A Robust Algorithm for Semidefinite Programming

Xuan Vinh Doan, vanxuan@uwaterloo.ca
Joint work with S. Kruk and H. Wolkowicz

Department of Combinatorics and Optimization
University of Waterloo

July 19, 2011

Motivation: SDP Methods

Motivation: SDP Methods

- Primal-dual interior-point methods
  - Solving *perturbed* optimality conditions with barrier parameter $\mu \downarrow 0$

Motivation: SDP Methods

- Primal-dual interior-point methods
  - Solving *perturbed* optimality conditions with barrier parameter $\mu \downarrow 0$
- SDP (perturbed) optimality conditions
  - *Overdetermined* system (different from LP)

Motivation: SDP Methods

- Primal-dual interior-point methods
  - Solving *perturbed* optimality conditions with barrier parameter $\mu \downarrow 0$
- SDP (perturbed) optimality conditions
  - *Overdetermined* system (different from LP)
  - Full-rank Jacobian at optimality under nondegeneracy assumptions

Motivation: SDP Methods

- Primal-dual interior-point methods
  - Solving *perturbed* optimality conditions with barrier parameter $\mu \downarrow 0$
- SDP (perturbed) optimality conditions
  - *Overdetermined* system (different from LP)
  - Full-rank Jacobian at optimality under nondegeneracy assumptions
- Current methods
  - Symmetrization to apply Newton's method: HRVW/KSM/M, NT

Motivation: SDP Methods

- Primal-dual interior-point methods
  - Solving *perturbed* optimality conditions with barrier parameter $\mu \downarrow 0$
- SDP (perturbed) optimality conditions
  - *Overdetermined* system (different from LP)
  - Full-rank Jacobian at optimality under nondegeneracy assumptions
- Current methods
  - Symmetrization to apply Newton's method: HRVW/KSM/M, NT
  - Block elimination to reduce size of the system

Motivation: SDP Methods

- Primal-dual interior-point methods
  - Solving *perturbed* optimality conditions with barrier parameter $\mu \downarrow 0$
- SDP (perturbed) optimality conditions
  - *Overdetermined* system (different from LP)
  - Full-rank Jacobian at optimality under nondegeneracy assumptions
- Current methods
  - Symmetrization to apply Newton's method: HRVW/KSM/M, NT
  - Block elimination to reduce size of the system
- Limitations
  - Increasingly ill-conditioned linear systems for search directions with singular Jacobian at optimality

Motivation: SDP Methods

- Primal-dual interior-point methods
  - Solving *perturbed* optimality conditions with barrier parameter $\mu \downarrow 0$
- SDP (perturbed) optimality conditions
  - *Overdetermined* system (different from LP)
  - Full-rank Jacobian at optimality under nondegeneracy assumptions
- Current methods
  - Symmetrization to apply Newton's method: HRVW/KSM/M, NT
  - Block elimination to reduce size of the system
- Limitations
  - Increasingly ill-conditioned linear systems for search directions with singular Jacobian at optimality
  - Difficult to find reasonable preconditioners
  - Difficult to exploit sparsity of data

Motivation: Gauss-Newton Method

Motivation: Gauss-Newton Method

- Newton's method: determined systems (LP optimality conditions)

Motivation: Gauss-Newton Method

- Newton's method: determined systems (LP optimality conditions)

Question

How about overdetermined systems like SDP optimality conditions?

Motivation: Gauss-Newton Method

- Newton's method: determined systems (LP optimality conditions)

Question

How about overdetermined systems like SDP optimality conditions?

- Gauss-Newton's method: least-squares solutions

Motivation: Gauss-Newton Method

- Newton's method: determined systems (LP optimality conditions)

Question

How about overdetermined systems like SDP optimality conditions?

- Gauss-Newton's method: least-squares solutions
- Properties:
  - Maintain well-conditioning of the system $\rightarrow$ solutions with high accuracy

Motivation: Gauss-Newton Method

- Newton's method: determined systems (LP optimality conditions)

Question

How about overdetermined systems like SDP optimality conditions?

- Gauss-Newton's method: least-squares solutions
- Properties:
  - Maintain well-conditioning of the system $\rightarrow$ solutions with high accuracy
  - Preconditioning techniques

Motivation: Gauss-Newton Method

- Newton's method: determined systems (LP optimality conditions)

Question

How about overdetermined systems like SDP optimality conditions?

- Gauss-Newton's method: least-squares solutions
- Properties:
  - Maintain well-conditioning of the system $\rightarrow$ solutions with high accuracy
  - Preconditioning techniques
  - Sparsity exploitation

Outline

Outline

1. Gauss-Newton Approach
   - Bilinear optimality conditions

Outline

1. Gauss-Newton Approach
   - Bilinear optimality conditions
   - Local convergence and crossover

Outline

1. Gauss-Newton Approach
   - Bilinear optimality conditions
   - Local convergence and crossover
   - Presolve and preconditioning

Outline

1. Gauss-Newton Approach
   - Bilinear optimality conditions
   - Local convergence and crossover
   - Presolve and preconditioning
2. Lovász Theta Function Problem

Outline

1. Gauss-Newton Approach
   - Bilinear optimality conditions
   - Local convergence and crossover
   - Presolve and preconditioning
2. Lovász Theta Function Problem
3. Numerics

Outline

1. Gauss-Newton Approach
   - Bilinear optimality conditions
   - Local convergence and crossover
   - Presolve and preconditioning
2. Lovász Theta Function Problem
3. Numerics
4. Summary

Primal-Dual SDP Pair

Primal-Dual SDP Pair

$$
\begin{array}{lrll}
 & p^* := \min & \mathbf{C} \cdot \mathbf{X} & \\
\text{(PSDP)} & \text{s.t.} & \mathcal{A}(\mathbf{X}) = \mathbf{b}, & (\mathbf{A}_i \cdot \mathbf{X} = b_i) \\
 & & \mathbf{X} \succeq 0, &
\end{array}
$$

Primal-Dual SDP Pair

$$
\begin{aligned}
& & p^* := \min \quad & \mathbf{C} \cdot \mathbf{X} \\
& \text{(PSDP)} & \text{s.t.} \quad & \mathcal{A}(\mathbf{X}) = \mathbf{b}, \quad (\mathbf{A}_i \cdot \mathbf{X} = b_i) \\
& & & \mathbf{X} \succeq 0, \\
\\
& & d^* := \max \quad & \mathbf{b}^T \mathbf{y} \\
& \text{(DSDP)} & \text{s.t.} \quad & \mathcal{A}^*(\mathbf{y}) + \mathbf{Z} = \mathbf{C}, \\
& & & \mathbf{Z} \succeq 0, \qquad (\mathbf{Z} = \mathbf{C} - \textstyle\sum_i y_i \mathbf{A}_i)
\end{aligned}
$$

Primal-Dual SDP Pair

$$
\begin{array}{lrll}
 & p^* := \min & \mathbf{C} \cdot \mathbf{X} & \\
\text{(PSDP)} & \text{s.t.} & \mathcal{A}(\mathbf{X}) = \mathbf{b}, & (\mathbf{A}_i \cdot \mathbf{X} = b_i) \\
 & & \mathbf{X} \succeq 0, &
\end{array}
$$

$$
\begin{array}{lrll}
 & d^* := \max & \mathbf{b}^T \mathbf{y} & \\
\text{(DSDP)} & \text{s.t.} & \mathcal{A}^*(\mathbf{y}) + \mathbf{Z} = \mathbf{C}, & \\
 & & \mathbf{Z} \succeq 0, & (\mathbf{Z} = \mathbf{C} - \sum_i y_i \mathbf{A}_i)
\end{array}
$$

- $\mathbf{C}, \mathbf{X}, \mathbf{Z} \in \mathcal{S}^n$: $n \times n$ real symmetric matrices

Primal-Dual SDP Pair

$$
\begin{array}{lrll}
 & p^* := \min & \mathbf{C} \cdot \mathbf{X} & \\
\text{(PSDP)} & \text{s.t.} & \mathcal{A}(\mathbf{X}) = \mathbf{b}, & (\mathbf{A}_i \cdot \mathbf{X} = b_i) \\
 & & \mathbf{X} \succeq 0, &
\end{array}
$$

$$
\begin{array}{lrll}
 & d^* := \max & \mathbf{b}^T \mathbf{y} & \\
\text{(DSDP)} & \text{s.t.} & \mathcal{A}^*(\mathbf{y}) + \mathbf{Z} = \mathbf{C}, & \\
 & & \mathbf{Z} \succeq 0, & (\mathbf{Z} = \mathbf{C} - \sum_i y_i \mathbf{A}_i)
\end{array}
$$

