# A Primal-Dual Augmented Lagrangian Method for the Solution of Convex Semidefinite Programming Problems with Applications in Structural Optimization

Arefeh kavand (ESR 12)

Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)

jointly with Michael Stingl and Michal Kočvara

POEMA Learning Week 2 (Toulouse), September 2021


# History of penalty/barrier multiplier methods

- Modified barrier functions for LP, QP and NLP, Polyak (1992–today)
- Penalty/Barrier-Multiplier (PBM) methods for convex NLP, Ben-Tal and Zibulevsky (1997)
- Exponential Multiplier Method for SDP, Doljansky and Teboulle (1998)
- PBM algorithm for SDP, Mosheyev and Zibulevsky (2000)
- PENNON: Augmented Lagrangian (AL) type solver for convex NLP and (nonlinear) SDP, Kočvara and Stingl (2003,2007,2009)
- Nonlinear Rescaling Method, primal-dual updates in NLP, Griva and Polyak (2006 and later)
- list certainly not comprehensive!

# Goal today

- consider convex SDP (here: focus on linear SDP)
- build on existing PBM/AL concept
- integrate idea of primal-dual updates (new in SDP)

  $\rightarrow$ gain in efficiency and stability
- extend to (certain) large scale SDPs by matrix free approach using (P)CG method
- IP inspired preconditioner for low rank SDPs

# Statement of the problem

# Statement of the problem

## Semidefinite program

$$\min_{y \in \mathbb{R}^n} \quad b^\top y \quad \text{s. t.} \tag{SDP}$$

$$\sum_{i=1}^{n} y_i A_i - C \preceq 0$$

$$Dy - d \leq 0.$$

- $b \in \mathbb{R}^n$
- $C, A_i \in \mathbb{S}^m, i = 1, \ldots, n$
- $D \in \mathbb{R}^{\nu \times n}$, $d \in \mathbb{R}^\nu$

# Statement of the problem

. . . rewrite as generic SDP by

$$\min_{y \in \mathbb{R}^n} b^\top y \quad \text{s. t.} \quad A(y) := \begin{pmatrix} A^{\text{lmi}}(y) & 0 \\ 0 & A^{\text{lin}}(y) \end{pmatrix} \preceq 0$$

$$A^{\text{lmi}}(y) = \underbrace{\sum_{i=1}^{n} A_i y_i}_{=:\mathbf{A}^0(y)} - C, \quad A^{\text{lin}}(y) = \underbrace{\text{diag}(Dy - d)}_{\text{diagonal structure}}$$

- reason for split: flexibility in choice of penalty function
- note: in practice there can be multiple LMI blocks of different size

# Statement of the problem

## Lagrange multiplier

$$X = \begin{pmatrix} X^{\text{lmi}} & 0 \\ 0 & X^{\text{lin}} \end{pmatrix} \in \mathbb{S}_+^{m+\nu};\ X^{\text{lmi}} \text{ LMI multiplier, } X^{\text{lin}} \text{ diagonal matrix}$$

## The Lagrangian for the SDP problem

$$L(y, X) = b^\top y + A(y) \bullet X = b^\top y + A^{\text{lmi}}(y) \bullet X^{\text{lmi}} + A^{\text{lin}}(y) \bullet X^{\text{lin}}$$

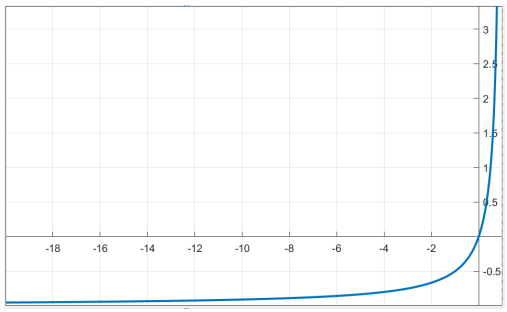
# Penalty-Barrier-Multiplier/Augmented-Lagrangian (PBM/AL) framework

