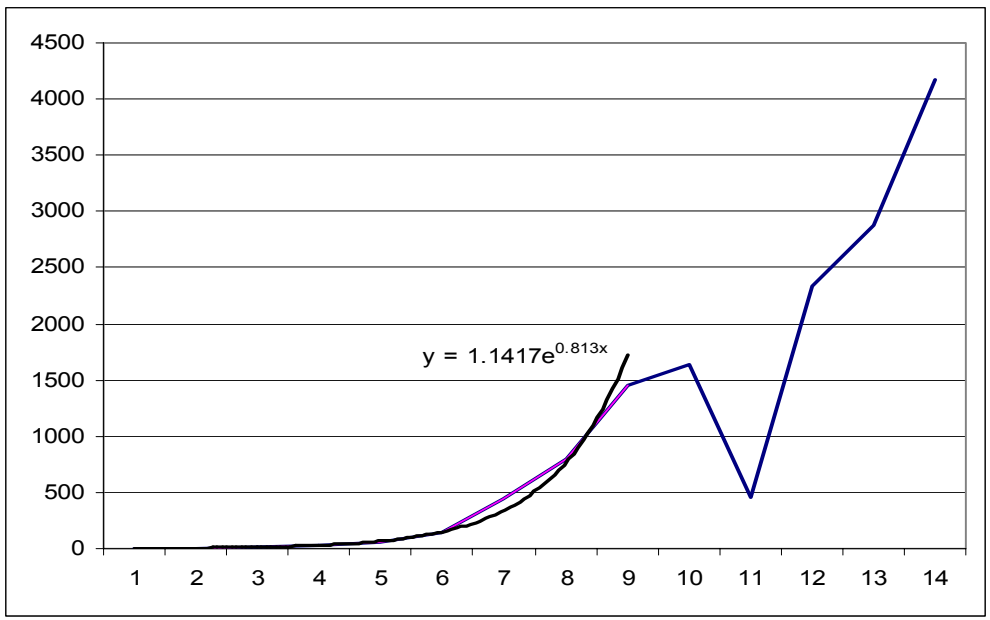
CG



time per one Nwt. step

1.00
1.00
1.33
2.33
3.67
6.00
9.50
12.50
17.50
29.50
41.00
55.50





twod

time per one Nwt. step

Cholesky

CG

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)

First preliminary results (dense SDP):

- number of Newton iterations about the same
- number of CG iterations 1–2 times higher

compared to exact Hessian

	A_PCG(BFGS) time	A_PCG(BFGS) Nwt. iter	A_PCG(BFGS) CG iter	PCG(BFGS) time	PCG(BFGS) Nwt. iter	PCG(BFGS) CG iter
control3	8	57	5185	3	55	3010
control4	62	74	18326	19	83	12071
control5	129	63	19797	45	65	12176
mcp250-1	8	36	143	3	36	143
mcp500-1	44	43	195	16	38	150
theta3	16	52	273	20	57	223
theta4	40	59	338	64	62	289
theta5	88	55	352	166	69	321
theta6	159	58	401	294	53	206
equalG11	1452	41	198	756	41	198
maxG11	163	37	216	51	37	212
qpG11	409	41	188	183	41	186
thetaG11	1006	112	1492	446	104	477

Problems with large n and small m

Linear SDP, dense Hessian:

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

- $O(n^3)$

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

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

Number of Newton steps (yellow) and QMR iterations (white)

problem	pensdp	pen_QMR		pen_approx-QMR	
ham_7_5_6	54	47	109	45	78
ham_9_8	54	57	132	61	91
ham_8_3_4		51	116	50	89
ham_9_5_6				59	108
theta32#	50	48	458	50	453
theta42#	53	52	435	53	718
theta6#	71	61	574	60	362
theta62#				52	404
theta8	61	62	744	62	504
theta82				57	482
theta83				58	647
theta10		68	748	62	473
theta102				58	744
theta103				56	769
theta104				56	834
theta12		63	606	66	518
keller4	47	54	376	52	864
sanr200-0.7	53	55	531	56	698