- $\mathbf{C}, \mathbf{X}, \mathbf{Z} \in \mathcal{S}^n$: $n \times n$ real symmetric matrices
- $\mathbf{C} \cdot \mathbf{X} = \text{trace}(\mathbf{CX})$: trace inner product

Primal-Dual SDP Pair

$$
\begin{array}{lrll}
 & p^* := \min & \mathbf{C} \cdot \mathbf{X} & \\
\text{(PSDP)} & \text{s.t.} & \mathcal{A}(\mathbf{X}) = \mathbf{b}, & (\mathbf{A}_i \cdot \mathbf{X} = b_i) \\
 & & \mathbf{X} \succeq 0, &
\end{array}
$$

$$
\begin{array}{lrll}
 & d^* := \max & \mathbf{b}^T \mathbf{y} & \\
\text{(DSDP)} & \text{s.t.} & \mathcal{A}^*(\mathbf{y}) + \mathbf{Z} = \mathbf{C}, & \\
 & & \mathbf{Z} \succeq 0, & (\mathbf{Z} = \mathbf{C} - \sum_i y_i \mathbf{A}_i)
\end{array}
$$

- $\mathbf{C}, \mathbf{X}, \mathbf{Z} \in \mathcal{S}^n$: $n \times n$ real symmetric matrices
- $\mathbf{C} \cdot \mathbf{X} = \text{trace}(\mathbf{CX})$: trace inner product
- $\mathcal{A} : \mathcal{S}^n \to \mathbb{R}^m$: linear transformation; $\mathcal{A}^*$: adjoint

Optimality Conditions

Optimality Conditions

- (Perturbed) optimality conditions
  - Dual feasibility

Optimality Conditions

- (Perturbed) optimality conditions
  - Dual feasibility
  - Primal feasibility

Optimality Conditions

- (Perturbed) optimality conditions
  - Dual feasibility
  - Primal feasibility
  - (Perturbed) complementary slackness condition

Optimality Conditions

- (Perturbed) optimality conditions
  - Dual feasibility
  - Primal feasibility
  - (Perturbed) complementary slackness condition

$$F_\mu(\mathbf{X}, \mathbf{y}, \mathbf{Z}) := \begin{pmatrix} \mathcal{A}^*(\mathbf{y}) + \mathbf{Z} - \mathbf{C} \\ \mathcal{A}(\mathbf{X}) - \mathbf{b} \\ \mathbf{ZX} - \mu\mathbf{I} \end{pmatrix} = \mathbf{0}, \quad \mu > 0.$$

Optimality Conditions

- (Perturbed) optimality conditions
  - Dual feasibility
  - Primal feasibility
  - (Perturbed) complementary slackness condition

$$F_\mu(\mathbf{X}, \mathbf{y}, \mathbf{Z}) := \begin{pmatrix} \mathcal{A}^*(\mathbf{y}) + \mathbf{Z} - \mathbf{C} \\ \mathcal{A}(\mathbf{X}) - \mathbf{b} \\ \mathbf{ZX} - \mu\mathbf{I} \end{pmatrix} = \mathbf{0}, \quad \mu > 0.$$

- An overdetermined system

Optimality Conditions

- (Perturbed) optimality conditions
  - Dual feasibility
  - Primal feasibility
  - (Perturbed) complementary slackness condition

$$F_\mu(\mathbf{X}, \mathbf{y}, \mathbf{Z}) := \begin{pmatrix} \mathcal{A}^*(\mathbf{y}) + \mathbf{Z} - \mathbf{C} \\ \mathcal{A}(\mathbf{X}) - \mathbf{b} \\ \mathbf{ZX} - \mu\mathbf{I} \end{pmatrix} = \mathbf{0}, \quad \mu > 0.$$

- An overdetermined system
- Characterization of optimality

Under constraint qualifications, $(\mathbf{X}, \mathbf{y}, \mathbf{Z})$ *with* $\mathbf{X}, \mathbf{Z} \succeq 0$ *is optimal if and only if* $F_0(\mathbf{X}, \mathbf{y}, \mathbf{Z}) = 0$.

Null Space Representation

- Primal feasibility

Null Space Representation

- Primal feasibility
  - $\mathbf{A} \in \mathbb{R}^{m \times t(n)}$ with $\text{svec}(\mathbf{A}_i)$ as row i, $i = 1, \ldots, m$

Null Space Representation

- Primal feasibility
  - $\mathbf{A} \in \mathbb{R}^{m \times t(n)}$ with $\text{svec}(\mathbf{A}_i)$ as row i, $i = 1, \ldots, m$
  - $\mathbf{Q} \in \mathbb{R}^{t(n) \times (t(n)-m)}$: orthonormal basis of $\mathcal{N}(\mathbf{A})$

Null Space Representation

- Primal feasibility
  - $\mathbf{A} \in \mathbb{R}^{m \times t(n)}$ with $\text{svec}(\mathbf{A}_i)$ as row i, $i = 1, \ldots, m$
  - $\mathbf{Q} \in \mathbb{R}^{t(n) \times (t(n)-m)}$: orthonormal basis of $\mathcal{N}(\mathbf{A})$
  - $\hat{\mathbf{X}} \succ 0$: primal feasible solution, $\hat{\mathbf{x}} = \text{svec}(\hat{\mathbf{X}})$

Null Space Representation

- Primal feasibility
  - $\mathbf{A} \in \mathbb{R}^{m \times t(n)}$ with $\text{svec}(\mathbf{A}_i)$ as row i, $i = 1, \ldots, m$
  - $\mathbf{Q} \in \mathbb{R}^{t(n) \times (t(n)-m)}$: orthonormal basis of $\mathcal{N}(\mathbf{A})$
  - $\hat{\mathbf{X}} \succ 0$: primal feasible solution, $\hat{\mathbf{x}} = \text{svec}(\hat{\mathbf{X}})$

$$\mathcal{A}(\mathbf{X}) = b \Leftrightarrow \exists\, \mathbf{v} \in \mathbb{R}^{t(n)-m} \; : \; \text{svec}(\mathbf{X}) = \hat{\mathbf{x}} + \mathbf{Q}\mathbf{v}$$

Null Space Representation

- Primal feasibility
  - $\mathbf{A} \in \mathbb{R}^{m \times t(n)}$ with $\text{svec}(\mathbf{A}_i)$ as row i, $i = 1, \ldots, m$
  - $\mathbf{Q} \in \mathbb{R}^{t(n) \times (t(n)-m)}$: orthonormal basis of $\mathcal{N}(\mathbf{A})$
  - $\hat{\mathbf{X}} \succ 0$: primal feasible solution, $\hat{\mathbf{x}} = \text{svec}(\hat{\mathbf{X}})$

$$\mathcal{A}(\mathbf{X}) = b \Leftrightarrow \exists\, \mathbf{v} \in \mathbb{R}^{t(n)-m} \; : \; \text{svec}(\mathbf{X}) = \hat{\mathbf{x}} + \mathbf{Q}\mathbf{v}$$

- Dual feasibility
  - $\mathbf{c} = \text{svec}(\mathbf{C})$

Null Space Representation

- Primal feasibility
  - $\mathbf{A} \in \mathbb{R}^{m \times t(n)}$ with $\text{svec}(\mathbf{A}_i)$ as row i, $i = 1, \ldots, m$
  - $\mathbf{Q} \in \mathbb{R}^{t(n) \times (t(n)-m)}$: orthonormal basis of $\mathcal{N}(\mathbf{A})$
  - $\hat{\mathbf{X}} \succ 0$: primal feasible solution, $\hat{\mathbf{x}} = \text{svec}(\hat{\mathbf{X}})$

$$\mathcal{A}(\mathbf{X}) = b \Leftrightarrow \exists\, \mathbf{v} \in \mathbb{R}^{t(n)-m} : \text{svec}(\mathbf{X}) = \hat{\mathbf{x}} + \mathbf{Q}\mathbf{v}$$

- Dual feasibility
  - $\mathbf{c} = \text{svec}(\mathbf{C})$

$$\mathcal{A}^*(\mathbf{y}) + \mathbf{Z} = \mathbf{C} \Leftrightarrow \text{svec}(\mathbf{Z}) = \mathbf{c} - \mathbf{A}^T\mathbf{y}$$