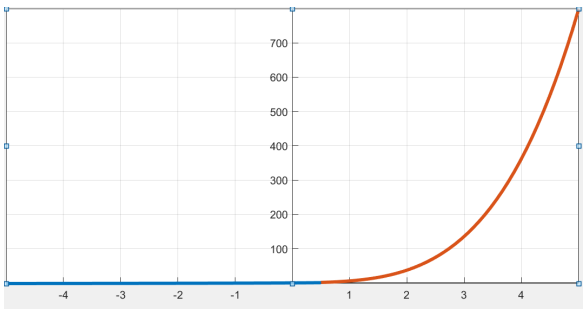
# Penalty/Barrier function

## Penalty/Barrier function

$$\Phi_\pi(A(y)) := \pi \begin{pmatrix} \Phi^{\mathrm{lmi}}(A^{\mathrm{lmi}}(y)/\pi) & 0 \\ 0 & \Phi^{\mathrm{lin}}(A^{\mathrm{lin}}(y)/\pi) \end{pmatrix}$$

- $\pi$ penalty parameter
- $\Phi^{\mathrm{lmi}}(A) = (I - A)^{-1} - I$ hyperbolic matrix penalty function $\rightarrow$ primary matrix function, but no EV decomposition required ($\star$)
- $\Phi^{\mathrm{lin}}$ scalar valued log-barrier function with quadratic extrapolation applied entry-wise to diagonal entries ($\star\star$)




($\star$) Kočvara, Stingl, *Optimization Methods and Software*, (2003)
($\star\star$) Zibulevsky, Ben-Tal, *SIAM Journal on Optimization*, (1997)

# Nonlinear rescaling

## Nonlinear rescaling of constraint

$$A(y) \preceq 0 \Leftrightarrow \Phi_\pi\left(A(y)\right) \preceq 0$$

## Lagrangian for (SDP) with rescaled constraint

$$F(y, X, \pi) = \underbrace{b^\top y}_{\text{objective}} + \underbrace{X \bullet \Phi_\pi\left(A(y)\right)}_{\substack{\text{inner product of} \\ \text{multiplier and scaled constraints}}}$$

# Augmented Lagrangian

## PBM/AL Algorithm: in every (outer) iteration $k$ ...

1. minimize $F(\cdot, X_k, \pi_k)$ with respect to $y$ with fixed $X_k$ and $\pi_k \rightarrow y_{k+1}$
2. update multiplier: $X_{k+1} = (1-\gamma)\bar{X}(y_{k+1}, X_k, \pi_k) + \gamma X_k$
3. update penalty parameter $\rightarrow \pi_{k+1}$

- for given $\Phi_\pi$ multiplier update function $\bar{X}$ uniquely determined by

$$\nabla_y F(y_{k+1}, X_k, \pi_k) = \nabla_y L(y_{k+1}, \bar{X}(y_{k+1}, X_k, \pi_k)) \ (= 0!)$$

- $0 \leq \gamma < 1$: damping parameter

# Unconstrained minimization

Step (1) of PBM/AL Algorithm: solve

$$\nabla_y F(y, X_k, \pi_k) = 0$$

by damped Newton's method

**Newton iteration: in every (inner) iteration $\ell$ ...**

$$\nabla^2_{yy} F(y_k^\ell, X_k, \pi_k)\Delta y = -\nabla_y F(y_k^\ell, X_k, \pi_k);\ y_k^{\ell+1} = y_k^\ell + \alpha_\ell \Delta y$$

works very well in most cases, but ...

- even with direct Cholesky solvers sometimes problems with high precision (see example next slide)
- assembly and factorization of $\nabla^2_{yy} F$ can be rather expensive