Total CPU time (white) and time per onew Newton step (yellow)

problem	n	m	pensdp		pen_QMR		pen_approx-QMR	
ham_7_5_6	1793	128	176	3.26	47	1.00	4	0.09
ham_9_8	2305	512	497	9.20	244	4.28	197	3.23
ham_8_3_4	16129	256			6944	136.16	90	1.80
ham_9_5_6	53761	512					1499	25.41
theta32#	2286	150	200	4.00	71	1.48	11	0.22
theta42#	5986	200	2998	56.57	827	15.90	49	0.92
theta6#	4375	300	1714	24.14	490	8.03	60	1.00
theta62#	13390	300					118	2.27
theta8	7905	400	15139	248.18	1975	31.85	350	5.65
theta82	23872	400					971	17.04
theta83	39862	400					3274	56.45
theta10	12470	500		961.28	5842	85.91	703	11.34
theta102	37467	500					3635	62.67
theta103	62516	500					9850	175.89
theta104	87845	500					20329	363.02
theta12	17979	600			14098	223.78	1365	20.68
keller4	5101	171	3236	68.85	587	10.87	86	1.65
sanr200-0.7	6033	200	5790	109.25	916	16.65	103	1.84

Number of QMR iterations per one Newton step

problem	n	m	pen_QMR	pen_appr-QMR
ham_7_5_6	1793	128	2	2
ham_9_8	2305	512	2	1
ham_8_3_4	16129	256	2	2
ham_9_5_6	53761	512		2
theta32	2286	150	10	9
theta42	5986	200	8	14
theta6	4375	300	9	6
theta62	13390	300		8
theta8	7905	400	12	8
theta82	23872	400		8
theta83	39862	400		11
theta10	12470	500	11	8
theta102	37467	500		13
theta103	62516	500		14
theta104	87845	500		15
theta12	17979	600	10	8
keller4	5101	171	7	17

Accuracy versus CPU time

	prec	slackness	pensdp	p_QMR(3)		p_app-QMR(2)		Objective
			time	time	CG	time	CG	
theta6	1.00E-04	5.70E-04	1714	467	574	57	362	63.47655
	5.00E-06	8.70E-06		488	602	84	593	63.47719
	1.00E-07	5.60E-06		683	1660	233	1909	63.47709
keller4	1.00E-04	6.90E-05	1687	552	400	34	721	14.01218
	5.00E-06	1.10E-05		638	753	36	755	14.01225
	1.00E-07	4.60E-07		763	1330	117	2732	14.01224

Nonlinear SDP—FMO with stability constraints

Can CG + approx. Hessian help?

Partly. . .

No preconditioning, approx. Hessian:
as many gradient evaluations as CG steps (good)
CG with no preconditioning inefficient (bad)

Nonlinear SDP—FMO with stability constraints

Can CG + approx. Hessian help?

Partly. . .

No preconditioning, approx. Hessian:
as many gradient evaluations as CG steps (good)
CG with no preconditioning inefficient (bad)

Evaluation of exact diagonal as expensive as evaluation of full Hessian
Evaluation of approx. diagonal . . . . .

Only L-BFGS preconditioner can be used — but it isn't really efficient

		pennon		app-CG(BFGS-N)		
	n	time	Nwt	time	Nwt	CG
shape2	200	1699	63	840	62	3192
shape3	420	18949	77	10622	75	8016

Conclusions

Much to be done. . .

Conclusions

Much to be done. . .

Dense SDP problems:

- PENNON-CG equal or faster (for most problems)

Conclusions

Much to be done. . .

Dense SDP problems:

- PENNON-CG equal or faster (for most problems)

Sparse NLP:

- Useful for sparse problems with dense Cholesky factor
- General problems: lack of good preconditioners:
  CG fast in first PENNON iterations, then slower and slower and. . .
  → combination CG - Cholesky ?

Conclusions

Much to be done. . .

Dense SDP problems:

- PENNON-CG equal or faster (for most problems)

Sparse NLP:

- Useful for sparse problems with dense Cholesky factor
- General problems: lack of good preconditioners:
  CG fast in first PENNON iterations, then slower and slower and. . .
  $\longrightarrow$ combination CG - Cholesky ?

Hessian-free SDP:

- First (very) promising results