Null Space Representation

- Primal feasibility
  - $\mathbf{A} \in \mathbb{R}^{m \times t(n)}$ with $\text{svec}(\mathbf{A}_i)$ as row i, $i = 1, \ldots, m$
  - $\mathbf{Q} \in \mathbb{R}^{t(n) \times (t(n)-m)}$: orthonormal basis of $\mathcal{N}(\mathbf{A})$
  - $\hat{\mathbf{X}} \succ 0$: primal feasible solution, $\hat{\mathbf{x}} = \text{svec}(\hat{\mathbf{X}})$

$$\mathcal{A}(\mathbf{X}) = b \Leftrightarrow \exists\, \mathbf{v} \in \mathbb{R}^{t(n)-m} : \text{svec}(\mathbf{X}) = \hat{\mathbf{x}} + \mathbf{Q}\mathbf{v}$$

- Dual feasibility
  - $\mathbf{c} = \text{svec}(\mathbf{C})$

$$\mathcal{A}^*(\mathbf{y}) + \mathbf{Z} = \mathbf{C} \Leftrightarrow \text{svec}(\mathbf{Z}) = \mathbf{c} - \mathbf{A}^T\mathbf{y}$$

- Equivalent optimality conditions

$$G_\mu(\mathbf{v}, \mathbf{y}) := \text{sMat}(\mathbf{c} - \mathbf{A}^T\mathbf{y})\text{sMat}(\hat{\mathbf{x}} + \mathbf{Q}\mathbf{v}) - \mu\mathbf{I} = 0$$

Gauss-Newton Method

$$G_\mu(\mathbf{v}, \mathbf{y}) := \mathrm{sMat}(\mathbf{c} - \mathbf{A}^T\mathbf{y})\mathrm{sMat}(\hat{\mathbf{x}} + \mathbf{Q}\mathbf{v}) - \mu\mathbf{I} = 0$$

Gauss-Newton Method

$$G_\mu(\mathbf{v}, \mathbf{y}) := \text{sMat}(\mathbf{c} - \mathbf{A}^T\mathbf{y})\text{sMat}(\hat{\mathbf{x}} + \mathbf{Q}\mathbf{v}) - \mu\mathbf{I} = 0$$

- Overdetermined system
  - $t(n) = n(n+1)/2$ variables, n^2 equations

Gauss-Newton Method

$$G_\mu(\mathbf{v}, \mathbf{y}) := \text{sMat}(\mathbf{c} - \mathbf{A}^T\mathbf{y})\text{sMat}(\hat{\mathbf{x}} + \mathbf{Q}\mathbf{v}) - \mu\mathbf{I} = 0$$

- Overdetermined system
  - $t(n) = n(n+1)/2$ variables, n^2 equations
- Gauss-Newton's method
  - Apply Newton's method to min $\frac{1}{2} \|G_\mu(\mathbf{v}, \mathbf{y})\|_2^2$

Gauss-Newton Method

$$G_\mu(\mathbf{v}, \mathbf{y}) := \text{sMat}(\mathbf{c} - \mathbf{A}^T\mathbf{y})\text{sMat}(\hat{\mathbf{x}} + \mathbf{Q}\mathbf{v}) - \mu\mathbf{I} = 0$$

- Overdetermined system
  - $t(n) = n(n+1)/2$ variables, n^2 equations
- Gauss-Newton's method
  - Apply Newton's method to min $\frac{1}{2}\|G_\mu(\mathbf{v}, \mathbf{y})\|_2^2$
- Gauss-Newton search direction: least-squares solution of the linear system

$$-G_\mu(\mathbf{v}, \mathbf{y}) = G'_\mu(\mathbf{v}, \mathbf{y}) \begin{pmatrix} \Delta\mathbf{v} \\ \Delta\mathbf{y} \end{pmatrix}$$

Gauss-Newton Method

$$G_\mu(\mathbf{v}, \mathbf{y}) := \text{sMat}(\mathbf{c} - \mathbf{A}^T\mathbf{y})\text{sMat}(\hat{\mathbf{x}} + \mathbf{Q}\mathbf{v}) - \mu\mathbf{I} = 0$$

- Overdetermined system
  - $t(n) = n(n+1)/2$ variables, n^2 equations
- Gauss-Newton's method
  - Apply Newton's method to min $\frac{1}{2}\|G_\mu(\mathbf{v}, \mathbf{y})\|_2^2$
- Gauss-Newton search direction: least-squares solution of the linear system

$$-G_\mu(\mathbf{v}, \mathbf{y}) = G'_\mu(\mathbf{v}, \mathbf{y}) \begin{pmatrix} \Delta\mathbf{v} \\ \Delta\mathbf{y} \end{pmatrix}$$

- Jacobian $J = G'_\mu : \mathbb{R}^{(t(n)-m)\times m} \to \mathcal{M}^n$, adjoint J^*

Gauss-Newton Method

$$G_\mu(\mathbf{v}, \mathbf{y}) := \text{sMat}(\mathbf{c} - \mathbf{A}^T\mathbf{y})\text{sMat}(\hat{\mathbf{x}} + \mathbf{Q}\mathbf{v}) - \mu\mathbf{I} = 0$$

- Overdetermined system
  - $t(n) = n(n+1)/2$ variables, n^2 equations
- Gauss-Newton's method
  - Apply Newton's method to min $\frac{1}{2}\|G_\mu(\mathbf{v}, \mathbf{y})\|_2^2$
- Gauss-Newton search direction: least-squares solution of the linear system

$$-G_\mu(\mathbf{v}, \mathbf{y}) = G'_\mu(\mathbf{v}, \mathbf{y}) \begin{pmatrix} \Delta\mathbf{v} \\ \Delta\mathbf{y} \end{pmatrix}$$

- Jacobian $J = G'_\mu : \mathbb{R}^{(t(n)-m)\times m} \to \mathcal{M}^n$, adjoint J^*
- *Matrix-free* iterative method to solve normal equations

$$J^* \circ J \begin{pmatrix} \Delta\mathbf{v} \\ \Delta\mathbf{y} \end{pmatrix} = -J^* \circ G_\mu(\mathbf{v}, \mathbf{y})$$

Infeasible Start

Question

(i) *How to find a starting point* $(\mathbf{X}_0, \mathbf{y}_0, \mathbf{Z}_0)$ *or equivalently* $(\mathbf{v}_0, \mathbf{y}_0)$*?*

Infeasible Start

Question

(i) *How to find a starting point* $(\mathbf{X}_0, \mathbf{y}_0, \mathbf{Z}_0)$ *or equivalently* $(\mathbf{v}_0, \mathbf{y}_0)$*?*

(ii) *How does the algorithm converge?*

Infeasible Start

Question

(i) *How to find a starting point* $(\mathbf{X}_0, \mathbf{y}_0, \mathbf{Z}_0)$ *or equivalently* $(\mathbf{v}_0, \mathbf{y}_0)$*?*

(ii) *How does the algorithm converge?*

- Difficult to find feasible starting point $\rightarrow$ infeasible start

Infeasible Start

Question

(i) *How to find a starting point* $(\mathbf{X}_0, \mathbf{y}_0, \mathbf{Z}_0)$ *or equivalently* $(\mathbf{v}_0, \mathbf{y}_0)$*?*

(ii) *How does the algorithm converge?*

- Difficult to find feasible starting point $\rightarrow$ infeasible start
  - Introduce additional variables $(\mathbf{x}, \mathbf{z})$ with residuals $\mathbf{r}_d, \mathbf{r}_x$, and $\mathbf{R}_c$

Infeasible Start

Question

(i) *How to find a starting point* $(\mathbf{X}_0, \mathbf{y}_0, \mathbf{Z}_0)$ *or equivalently* $(\mathbf{v}_0, \mathbf{y}_0)$*?*

(ii) *How does the algorithm converge?*

- Difficult to find feasible starting point $\rightarrow$ infeasible start
  - Introduce additional variables $(\mathbf{x}, \mathbf{z})$ with residuals $\mathbf{r}_d, \mathbf{r}_x$, and $\mathbf{R}_c$

$$F_\mu(\mathbf{v}, \mathbf{y}, \mathbf{x}, \mathbf{z}) := \begin{pmatrix} \mathbf{z} - \mathbf{c} + \mathbf{A}^T\mathbf{y} \\ \mathbf{x} - \hat{\mathbf{x}} - \mathbf{Q}\mathbf{v} \\ \mathrm{sMat}(\mathbf{z})\mathrm{sMat}(\mathbf{x}) - \mu\mathbf{I} \end{pmatrix} =: \begin{pmatrix} \mathbf{r}_d \\ \mathbf{r}_x \\ \mathbf{R}_c \end{pmatrix} = \mathbf{0}$$

Infeasible Start

Question

(i) *How to find a starting point* $(\mathbf{X}_0, \mathbf{y}_0, \mathbf{Z}_0)$ *or equivalently* $(\mathbf{v}_0, \mathbf{y}_0)$*?*