# test case gpp500-1 (SDPLIB, B. Borchers)

```
                                                  .
                                                  .
                                                  .
O  76 267: f=-2.532054e+01, d=-2.532054e+01, alp=   1, pfeas= 3.08e-09, dfeas= 6.37e-07, dgap= 2.26e-09
I  76 268: f=-2.532054e+01, d=-2.532054e+01, alp=   1, pfeas= 3.08e-09, dfeas= 2.93e-06, dgap= 8.59e-08
I  76 269: f=-2.532054e+01, d=-2.532055e+01, alp=   1, pfeas= 3.08e-09, dfeas= 3.29e-06, dgap= 7.59e-08
I  76 270: f=-2.532054e+01, d=-2.532054e+01, alp=   1, pfeas= 3.08e-09, dfeas= 2.35e-06, dgap= 7.41e-08
I  76 271: f=-2.532054e+01, d=-2.532055e+01, alp=   1, pfeas= 3.08e-09, dfeas= 1.55e-06, dgap= 3.25e-08
I  76 272: f=-2.532054e+01, d=-2.532054e+01, alp=   1, pfeas= 3.08e-09, dfeas= 1.21e-06, dgap= 1.81e-08
I  76 273: f=-2.532054e+01, d=-2.532054e+01, alp=   1, pfeas= 3.08e-09, dfeas= 3.01e-06, dgap= 8.17e-08
I  76 274: f=-2.532054e+01, d=-2.532055e+01, alp=   1, pfeas= 3.08e-09, dfeas= 3.71e-06, dgap= 7.66e-08
I  76 275: f=-2.532054e+01, d=-2.532054e+01, alp=   1, pfeas= 3.08e-09, dfeas= 2.22e-06, dgap= 5.88e-08
I  76 276: f=-2.532054e+01, d=-2.532054e+01, alp=   1, pfeas= 3.08e-09, dfeas= 1.46e-06, dgap= 3.55e-08
                                                  .
                                                  .
                                                  .
```

- dual feasibility (`dfeas`) determined by $\|\nabla_y F(y_k^\ell, X_k, \pi_k)\|$
- problem: $\|\nabla_y F(y_k^\ell, X_k, \pi_k)\| \approx 10^{-6}$ while $\langle \nabla_y F(y_k^\ell, X_k, \pi_k), \Delta y \rangle \approx 10^{-16}$
- many iterations without improvement in dual feasibility

# Ill conditioning

## Hessian formula

$$\mathbf{H}(y) = 2\mathbf{A}^\top (W(y) \otimes V(y))\mathbf{A}$$

where

- $\mathbf{A} \in \mathbb{R}^{m^2 \times n}$ vectorized LMI data matrix; linear constraints neglected
- $V(y) = \pi \left(\pi I - A(y)\right)^{-1}$

  ... eigenvalues between 0 and 1 if $y$ feasible
- $W(y) = \pi^{-1} \bar{X}(y)$

  ... $\bar{X}$ multiplier update function $\rightarrow$ can be very ill-conditioned!
- ill conditioning carries over to $\mathbf{H}$

# Complexity

**Complexity of a Newton step in PBM/AL method ***

- cost of the solution of the Newton system $\rightarrow \mathcal{O}(n^3)$
- total complexity of the assembly of the Hessian $\rightarrow \mathcal{O}(nm^3)$

**Alternative**: use iterative solver; avoid assembly (matrix-free approach)

## Hessian-vector multiplication

$$\mathbf{H}(y)d = \mathbf{A}^* Q(y,d), \quad Q(y,d) = \bar{X}(y,X,\pi)\mathbf{A}^0(d)(\pi I - A(y))^{-1}$$

- assuming sparse data matrices $\rightarrow \mathcal{O}(n + m^3)$ per iteration

  ... but: number of CG iterations again strongly depending on conditioning!

* Kočvara, Stingl, *Mathematical Programming* (2007)

# Primal-Dual Augmented Lagrangian approach (PDAL)

# Recall ...

## Our motivation

- observation: in ill-conditioned cases, sometimes hard to solve Newton systems in PBM/AL algorithm to high precision
- particularly problematic if an iterative solver is used for the Newton system (which is necessary for large scale SDPs!)

## Note

PDAL is inspired by the work of Griva and Polyak in the NLP setting *

* Griva and Polyak, "Primal-dual nonlinear rescaling method with dynamic scaling parameter update", *Mathematical Programming*, (2006)

# PDAL idea

## ... split the Newton system

$$\nabla_y F(y, X_k, \pi_k) = b + \mathbf{A}^* \bar{X}(y, X_k, \pi_k) = 0 \tag{1}$$

$$\Longleftrightarrow$$

$$b + \mathbf{A}^* \widehat{X} = 0 \tag{2}$$

$$\widehat{X} - \bar{X}(y, X_k, \pi_k) = 0 \tag{3}$$

where $\bar{X}(\cdot)$ is the previous multiplier update function and $\widehat{X}$ a new variable

- linear structure of (2) $\rightarrow$ always satisfied (at least with direct solver!)
- $\widehat{X} \succ 0$ as soon as nonlinear equation (3) is solved sufficiently well

$\rightarrow \widehat{X}$ feasible for dual SDP after few iterations ... **early stopping!**

# Derivation of Newton iteration

linearize the following equations:

$$G_1(y, \widehat{X}) = b + \mathbf{A}^* \widehat{X} \stackrel{!}{=} 0,$$
$$G_2(y, \widehat{X}) = \widehat{X} - \bar{X}(y, X_k, \pi_k) \stackrel{!}{=} 0.$$

## Directional derivatives of $G_1$ and $G_2$:

$$\partial_y G_1(y, \widehat{X})[\Delta y] = 0,$$
$$\partial_y G_2(y, \widehat{X})[\Delta y] = -\bar{X}(y)\mathbf{A}^0(\Delta y)\mathcal{Z}(y) - \mathcal{Z}(y)\mathbf{A}^0(\Delta y)\bar{X}(y),$$
$$\partial_{\widehat{X}} G_1(y, \widehat{X})[\Delta \widehat{X}] = \mathbf{A}^* \Delta \widehat{X},$$
$$\partial_{\widehat{X}} G_2(y, \widehat{X})[\Delta \widehat{X}] = \Delta \widehat{X}.$$

# Derivation of Newton iteration (cont.)

## Primal-Dual Newton system

$$\mathbf{A}^* \Delta \widehat{X}^\ell = -b - \mathbf{A}^* \widehat{X}^\ell \tag{4}$$

$$\bar{X}(y^\ell)\mathbf{A}^0(\Delta y^\ell)\mathcal{Z}(y^\ell) + \mathcal{Z}(y^\ell)\mathbf{A}^0(\Delta y^\ell)\bar{X}(y^\ell) - \Delta \widehat{X}^\ell = \widehat{X}^\ell - \bar{X}(y^\ell) \tag{5}$$

Solving (5) for $\Delta \widehat{X}^\ell$, inserting into (4) and sorting yields:

$$\mathbf{H}(y^\ell)\Delta y^\ell = -\nabla_y F(y^\ell)$$

$$\Delta \widehat{X}^\ell = \bar{X}(y^\ell)\mathbf{A}^0(\Delta y^\ell)\mathcal{Z}(y^\ell) + \mathcal{Z}(y^\ell)\mathbf{A}^0(\Delta y^\ell)\bar{X}(y^\ell) + \bar{X}(y^\ell) - \widehat{X}^\ell$$

# Derivation of Newton iteration (cont.)

$$\mathbf{H}(y^\ell)\Delta y^\ell = -\nabla_y F(y^\ell) \tag{6}$$

$$\Delta\widehat{X}^\ell = \bar{X}(y^\ell)\mathbf{A}^0(\Delta y^\ell)\mathcal{Z}(y^\ell) + \mathcal{Z}(y^\ell)\mathbf{A}^0(\Delta y^\ell)\bar{X}(y^\ell) + \bar{X}(y^\ell) - \widehat{X}^\ell \tag{7}$$

- neglected explicit dependencies on $X_k, \pi_k$ and outer index $k$ here
- (6): same system for $\Delta y^\ell$ as in primal PBM/AL iteration
- (7): explicit formula for $\Delta\widehat{X}^\ell$ (alternative multiplier update formula!)

In order to damp the Newton step, we introduce the merit function:

**Merit function**

$$M(y, \widehat{X}) = \frac{1}{2}\left(\|G_1(y, \widehat{X})\|^2 + \|G_2(y, \widehat{X})\|^2\right) .$$

# The (hybrid) PDAL algorithm

for $k = 0, 1, \ldots$ do until DIMACS error small enough ...

- set $\widehat{X}_k^0 := X_k$ and $y_k^0 := y_k$
- for $\ell = 0, 1, \ldots$ do
  - Newton update: compute $(\Delta y_k^\ell, \Delta \widehat{X}_k^\ell)$ from (6,7)
  - Backtracking line search: compute a step length $0 < \alpha \leq 1$ such that

    $$M(y_k^\ell + \alpha \Delta y_k^\ell, \widehat{X}_k^\ell + \alpha \Delta \widehat{X}_k^\ell) \leq M(y_k^\ell, \widehat{X}_k^\ell) + 0.05 \alpha M'(y_k^\ell, \widehat{X}_k^\ell)[\Delta y_k^\ell, \Delta \widehat{X}_k^\ell],$$