(ii) *How does the algorithm converge?*

- Difficult to find feasible starting point $\rightarrow$ infeasible start
  - Introduce additional variables $(\mathbf{x}, \mathbf{z})$ with residuals $\mathbf{r}_d, \mathbf{r}_x$, and $\mathbf{R}_c$

$$F_\mu(\mathbf{v}, \mathbf{y}, \mathbf{x}, \mathbf{z}) := \begin{pmatrix} \mathbf{z} - \mathbf{c} + \mathbf{A}^T\mathbf{y} \\ \mathbf{x} - \hat{\mathbf{x}} - \mathbf{Q}\mathbf{v} \\ \mathrm{sMat}(\mathbf{z})\mathrm{sMat}(\mathbf{x}) - \mu\mathbf{I} \end{pmatrix} =: \begin{pmatrix} \mathbf{r}_d \\ \mathbf{r}_x \\ \mathbf{R}_c \end{pmatrix} = \mathbf{0}$$

  - Maintain Gauss-Newton search direction for $\Delta\mathbf{v}$ and $\Delta\mathbf{y}$

Infeasible Start

Question

(i) *How to find a starting point* $(\mathbf{X}_0, \mathbf{y}_0, \mathbf{Z}_0)$ *or equivalently* $(\mathbf{v}_0, \mathbf{y}_0)$*?*

(ii) *How does the algorithm converge?*

- Difficult to find feasible starting point $\rightarrow$ infeasible start
  - Introduce additional variables $(\mathbf{x}, \mathbf{z})$ with residuals $\mathbf{r}_d, \mathbf{r}_x$, and $\mathbf{R}_c$

$$F_\mu(\mathbf{v}, \mathbf{y}, \mathbf{x}, \mathbf{z}) := \begin{pmatrix} \mathbf{z} - \mathbf{c} + \mathbf{A}^T\mathbf{y} \\ \mathbf{x} - \hat{\mathbf{x}} - \mathbf{Q}\mathbf{v} \\ \mathrm{sMat}(\mathbf{z})\mathrm{sMat}(\mathbf{x}) - \mu\mathbf{I} \end{pmatrix} =: \begin{pmatrix} \mathbf{r}_d \\ \mathbf{r}_x \\ \mathbf{R}_c \end{pmatrix} = \mathbf{0}$$

  - Maintain Gauss-Newton search direction for $\Delta\mathbf{v}$ and $\Delta\mathbf{y}$
  - Additional search directions: $\Delta\mathbf{x} = \mathbf{Q}\Delta\mathbf{v} - \mathbf{r}_x$, $\Delta\mathbf{z} = -\mathbf{A}^T\Delta\mathbf{y} - \mathbf{r}_d$

Infeasible Start

Question

(i) *How to find a starting point* $(\mathbf{X}_0, \mathbf{y}_0, \mathbf{Z}_0)$ *or equivalently* $(\mathbf{v}_0, \mathbf{y}_0)$*?*

(ii) *How does the algorithm converge?*

- Difficult to find feasible starting point $\rightarrow$ infeasible start
  - Introduce additional variables $(\mathbf{x}, \mathbf{z})$ with residuals $\mathbf{r}_d, \mathbf{r}_x$, and $\mathbf{R}_c$

$$F_\mu(\mathbf{v}, \mathbf{y}, \mathbf{x}, \mathbf{z}) := \begin{pmatrix} \mathbf{z} - \mathbf{c} + \mathbf{A}^T\mathbf{y} \\ \mathbf{x} - \hat{\mathbf{x}} - \mathbf{Q}\mathbf{v} \\ \mathrm{sMat}(\mathbf{z})\mathrm{sMat}(\mathbf{x}) - \mu\mathbf{I} \end{pmatrix} =: \begin{pmatrix} \mathbf{r}_d \\ \mathbf{r}_x \\ \mathbf{R}_c \end{pmatrix} = \mathbf{0}$$

  - Maintain Gauss-Newton search direction for $\Delta\mathbf{v}$ and $\Delta\mathbf{y}$
  - Additional search directions: $\Delta\mathbf{x} = \mathbf{Q}\Delta\mathbf{v} - \mathbf{r}_x$, $\Delta\mathbf{z} = -\mathbf{A}^T\Delta\mathbf{y} - \mathbf{r}_d$
- Residual updates with step size α: $\mathbf{r}_d \leftarrow (1-\alpha)\mathbf{r}_d$, $\mathbf{r}_x \leftarrow (1-\alpha)\mathbf{r}_x$

Local Convergence and Crossover

- Local quadratic convergence

Local Convergence and Crossover

- Local quadratic convergence
  - Lower bound on smallest singular value of Jacobian

Local Convergence and Crossover

- Local quadratic convergence
  - Lower bound on smallest singular value of Jacobian
  - Kruk et al. (2001): under strict complementary slackness condition, full-rank Jacobian at optimality

Local Convergence and Crossover

- Local quadratic convergence
  - Lower bound on smallest singular value of Jacobian
  - Kruk et al. (2001): under strict complementary slackness condition, full-rank Jacobian at optimality
- Implication

  Under nondegeneracy, there exists a neighborhood around the optimal solution where free Gauss-Newton method converges quadratically.

Local Convergence and Crossover

- Local quadratic convergence
  - Lower bound on smallest singular value of Jacobian
  - Kruk et al. (2001): under strict complementary slackness condition, full-rank Jacobian at optimality
- Implication

  Under nondegeneracy, there exists a neighborhood around the optimal solution where free Gauss-Newton method converges quadratically.

- Crossover techniques

Local Convergence and Crossover

- Local quadratic convergence
  - Lower bound on smallest singular value of Jacobian
  - Kruk et al. (2001): under strict complementary slackness condition, full-rank Jacobian at optimality
- Implication

  Under nondegeneracy, there exists a neighborhood around the optimal solution where free Gauss-Newton method converges quadratically.

- Crossover techniques
  - Region of quadratic convergence: heuristic rules

Local Convergence and Crossover

- Local quadratic convergence
  - Lower bound on smallest singular value of Jacobian
  - Kruk et al. (2001): under strict complementary slackness condition, full-rank Jacobian at optimality
- Implication

  Under nondegeneracy, there exists a neighborhood around the optimal solution where free Gauss-Newton method converges quadratically.

- Crossover techniques
  - Region of quadratic convergence: heuristic rules
  - Affine scaling: set barrier parameter $\mu = 0$

Local Convergence and Crossover

- Local quadratic convergence
  - Lower bound on smallest singular value of Jacobian
  - Kruk et al. (2001): under strict complementary slackness condition, full-rank Jacobian at optimality
- Implication

  Under nondegeneracy, there exists a neighborhood around the optimal solution where free Gauss-Newton method converges quadratically.

- Crossover techniques
  - Region of quadratic convergence: heuristic rules
  - Affine scaling: set barrier parameter $\mu = 0$
  - Set Gauss-Newton method free: take full step lengths $\alpha = 1$, no backtracking

Presolve and Preconditioning

Presolve and Preconditioning

- Gauss-Newton search direction

Presolve and Preconditioning

- Gauss-Newton search direction
  - Least-squares solution of $J \begin{pmatrix} \Delta \mathbf{v} \\ \Delta \mathbf{y} \end{pmatrix} = -G_\mu(\mathbf{v}, \mathbf{y}) \quad (*)$

Presolve and Preconditioning

- Gauss-Newton search direction
  - Least-squares solution of $J \begin{pmatrix} \Delta \mathbf{v} \\ \Delta \mathbf{y} \end{pmatrix} = -G_\mu(\mathbf{v}, \mathbf{y}) \quad (*)$
  - J and J^*: depend on $\mathbf{A}$ and $\mathbf{Q}$

Presolve and Preconditioning

- Gauss-Newton search direction
  - Least-squares solution of $J \begin{pmatrix} \Delta \mathbf{v} \\ \Delta \mathbf{y} \end{pmatrix} = -G_\mu(\mathbf{v}, \mathbf{y}) \quad (*)$
  - J and J^*: depend on $\mathbf{A}$ and $\mathbf{Q}$
  - Matrix-free iterative method $\rightarrow$ need to be able to calculate $J(\mathbf{v}, \mathbf{y})$ and $J^*(\mathbf{M})$ efficiently

Presolve and Preconditioning

- Gauss-Newton search direction
  - Least-squares solution of $J \begin{pmatrix} \Delta \mathbf{v} \\ \Delta \mathbf{y} \end{pmatrix} = -G_\mu(\mathbf{v}, \mathbf{y}) \quad (*)$
  - J and J^*: depend on $\mathbf{A}$ and $\mathbf{Q}$
  - Matrix-free iterative method $\rightarrow$ need to be able to calculate $J(\mathbf{v}, \mathbf{y})$ and $J^*(\mathbf{M})$ efficiently
- Presolve: exploit sparsity of $\mathbf{A}$