  - if $\widehat{X}_k^\ell + \alpha \Delta \widehat{X}_k^\ell \succ 0$ (dual feasibility!), check for early stopping
    ... if satisfied, set $X_k^* := \widehat{X}_k^\ell + \alpha \Delta \widehat{X}_k^\ell$, $y_{k+1} := y_k^\ell + \alpha \Delta y_k^\ell$; STOP
  - (optionally): check early stopping with primal multiplier update;
    ... if satisfied, set $X_k^* := \bar{X}(y_k^\ell + \alpha \Delta y_k^\ell)$, $y_{k+1} := y_k^\ell + \alpha \Delta y_k^\ell$; STOP
  - $y_k^{\ell+1} := y_k^\ell + \alpha \Delta y_k^\ell$, $\widehat{X}_k^{\ell+1} := \widehat{X}_k^\ell + \alpha \Delta \widehat{X}_k^\ell$
- update multiplier & penalty: $X_{k+1} = (1 - \gamma) X_k^* + \gamma X_k, \quad \pi_k \to \pi_{k+1}$

# The (hybrid) PDAL algorithm (cont.)

- $M'(y_k^\ell, \widehat{X}_k^\ell)[\Delta y_k^\ell, \Delta \widehat{X}_k^\ell]$ is the directional derivative of the merit function at $(y_k^\ell, \widehat{X}_k^\ell)$ in the direction of $(\Delta y_k^\ell, \Delta \widehat{X}_k^\ell)$.
- early stopping: duality gap improved?
- additionally we can also monitor the primal multiplier update and check if the stopping criteria satisfied.

Low rank SDPs, iterative solvers and preconditioning

# Idea and assumption

**Goal**: use PCG method for solution of Newton system

$$\mathbf{H}(y^\ell)\Delta y^\ell = -\nabla_y F(y^\ell)$$

Recall: cost of a Hessian-vector multiplication $\rightarrow \mathcal{O}(n + m^3)$

**for construction of preconditioner: use low-rank type assumption**

Assume that the optimal multiplier $X^*$ has $k$ outlying eigenvalues, i.e.

$$0 \leq \lambda_1(X^*) \leq \cdots \leq \lambda_{m-k}(X^*) \ll \lambda_{m-k+1}(X^*) \leq \cdots \leq \lambda_m(X^*)$$

# Decomposition of Hessian

## Hessian for multiple SDP blocks + linear constraints

$$\mathbf{H} = 2\sum_{i=1}^{p} \mathbf{A}_i^\top (W_i \otimes V_i)\mathbf{A}_i + \mathbf{H}_{\text{lin}}$$

where

$$W_i := \pi^{-1}\bar{X}_i(y) \qquad \text{scaled multiplier, low-rank}$$

$$V_i := \pi\left(\pi I_i - A_i^{\text{lmi}}(y)\right)^{-1} \qquad 0 < \lambda_j(V_i) \leq 1$$

## low-rank Decomposition of $W_i$, $i = 1, \ldots, p$

$$W_i = W_i^0 + \tilde{W}_i \tilde{W}_i^\top$$

Zhang, Lavaei,"Modified interior-point method for large-and-sparse low-rank semidefinite programs" *IEEE*, (2017)

# Low rank Preconditioner

## Lemma

*$\boldsymbol{H}$ can be written as:*

$$\boldsymbol{H} = \boldsymbol{H}_{\text{lin}} + H_\gamma^0 + H_\gamma^{\text{lr}}, \text{ where}$$

$$H_\gamma^0 = 2\sum_{i=1}^{p} \mathrm{A}_i^\top (W_i^0 \otimes V_i)\mathrm{A}_i,$$

$$H_\gamma^{\text{lr}} = 2\sum_{i=1}^{p} \mathrm{A}_i^\top (\tilde{W}_i \otimes \Delta_i)(\tilde{W}_i \otimes \Delta_i)^\top \mathrm{A}_i$$

*and $\Delta_i$ is chosen such that $V_i = \Delta_i \Delta_i^\top$ for all $i = 1, \ldots, p$.*

# Low rank Preconditioner (cont.)

## $H_\gamma$ preconditioner

$$\mathbf{H}_\gamma = \mathbf{H}_{\text{lin}} + \sum_{i=1}^{p} \tau_i \text{diag}\left(\mathsf{A}_i \mathsf{A}_i^\top\right) + H_\gamma^{\text{lr}},$$

- $\mathbf{H}_{\text{lin}}$ and $H_\gamma^{\text{lr}}$ unchanged.
- Second term is an approximation of $H_\gamma^0$ (small eigenvalues!)
- $\tau_i$ scaling parameter.

Details about properties and practical handling (via SMW formula) for formally analogous preconditioner in IP setting can be found in "Habibi, Kavand, Kocvara, Stingl, "Barrier and penalty methods for low-rank semidefinite programming with application to truss topology design",https://hal.archives-ouvertes.fr/hal-03229944"

# Numerical results

# Testing strategy

**Test sets**

- selected SDPLIB examples (by B. Borchers)
  - solution with direct solver
  - DIMACS error 1.0e-7
- selected large scale examples from collection by K. Toh
  - iterative solver (CG) required
  - no strong preconditioner available $\rightarrow$ DIMACS error 1.0e-5
- medium to large problems from truss topology design (by M. Kočvara)
  - $H_\gamma$ low rank preconditioner applicable
  - test DIMACS error 1.0e-5 to 1.0e-7

# Testing strategy

**Purpose and mode of testing**

- compare performance of
  - 'traditional' PBM/AL (= purely primal update strategy)
  - PDAL without primal updates (= purely dual update strategy)
  - PDAL with optional primal updates (= hybrid update strategy)
- report # Newton iterations and # CG iterations rather than computation time
- use different precisions (DIMACS error: sums up primal feasibility, dual feasibility, duality gap, complementarity slackness errors)

# SDPLIB collection by B. Borchers

| problem | n | m | problem | n | m |
|---|---|---|---|---|---|
| equalG11 | 801 | 801 | mcp124-4 | 124 | 124 |
| equalG51 | 1001 | 1001 | mcp250-1 | 250 | 250 |
| gpp100 | 101 | 100 | mcp250-2 | 250 | 250 |
| gpp124-1 | 125 | 124 | mcp250-3 | 250 | 250 |
| gpp124-2 | 125 | 124 | mcp250-4 | 250 | 250 |
| gpp124-3 | 125 | 124 | mcp500-1 | 500 | 500 |
| gpp124-4 | 125 | 124 | mcp500-2 | 500 | 500 |
| gpp250-1 | 250 | 250 | mcp500-3 | 500 | 500 |
| gpp250-2 | 250 | 250 | mcp500-4 | 500 | 500 |
| gpp250-3 | 250 | 250 | qpG11 | 800 | 1600 |
| gpp250-4 | 250 | 250 | qpG51 | 1000 | 2000 |
| maxG11 | 800 | 800 | ss30 | 132 | 426 |
| maxG32 | 2000 | 2000 | theta1 | 104 | 50 |
| maxG51 | 1000 | 1000 | theta2 | 498 | 100 |
| maxG55 | 5000 | 5000 | theta3 | 1106 | 150 |
| maxG60 | 7000 | 7000 | theta4 | 1949 | 200 |
| mcp100 | 101 | 100 | theta5 | 3028 | 250 |
| mcp124-1 | 124 | 124 | theta6 | 4375 | 300 |
| mcp124-2 | 124 | 124 | thetaG11 | 2401 | 801 |
| mcp124-3 | 124 | 124 | thetaG51 | 6910 | 1001 |

# SDPLIB collection by B. Borchers: results

- on average PDAL clearly outperforms purely primal strategy
- hybrid update strategy can further speed up performance
- selected examples . . .

| instance | prim | dual | hybr | primal | dual | hybrid |
|---|---|---|---|---|---|---|
| gpp500-3 | 400 | 46 | 37 | 1.8e-06 | 1.7e-08 | 2.0e-08 |
| gpp500-4 | 400 | 44 | 30 | 2.1e-06 | 3.4e-08 | 5.8e-09 |
| maxG32 | 71 | 83 | 71 | 3.5e-09 | 5.1e-09 | 1.2e-09 |
| maxG55 | 84 | 64 | 55 | 1.4e-08 | 1.8e-08 | 1.6e-08 |
| mcp500-3 | 44 | 45 | 39 | 4.6e-09 | 1.1e-08 | 1.2e-08 |
| mcp500-4 | 56 | 45 | 36 | 3.0e-09 | 1.3e-08 | 1.7e-08 |
| ss30 | 80 | 51 | 47 | 1.2e-08 | 1.4e-08 | 1.4e-08 |
| theta5 | 56 | 50 | 48 | 1.3e-08 | 1.3e-08 | 1.3e-08 |
| theta6 | 54 | 28 | 28 | 1.9e-08 | 1.3e-08 | 1.3e-08 |

note: average DIMACS precision reported (rather than sum)