Presolve and Preconditioning

- Gauss-Newton search direction
  - Least-squares solution of $J \begin{pmatrix} \Delta \mathbf{v} \\ \Delta \mathbf{y} \end{pmatrix} = -G_\mu(\mathbf{v}, \mathbf{y}) \quad (*)$
  - J and J^*: depend on $\mathbf{A}$ and $\mathbf{Q}$
  - Matrix-free iterative method $\rightarrow$ need to be able to calculate $J(\mathbf{v}, \mathbf{y})$ and $J^*(\mathbf{M})$ efficiently
- Presolve: exploit sparsity of $\mathbf{A}$
  - Find permutations such that $\mathbf{P}_r \mathbf{A} \mathbf{P}_c = [\mathbf{S} \mid \mathbf{E}]$, where $\mathbf{S} \in \mathbb{R}^{m \times m}$ well-conditioned, (approx) triangular

Presolve and Preconditioning

- Gauss-Newton search direction
  - Least-squares solution of $J \begin{pmatrix} \Delta \mathbf{v} \\ \Delta \mathbf{y} \end{pmatrix} = -G_\mu(\mathbf{v}, \mathbf{y}) \quad (*)$
  - J and J^*: depend on $\mathbf{A}$ and $\mathbf{Q}$
  - Matrix-free iterative method $\rightarrow$ need to be able to calculate $J(\mathbf{v}, \mathbf{y})$ and $J^*(\mathbf{M})$ efficiently
- Presolve: exploit sparsity of $\mathbf{A}$
  - Find permutations such that $\mathbf{P}_r \mathbf{A} \mathbf{P}_c = [\mathbf{S} \mid \mathbf{E}]$, where $\mathbf{S} \in \mathbb{R}^{m \times m}$ well-conditioned, (approx) triangular
  - Sparse basis of $\mathcal{N}(\mathbf{A})$: $\mathbf{Q} = \begin{pmatrix} -\mathbf{S}^{-1}\mathbf{E} \\ \mathbf{I} \end{pmatrix}$

Presolve and Preconditioning

- Gauss-Newton search direction
  - Least-squares solution of $J \begin{pmatrix} \Delta \mathbf{v} \\ \Delta \mathbf{y} \end{pmatrix} = -G_\mu(\mathbf{v}, \mathbf{y}) \quad (*)$
  - J and J^*: depend on $\mathbf{A}$ and $\mathbf{Q}$
  - Matrix-free iterative method $\rightarrow$ need to be able to calculate $J(\mathbf{v}, \mathbf{y})$ and $J^*(\mathbf{M})$ efficiently
- Presolve: exploit sparsity of $\mathbf{A}$
  - Find permutations such that $\mathbf{P}_r \mathbf{A} \mathbf{P}_c = [\mathbf{S} \mid \mathbf{E}]$, where $\mathbf{S} \in \mathbb{R}^{m \times m}$ well-conditioned, (approx) triangular
  - Sparse basis of $\mathcal{N}(\mathbf{A})$: $\mathbf{Q} = \begin{pmatrix} -\mathbf{S}^{-1}\mathbf{E} \\ \mathbf{I} \end{pmatrix}$
- Preconditioning: improve condition of $(*)$ by changing variables with nonsingular matrix $\mathbf{D}$

Presolve and Preconditioning

- Gauss-Newton search direction
  - Least-squares solution of $J \begin{pmatrix} \Delta \mathbf{v} \\ \Delta \mathbf{y} \end{pmatrix} = -G_\mu(\mathbf{v}, \mathbf{y}) \quad (*)$
  - J and J^*: depend on $\mathbf{A}$ and $\mathbf{Q}$
  - Matrix-free iterative method $\rightarrow$ need to be able to calculate $J(\mathbf{v}, \mathbf{y})$ and $J^*(\mathbf{M})$ efficiently
- Presolve: exploit sparsity of $\mathbf{A}$
  - Find permutations such that $\mathbf{P}_r \mathbf{A} \mathbf{P}_c = [\mathbf{S} \mid \mathbf{E}]$, where $\mathbf{S} \in \mathbb{R}^{m \times m}$ well-conditioned, (approx) triangular
  - Sparse basis of $\mathcal{N}(\mathbf{A})$: $\mathbf{Q} = \begin{pmatrix} -\mathbf{S}^{-1}\mathbf{E} \\ \mathbf{I} \end{pmatrix}$
- Preconditioning: improve condition of $(*)$ by changing variables with nonsingular matrix $\mathbf{D}$
  - Minimize the ω-condition number, $\omega(\mathbf{X}) = \dfrac{\text{trace}(\mathbf{X})/n}{\det(\mathbf{X})^{\frac{1}{n}}}$ for $\mathbf{X} \succ 0$

Presolve and Preconditioning

- Gauss-Newton search direction
  - Least-squares solution of $J \begin{pmatrix} \Delta \mathbf{v} \\ \Delta \mathbf{y} \end{pmatrix} = -G_\mu(\mathbf{v}, \mathbf{y}) \quad (*)$
  - J and J^*: depend on $\mathbf{A}$ and $\mathbf{Q}$
  - Matrix-free iterative method $\rightarrow$ need to be able to calculate $J(\mathbf{v}, \mathbf{y})$ and $J^*(\mathbf{M})$ efficiently
- Presolve: exploit sparsity of $\mathbf{A}$
  - Find permutations such that $\mathbf{P}_r \mathbf{A} \mathbf{P}_c = [\mathbf{S} \mid \mathbf{E}]$, where $\mathbf{S} \in \mathbb{R}^{m \times m}$ well-conditioned, (approx) triangular
  - Sparse basis of $\mathcal{N}(\mathbf{A})$: $\mathbf{Q} = \begin{pmatrix} -\mathbf{S}^{-1}\mathbf{E} \\ \mathbf{I} \end{pmatrix}$
- Preconditioning: improve condition of $(*)$ by changing variables with nonsingular matrix $\mathbf{D}$
  - Minimize the ω-condition number, $\omega(\mathbf{X}) = \dfrac{\text{trace}(\mathbf{X})/n}{\det(\mathbf{X})^{\frac{1}{n}}}$ for $\mathbf{X} \succ 0$
  - ω-optimal diagonal preconditioner: commonly used, $D_{ii} = 1/\left\|\mathbf{J}_{:,i}\right\|$

Presolve and Preconditioning

- Gauss-Newton search direction
  - Least-squares solution of $J \begin{pmatrix} \Delta \mathbf{v} \\ \Delta \mathbf{y} \end{pmatrix} = -G_\mu(\mathbf{v}, \mathbf{y}) \quad (*)$
  - J and J^*: depend on $\mathbf{A}$ and $\mathbf{Q}$
  - Matrix-free iterative method $\rightarrow$ need to be able to calculate $J(\mathbf{v}, \mathbf{y})$ and $J^*(\mathbf{M})$ efficiently
- Presolve: exploit sparsity of $\mathbf{A}$
  - Find permutations such that $\mathbf{P}_r \mathbf{A} \mathbf{P}_c = [\mathbf{S} \mid \mathbf{E}]$, where $\mathbf{S} \in \mathbb{R}^{m \times m}$ well-conditioned, (approx) triangular
  - Sparse basis of $\mathcal{N}(\mathbf{A})$: $\mathbf{Q} = \begin{pmatrix} -\mathbf{S}^{-1}\mathbf{E} \\ \mathbf{I} \end{pmatrix}$
- Preconditioning: improve condition of $(*)$ by changing variables with nonsingular matrix $\mathbf{D}$
  - Minimize the ω-condition number, $\omega(\mathbf{X}) = \dfrac{\text{trace}(\mathbf{X})/n}{\det(\mathbf{X})^{\frac{1}{n}}}$ for $\mathbf{X} \succ 0$
  - ω-optimal diagonal preconditioner: commonly used, $D_{ii} = 1/\left\|\mathbf{J}_{:,i}\right\|$
  - ω-optimal diagonal block preconditioner

Lovász Theta Function Problem

- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ undirected graph, $n = |\mathcal{V}|$ nodes, $m = |\mathcal{E}|$ edges

Lovász Theta Function Problem

- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ undirected graph, $n = |\mathcal{V}|$ nodes, $m = |\mathcal{E}|$ edges
- $\mathbf{E}$ matrix of all ones, $\mathbf{E}_{ij} = (\mathbf{e}_i \mathbf{e}_j^T + \mathbf{e}_j \mathbf{e}_i^T)/\sqrt{2}$ ij-th unit matrix, $\mathbf{e}_i$ i-th unit vector

Lovász Theta Function Problem

- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ undirected graph, $n = |\mathcal{V}|$ nodes, $m = |\mathcal{E}|$ edges
- $\mathbf{E}$ matrix of all ones, $\mathbf{E}_{ij} = (\mathbf{e}_i \mathbf{e}_j^T + \mathbf{e}_j \mathbf{e}_i^T)/\sqrt{2}$ ij-th unit matrix, $\mathbf{e}_i$ i-th unit vector
- Lovász theta number $\vartheta(\mathcal{G})$: bounds for stability and chromatic numbers of $\mathcal{G}$

Lovász Theta Function Problem

- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ undirected graph, $n = |\mathcal{V}|$ nodes, $m = |\mathcal{E}|$ edges
- $\mathbf{E}$ matrix of all ones, $\mathbf{E}_{ij} = (\mathbf{e}_i \mathbf{e}_j^T + \mathbf{e}_j \mathbf{e}_i^T)/\sqrt{2}$ ij-th unit matrix, $\mathbf{e}_i$ i-th unit vector
- Lovász theta number $\vartheta(\mathcal{G})$: bounds for stability and chromatic numbers of $\mathcal{G}$

$$
\text{(TN)} \qquad
\begin{aligned}
\vartheta(\mathcal{G}) := p^* := \quad & \max \quad && \mathbf{E} \cdot \mathbf{X} \\
& \text{s.t.} \quad && \mathbf{I} \cdot \mathbf{X} = 1 \\
& && \mathbf{E}_{ij} \cdot \mathbf{X} = 0, \quad \forall\, (i,j) \in \mathcal{E} \\
& && \mathbf{X} \succeq 0
\end{aligned}
$$

Lovász Theta Function Problem

- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ undirected graph, $n = |\mathcal{V}|$ nodes, $m = |\mathcal{E}|$ edges
- $\mathbf{E}$ matrix of all ones, $\mathbf{E}_{ij} = (\mathbf{e}_i \mathbf{e}_j^T + \mathbf{e}_j \mathbf{e}_i^T)/\sqrt{2}$ ij-th unit matrix, $\mathbf{e}_i$ i-th unit vector
- Lovász theta number $\vartheta(\mathcal{G})$: bounds for stability and chromatic numbers of $\mathcal{G}$

(TN)
$$\begin{aligned} \vartheta(\mathcal{G}) := p^* := \quad & \max \quad \mathbf{E} \cdot \mathbf{X} \\ & \text{s.t.} \quad \mathbf{I} \cdot \mathbf{X} = 1 \\ & \quad\;\; \mathbf{E}_{ij} \cdot \mathbf{X} = 0, \quad \forall (i,j) \in \mathcal{E} \\ & \quad\;\; \mathbf{X} \succeq 0 \end{aligned}$$

(DTN)
$$\begin{aligned} d^* := \quad & \min \quad z \\ & \text{s.t.} \quad z\mathbf{I} + \sum_{(i,j)\in\mathcal{E}} y_{ij}\mathbf{E}_{ij} - \mathbf{Z} = \mathbf{E}, \\ & \quad\;\; \mathbf{Z} \succeq 0, \end{aligned}$$

Gauss-Newton Approach for $\vartheta(\mathcal{G})$

- Null space representation: **A** and **Q** are sparse

Gauss-Newton Approach for $\vartheta(\mathcal{G})$

- Null space representation: $\mathbf{A}$ and $\mathbf{Q}$ are sparse
  - Dual feasibility $\mathbf{Z} = -\mathbf{E} + z\mathbf{I} + \mathrm{sMat}_{\mathcal{E}}(\mathbf{y})$

Gauss-Newton Approach for $\vartheta(\mathcal{G})$

- Null space representation: $\mathbf{A}$ and $\mathbf{Q}$ are sparse
  - Dual feasibility $\mathbf{Z} = -\mathbf{E} + z\mathbf{I} + \mathrm{sMat}_{\mathcal{E}}(\mathbf{y})$
  - Primal feasibility, $\mathbf{V} = \begin{pmatrix} -\mathbf{e}^T \\ \mathbf{I} \end{pmatrix} \in \mathbb{R}^{n\times(n-1)}$:

Gauss-Newton Approach for $\vartheta(\mathcal{G})$

- Null space representation: $\mathbf{A}$ and $\mathbf{Q}$ are sparse
  - Dual feasibility $\mathbf{Z} = -\mathbf{E} + z\mathbf{I} + \mathrm{sMat}_{\mathcal{E}}(\mathbf{y})$
  - Primal feasibility, $\mathbf{V} = \begin{pmatrix} -\mathbf{e}^T \\ \mathbf{I} \end{pmatrix} \in \mathbb{R}^{n\times(n-1)}$:

$$\mathbf{X} = \frac{1}{n}\mathbf{I} + \mathrm{Diag}(\mathbf{V}\mathbf{w}) + \mathrm{sMat}_{\mathcal{E}^c}(\mathbf{v})$$

Gauss-Newton Approach for $\vartheta(\mathcal{G})$

- Null space representation: $\mathbf{A}$ and $\mathbf{Q}$ are sparse
  - Dual feasibility $\mathbf{Z} = -\mathbf{E} + z\mathbf{I} + \mathrm{sMat}_{\mathcal{E}}(\mathbf{y})$
  - Primal feasibility, $\mathbf{V} = \begin{pmatrix} -\mathbf{e}^T \\ \mathbf{I} \end{pmatrix} \in \mathbb{R}^{n\times(n-1)}$:

$$\mathbf{X} = \frac{1}{n}\mathbf{I} + \mathrm{Diag}(\mathbf{V}\mathbf{w}) + \mathrm{sMat}_{\mathcal{E}^c}(\mathbf{v})$$

- Presolve
  - Permutations of $\mathbf{A}$: depend of orders of nodes and indices of edges

Gauss-Newton Approach for $\vartheta(\mathcal{G})$

- Null space representation: $\mathbf{A}$ and $\mathbf{Q}$ are sparse
  - Dual feasibility $\mathbf{Z} = -\mathbf{E} + z\mathbf{I} + \text{sMat}_{\mathcal{E}}(\mathbf{y})$
  - Primal feasibility, $\mathbf{V} = \begin{pmatrix} -\mathbf{e}^T \\ \mathbf{I} \end{pmatrix} \in \mathbb{R}^{n\times(n-1)}$:

$$\mathbf{X} = \frac{1}{n}\mathbf{I} + \text{Diag}(\mathbf{V}\mathbf{w}) + \text{sMat}_{\mathcal{E}^c}(\mathbf{v})$$

- Presolve
  - Permutations of $\mathbf{A}$: depend of orders of nodes and indices of edges
- Preconditioning
  - Simple formulation for both ω-optimal diagonal and block diagonal preconditioners

Gauss-Newton Approach for $\vartheta(\mathcal{G})$

- Null space representation: $\mathbf{A}$ and $\mathbf{Q}$ are sparse
  - Dual feasibility $\mathbf{Z} = -\mathbf{E} + z\mathbf{I} + \text{sMat}_{\mathcal{E}}(\mathbf{y})$
  - Primal feasibility, $\mathbf{V} = \begin{pmatrix} -\mathbf{e}^T \\ \mathbf{I} \end{pmatrix} \in \mathbb{R}^{n\times(n-1)}$:

$$\mathbf{X} = \frac{1}{n}\mathbf{I} + \text{Diag}(\mathbf{V}\mathbf{w}) + \text{sMat}_{\mathcal{E}^c}(\mathbf{v})$$

- Presolve
  - Permutations of $\mathbf{A}$: depend of orders of nodes and indices of edges
- Preconditioning
  - Simple formulation for both ω-optimal diagonal and block diagonal preconditioners
  - ω-optimal diagonal preconditioner

$$D_{ii} = 1/\sqrt{\left\|\mathbf{Z}_{:,k(i)}\right\|^2 + \left\|\mathbf{Z}_{:,l(i)}\right\|^2} \text{ or } D_{ii} = 1/\sqrt{\left\|\mathbf{X}_{k(i),:}\right\|^2 + \left\|\mathbf{X}_{l(i),:}\right\|^2}$$

Numerics

Numerics

- Three versions of the Gauss-Newton algorithm
  - General version for full matrices
  - Sparse version for sparse matrices
  - Specialized version for Lovász theta function problem