# collection by K. Toh: selected results

| instance | prim (it & CG) | | dual (it & CG) | | hybr (it & CG) | |
|---|---|---|---|---|---|---|
| hamming-7-5-6 | 48 | 74 | 26 | 44 | 26 | 44 |
| hamming-8-3-4 | 47 | 67 | 25 | 41 | 25 | 41 |
| hamming-9-5-6 | 44 | 65 | 23 | 38 | 22 | 36 |
| theta10 | 64 | 1568 | 67 | 1686 | 66 | 1512 |
| theta102 | 71 | 2249 | 74 | 3984 | 56 | 1985 |
| theta103 | 59 | 1818 | 78 | 4038 | 57 | 1857 |
| theta104 | 75 | 3569 | 76 | 3559 | 54 | 2707 |
| theta12 | 71 | 1382 | 59 | 2849 | 46 | 1820 |
| theta123 | 66 | 2329 | 91 | 3213 | 55 | 1627 |
| theta162 | 74 | 2213 | 70 | 3245 | 51 | 2269 |
| theta42 | 57 | 1253 | 51 | 2282 | 49 | 2293 |
| theta6 | 64 | 1700 | 41 | 1544 | 41 | 1544 |
| theta62 | 64 | 1671 | 39 | 1789 | 38 | 2053 |
| theta8 | 66 | 1683 | 47 | 1546 | 49 | 1878 |
| theta83 | 59 | 1994 | 54 | 1961 | 61 | 2618 |

- CG, no preconditioner, DIMACS prec = 1.0e-5 (up to 1.0e-6 possible)
- hybrid is best in terms of Newton steps, but primal needs less CG iters

# Truss collection by M. Kočvara

Table: problems tru* and vib*

| problem | n | m | lin. constr. |
|---|---|---|---|
| tru15 | 25200 | 421 | 50400 |
| tru17 | 41616 | 545 | 83232 |
| tru19 | 64980 | 685 | 129960 |
| tru21 | 97020 | 841 | 194040 |
| tru23 | 139656 | 1013 | 279312 |
| tru25 | 195000 | 1201 | 390000 |

| problem | n | m | lin. constr. |
|---|---|---|---|
| vib15 | 25200 | (421, 420) | 50400 |
| vib17 | 41616 | (545, 544) | 83232 |
| vib19 | 64980 | (685, 684) | 129960 |
| vib21 | 97020 | (841, 840) | 194040 |
| vib23 | 139656 | (1013, 1012) | 279312 |
| vib25 | 195000 | (1201, 1200) | 390000 |

- beyond scope of SDP software using direct solver
- recall: low rank assumption is satisfied for all these examples

# Truss collection by M. Kočvara: results

tru∗ problems with DIMACS precision 1.0e-5

| | primal Newt | primal CG | dual Newt | dual CG | hybrid Newt | hybrid CG | primal precision | dual precision | hybrid precision |
|---|---|---|---|---|---|---|---|---|---|
| tru15 | 148 | 2135 | 100 | 676 | 91 | 645 | 1.4e-06 | 1.5e-06 | 1.4e-06 |
| tru19 | 158 | 1694 | 109 | 851 | 106 | 810 | 1.6e-06 | 1.6e-06 | 1.5e-06 |
| tru21 | 172 | 1772 | 111 | 970 | 103 | 911 | 1.4e-06 | 7.5e-07 | 1.0e-06 |
| tru23 | 136 | 1580 | 114 | 944 | 148 | 1961 | 1.2e-06 | 1.3e-06 | 1.5e-06 |
| tru25 | 139 | 2729 | 139 | 1199 | 148 | 1586 | 6.2e-07 | 7.1e-07 | 1.4e-06 |