Numerics

- Three versions of the Gauss-Newton algorithm
  - General version for full matrices
  - Sparse version for sparse matrices
  - Specialized version for Lovász theta function problem
- Three different preconditioners
  - Diagonal
  - Two block diagonal: $J = [\mathcal{Z} \mid \mathcal{X}]$
  - Multiple block diagonal (for Lovász theta function problem)

Numerics

- Three versions of the Gauss-Newton algorithm
  - General version for full matrices
  - Sparse version for sparse matrices
  - Specialized version for Lovász theta function problem
- Three different preconditioners
  - Diagonal
  - Two block diagonal: $J = [\mathcal{Z} \mid \mathcal{X}]$
  - Multiple block diagonal (for Lovász theta function problem)
- MATLAB code, LSMR for Gauss-Newton search directions (Fong and Saunders (2010))

Numerics

- Three versions of the Gauss-Newton algorithm
  - General version for full matrices
  - Sparse version for sparse matrices
  - Specialized version for Lovász theta function problem
- Three different preconditioners
  - Diagonal
  - Two block diagonal: $J = [\mathcal{Z} \mid \mathcal{X}]$
  - Multiple block diagonal (for Lovász theta function problem)
- MATLAB code, LSMR for Gauss-Newton search directions (Fong and Saunders (2010))
- Comparison with CSDP, SDPA, SeDuMi, and SDPT3

Numerics

- Three versions of the Gauss-Newton algorithm
  - General version for full matrices
  - Sparse version for sparse matrices
  - Specialized version for Lovász theta function problem
- Three different preconditioners
  - Diagonal
  - Two block diagonal: $J = [\mathcal{Z} \mid \mathcal{X}]$
  - Multiple block diagonal (for Lovász theta function problem)
- MATLAB code, LSMR for Gauss-Newton search directions (Fong and Saunders (2010))
- Comparison with CSDP, SDPA, SeDuMi, and SDPT3
- Performance measures
  - DIMACS errors

Numerics

- Three versions of the Gauss-Newton algorithm
  - General version for full matrices
  - Sparse version for sparse matrices
  - Specialized version for Lovász theta function problem
- Three different preconditioners
  - Diagonal
  - Two block diagonal: $J = [\mathcal{Z} \mid \mathcal{X}]$
  - Multiple block diagonal (for Lovász theta function problem)
- MATLAB code, LSMR for Gauss-Newton search directions (Fong and Saunders (2010))
- Comparison with CSDP, SDPA, SeDuMi, and SDPT3
- Performance measures
  - DIMACS errors
  - $\text{RelZXnorm} = \dfrac{\|\mathbf{ZX}\|_F}{|\mathbf{C} \cdot \mathbf{X}| + 1}, \quad \text{Relmineig} = \dfrac{\min\{\lambda_{\min}(\mathbf{X}), \lambda_{\min}(\mathbf{Z})\}}{|\mathbf{C} \cdot \mathbf{X}| + 1}$

Numerics

- Three versions of the Gauss-Newton algorithm
  - General version for full matrices
  - Sparse version for sparse matrices
  - Specialized version for Lovász theta function problem
- Three different preconditioners
  - Diagonal
  - Two block diagonal: $J = [\mathcal{Z} \mid \mathcal{X}]$
  - Multiple block diagonal (for Lovász theta function problem)
- MATLAB code, LSMR for Gauss-Newton search directions (Fong and Saunders (2010))
- Comparison with CSDP, SDPA, SeDuMi, and SDPT3
- Performance measures
  - DIMACS errors
  - RelZXnorm $= \dfrac{\|\mathbf{ZX}\|_F}{|\mathbf{C} \cdot \mathbf{X}| + 1}$, Relmineig $= \dfrac{\min\{\lambda_{\min}(\mathbf{X}), \lambda_{\min}(\mathbf{Z})\}}{|\mathbf{C} \cdot \mathbf{X}| + 1}$
  - Computational time

Preconditioner Testing

- Random sparse matrices, algorithm with crossover

Preconditioner Testing

- Random sparse matrices, algorithm with crossover
- Ratios of numbers of LSMR iterations

n	10	20	30	40	50	60	70	80	90	100
(D)	0.67	0.55	0.55	0.39	0.42	0.42	0.47	0.36	0.34	0.32
(BD)	0.30	0.16	0.13	0.09	0.09	0.08	0.08	0.07	0.07	0.06

Preconditioner Testing

- Random sparse matrices, algorithm with crossover
- Ratios of numbers of LSMR iterations

n	10	20	30	40	50	60	70	80	90	100
(D)	0.67	0.55	0.55	0.39	0.42	0.42	0.47	0.36	0.34	0.32
(BD)	0.30	0.16	0.13	0.09	0.09	0.08	0.08	0.07	0.07	0.06

- Ratios of total computational time

n	10	20	30	40	50	60	70	80	90	100
(D)	0.63	0.58	0.59	0.45	0.49	0.51	0.62	0.45	0.48	0.44
(BD)	0.33	0.27	0.46	0.57	1.05	1.44	2.20	2.26	3.50	3.26

Preconditioner Testing

- Random sparse matrices, algorithm with crossover
- Ratios of numbers of LSMR iterations

n	10	20	30	40	50	60	70	80	90	100
(D)	0.67	0.55	0.55	0.39	0.42	0.42	0.47	0.36	0.34	0.32
(BD)	0.30	0.16	0.13	0.09	0.09	0.08	0.08	0.07	0.07	0.06

- Ratios of total computational time

n	10	20	30	40	50	60	70	80	90	100
(D)	0.63	0.58	0.59	0.45	0.49	0.51	0.62	0.45	0.48	0.44
(BD)	0.33	0.27	0.46	0.57	1.05	1.44	2.20	2.26	3.50	3.26

- Recommendation: ω-optimal diagonal preconditioner

Accuracy Testing

- Random sparse matrices, $n = m = 100$

Accuracy Testing

- Random sparse matrices, $n = m = 100$

	(D)	SeDuMi	CSDP	SDPA	SDPT3
Iteration	23.45	19.15	16.50	16.30	24.50

Accuracy Testing

- Random sparse matrices, $n = m = 100$

	(D)	SeDuMi	CSDP	SDPA	SDPT3
Iteration	23.45	19.15	16.50	16.30	24.50
RelZXnorm	7.18×10^{-15}	1.50×10^{-07}	8.92×10^{-08}	5.86×10^{-08}	1.01×10^{-10}
Relmineig	-1.12×10^{-15}	-8.72×10^{-15}	3.28×10^{-13}	1.46×10^{-13}	5.99×10^{-17}

Accuracy Testing

- Random sparse matrices, $n = m = 100$

	(D)	SeDuMi	CSDP	SDPA	SDPT3
Iteration	23.45	19.15	16.50	16.30	24.50
RelZXnorm	7.18×10^{-15}	1.50×10^{-07}	8.92×10^{-08}	5.86×10^{-08}	1.01×10^{-10}
Relmineig	-1.12×10^{-15}	-8.72×10^{-15}	3.28×10^{-13}	1.46×10^{-13}	5.99×10^{-17}
DIMACS1	1.28×10^{-15}	7.93×10^{-11}	1.24×10^{-13}	2.48×10^{-13}	3.19×10^{-13}
DIMACS2	2.28×10^{-14}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}
DIMACS3	9.39×10^{-16}	3.53×10^{-16}	1.64×10^{-08}	1.44×10^{-15}	8.72×10^{-14}
DIMACS4	3.22×10^{-15}	3.42×10^{-13}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}
DIMACS5	1.19×10^{-15}	9.72×10^{-13}	1.54×10^{-09}	4.96×10^{-10}	1.10×10^{-13}
DIMACS6	4.65×10^{-16}	2.84×10^{-14}	1.16×10^{-09}	4.96×10^{-10}	2.01×10^{-13}

Accuracy Testing

- Random sparse matrices, $n = m = 100$

	(D)	SeDuMi	CSDP	SDPA	SDPT3
Iteration	23.45	19.15	16.50	16.30	24.50
RelZXnorm	7.18×10^{-15}	1.50×10^{-07}	8.92×10^{-08}	5.86×10^{-08}	1.01×10^{-10}
Relmineig	-1.12×10^{-15}	-8.72×10^{-15}	3.28×10^{-13}	1.46×10^{-13}	5.99×10^{-17}
DIMACS1	1.28×10^{-15}	7.93×10^{-11}	1.24×10^{-13}	2.48×10^{-13}	3.19×10^{-13}
DIMACS2	2.28×10^{-14}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}
DIMACS3	9.39×10^{-16}	3.53×10^{-16}	1.64×10^{-08}	1.44×10^{-15}	8.72×10^{-14}
DIMACS4	3.22×10^{-15}	3.42×10^{-13}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}
DIMACS5	1.19×10^{-15}	9.72×10^{-13}	1.54×10^{-09}	4.96×10^{-10}	1.10×10^{-13}
DIMACS6	4.65×10^{-16}	2.84×10^{-14}	1.16×10^{-09}	4.96×10^{-10}	2.01×10^{-13}
Time	61.89	1.23	0.60	0.74	1.13