tru∗ problems with DIMACS precision 1.0e-7

| | primal Newt | primal CG | dual Newt | dual CG | hybrid Newt | hybrid CG | primal precision | dual precision | hybrid precision |
|---|---|---|---|---|---|---|---|---|---|
| tru15 | 186 | 2454 | 120 | 831 | 118 | 827 | 2.0e-09 | 7.2e-09 | 1.7e-08 |
| tru19 | 148 | 1861 | 131 | 1048 | 139 | 1096 | 3.2e-09 | 8.2e-09 | 1.4e-08 |
| tru21 | 160 | 1873 | 138 | 1191 | 149 | 1196 | 7.9e-09 | 1.2e-08 | 7.7e-09 |
| tru23 | 177 | 2089 | 165 | 1363 | 199 | 2853 | 1.5e-08 | 6.0e-09 | 3.8e-09 |
| tru25 | 173 | 3046 | 159 | 1429 | 174 | 1930 | 4.8e-09 | 1.6e-08 | 1.7e-08 |

best CPU time for largest cases: $\approx$ 13 m (low prec) vs. $\approx$ 16 m (high prec)

# Truss collection by M. Kočvara cont.

vib* problems with DIMACS precision 1.0e-5

| | primal Newt | primal CG | dual Newt | dual CG | hybrid Newt | hybrid CG | primal precision | dual precision | hybrid precision |
|---|---|---|---|---|---|---|---|---|---|
| vib15 | 239 | 1163 | 121 | 428 | 112 | 451 | 1.4e-06 | 1.7e-06 | 9.7e-07 |
| vib19 | FAIL | FAIL | 148 | 504 | 140 | 420 | n.a. | 1.1e-06 | 1.7e-06 |
| vib21 | FAIL | FAIL | 164 | 563 | 192 | 638 | n.a. | 1.5e-06 | 1.5e-06 |
| vib23 | FAIL | FAIL | 188 | 661 | 210 | 673 | n.a. | 1.5e-06 | 1.4e-06 |
| vib25 | FAIL | FAIL | 206 | 650 | 279 | 1053 | n.a. | 1.5e-06 | 1.5e-06 |

vib$*$ problems with DIMACS precision 1.0e-7

| | primal Newt | primal CG | dual Newt | dual CG | hybrid Newt | hybrid CG | primal precision | dual precision | hybrid precision |
|---|---|---|---|---|---|---|---|---|---|
| vib15 | n.a. | n.a. | 147 | 858 | 140 | 1068 | n.a. | 1.4e-08 | 1.5e-08 |
| vib19 | n.a. | n.a. | 202 | 2191 | 190 | 2797 | n.a. | 1.2e-08 | 1.6e-08 |
| vib21 | n.a. | n.a. | 208 | 2325 | 244 | 2483 | n.a. | 1.5e-08 | 1.6e-08 |
| vib23 | n.a. | n.a. | 239 | 3109 | 262 | 3325 | n.a. | 1.5e-08 | 1.5e-08 |
| vib25 | n.a. | n.a. | 281 | 5852 | 365 | 7014 | n.a. | 1.7e-08 | 1.6e-08 |

best CPU time for largest cases: $\approx$ 1 h (low prec) vs. $\approx$ 2 h (high prec)

## Summary

- primal-dual updates improve performance and stability of PBM/AL methods
- application to large scale SDPs possible
- high precision results obtained, if reasonable preconditioner available

## Outlook

- application to low rank examples from polynomial optimization

# Thanks!

# References


- Mosheyev, Leonid, and Michael Zibulevsky. "Penalty/Barrier Multiplier Algorithm for Semidefinite Programming: Dual Bounds and Implementation." Optimization Methods and Software. 1996.
- Griva, Igor, and Roman A. Polyak. "Primal-dual nonlinear rescaling method with dynamic scaling parameter update." Mathematical Programming 106.2 (2006): 237-259.
- Michal Kočvara and Michael Stingl, PENNON: A code for convex nonlinear and semidefinite programming,Optimization Methods and Software, 18.3 (2003): 317-333
- Habibi, Soodeh and Kavand, Arefeh and Kocvara, Michal and Stingl, Michael, "Barrier and penalty methods for low-rank semidefinite programming with application to truss topology design", `https://hal.archives-ouvertes.fr/hal-03229944`