Computational Time Testing

Computational Time Testing

- Change in number of constraints

Computational Time Testing

- Change in number of constraints
  - $n = 100$, $m = 500$ to 5000, same sparseness density

Computational Time Testing

- Change in number of constraints
  - $n = 100$, $m = 500$ to 5000, same sparseness density

m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
SeDuMi	168.74	49.99	21.47	4.18	1.92	1.89	1.09	0.66	0.68	0.32
CSDP	144.02	39.62	21.64	7.58	5.89	5.56	3.53	2.40	2.88	1.62
SDPA	488.28	203.06	134.49	50.99	37.09	33.37	20.98	15.55	17.83	9.41
SDPT3	87.49	27.96	16.10	6.00	3.80	4.42	2.11	2.30	2.89	1.81

Computational Time Testing

- Change in number of constraints
  - $n = 100$, $m = 500$ to 5000, same sparseness density

m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
SeDuMi	168.74	49.99	21.47	4.18	1.92	1.89	1.09	0.66	0.68	0.32
CSDP	144.02	39.62	21.64	7.58	5.89	5.56	3.53	2.40	2.88	1.62
SDPA	488.28	203.06	134.49	50.99	37.09	33.37	20.98	15.55	17.83	9.41
SDPT3	87.49	27.96	16.10	6.00	3.80	4.42	2.11	2.30	2.89	1.81

- Change in sparseness density

Computational Time Testing

- Change in number of constraints
  - $n = 100$, $m = 500$ to 5000, same sparseness density

m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
SeDuMi	168.74	49.99	21.47	4.18	1.92	1.89	1.09	0.66	0.68	0.32
CSDP	144.02	39.62	21.64	7.58	5.89	5.56	3.53	2.40	2.88	1.62
SDPA	488.28	203.06	134.49	50.99	37.09	33.37	20.98	15.55	17.83	9.41
SDPT3	87.49	27.96	16.10	6.00	3.80	4.42	2.11	2.30	2.89	1.81

- Change in sparseness density
  - $n = 100$, $m = 2500$, sparseness density $1/(4sn)$, $s = 1, \ldots, 10$

Computational Time Testing

- Change in number of constraints
  - $n = 100$, $m = 500$ to 5000, same sparseness density

m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
SeDuMi	168.74	49.99	21.47	4.18	1.92	1.89	1.09	0.66	0.68	0.32
CSDP	144.02	39.62	21.64	7.58	5.89	5.56	3.53	2.40	2.88	1.62
SDPA	488.28	203.06	134.49	50.99	37.09	33.37	20.98	15.55	17.83	9.41
SDPT3	87.49	27.96	16.10	6.00	3.80	4.42	2.11	2.30	2.89	1.81

- Change in sparseness density
  - $n = 100$, $m = 2500$, sparseness density $1/(4sn)$, $s = 1, \ldots, 10$

s	1	2	3	4	5	6	7	8	9	10
SeDuMi	2.84	2.74	1.26	1.15	1.49	0.62	0.53	0.49	0.34	0.30
CSDP	7.33	12.17	6.07	6.77	8.06	4.14	3.21	3.39	2.49	2.17
SDPA	43.35	48.63	22.13	20.95	25.47	12.33	9.25	9.82	7.08	6.04
SDPT3	4.60	8.82	5.79	5.74	7.50	3.94	2.86	2.75	2.27	2.09

Random Hard Instances I

Random Hard Instances I

- Strict complementarity fails (Wei and Wolkowicz (2010))

Random Hard Instances I

- Strict complementarity fails (Wei and Wolkowicz (2010))
- $n = 50, m = 1000$, algorithm without crossover, block diagonal preconditioner

Random Hard Instances I

- Strict complementarity fails (Wei and Wolkowicz (2010))
- $n = 50, m = 1000$, algorithm without crossover, block diagonal preconditioner

	(BD)	SeDuMi	CSDP	SDPA	SDPT3
Iteration	29.20	13.60	12.10	12.00	22.70

Random Hard Instances I

- Strict complementarity fails (Wei and Wolkowicz (2010))
- $n = 50, m = 1000$, algorithm without crossover, block diagonal preconditioner

	(BD)	SeDuMi	CSDP	SDPA	SDPT3
Iteration	29.20	13.60	12.10	12.00	22.70
RelZXnorm	1.32×10^{-12}	4.12×10^{-06}	6.53×10^{-05}	1.91×10^{-07}	2.97×10^{-08}
Relmineig	5.96×10^{-20}	6.20×10^{-14}	1.44×10^{-15}	3.25×10^{-12}	1.21×10^{-18}

Random Hard Instances I

- Strict complementarity fails (Wei and Wolkowicz (2010))
- $n = 50, m = 1000$, algorithm without crossover, block diagonal preconditioner

	(BD)	SeDuMi	CSDP	SDPA	SDPT3
Iteration	29.20	13.60	12.10	12.00	22.70
RelZXnorm	1.32×10^{-12}	4.12×10^{-06}	6.53×10^{-05}	1.91×10^{-07}	2.97×10^{-08}
Relmineig	5.96×10^{-20}	6.20×10^{-14}	1.44×10^{-15}	3.25×10^{-12}	1.21×10^{-18}
DIMACS1	2.16×10^{-12}	2.01×10^{-07}	3.50×10^{-10}	3.79×10^{-07}	4.62×10^{-10}
DIMACS2	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}
DIMACS3	1.81×10^{-11}	6.34×10^{-15}	2.10×10^{-08}	2.09×10^{-14}	1.54×10^{-11}
DIMACS4	0.00×10^{00}	1.48×10^{-14}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}
DIMACS5	2.07×10^{-12}	9.23×10^{-09}	1.23×10^{-08}	1.58×10^{-07}	5.70×10^{-11}
DIMACS6	1.86×10^{-12}	4.99×10^{-09}	4.25×10^{-08}	3.50×10^{-07}	1.06×10^{-10}

Random Hard Instances I

- Strict complementarity fails (Wei and Wolkowicz (2010))
- $n = 50, m = 1000$, algorithm without crossover, block diagonal preconditioner

	(BD)	SeDuMi	CSDP	SDPA	SDPT3
Iteration	29.20	13.60	12.10	12.00	22.70
RelZXnorm	1.32×10^{-12}	4.12×10^{-06}	6.53×10^{-05}	1.91×10^{-07}	2.97×10^{-08}
Relmineig	5.96×10^{-20}	6.20×10^{-14}	1.44×10^{-15}	3.25×10^{-12}	1.21×10^{-18}
DIMACS1	2.16×10^{-12}	2.01×10^{-07}	3.50×10^{-10}	3.79×10^{-07}	4.62×10^{-10}
DIMACS2	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}
DIMACS3	1.81×10^{-11}	6.34×10^{-15}	2.10×10^{-08}	2.09×10^{-14}	1.54×10^{-11}
DIMACS4	0.00×10^{00}	1.48×10^{-14}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}
DIMACS5	2.07×10^{-12}	9.23×10^{-09}	1.23×10^{-08}	1.58×10^{-07}	5.70×10^{-11}
DIMACS6	1.86×10^{-12}	4.99×10^{-09}	4.25×10^{-08}	3.50×10^{-07}	1.06×10^{-10}
Time	281.45	64.02	69.50	27.31	60.25

Random Hard Instances II

- Dual Slater's condition (almost) fails

Random Hard Instances II

- Dual Slater's condition (almost) fails
- $n = 50, m = 1000$, algorithm without crossover, block diagonal preconditioner

Random Hard Instances II

- Dual Slater's condition (almost) fails
- $n = 50, m = 1000$, algorithm without crossover, block diagonal preconditioner

	(BD)	SeDuMi	CSDP	SDPA	SDPT3
Iteration	26.00	17.10	13.80	15.20	20.40

Random Hard Instances II

- Dual Slater's condition (almost) fails
- $n = 50, m = 1000$, algorithm without crossover, block diagonal preconditioner

	(BD)	SeDuMi	CSDP	SDPA	SDPT3
Iteration	26.00	17.10	13.80	15.20	20.40
RelZXnorm	9.90×10^{-16}	6.14×10^{-07}	4.61×10^{-07}	2.05×10^{-07}	2.13×10^{-08}
Relmineig	1.05×10^{-17}	1.06×10^{-15}	1.60×10^{-12}	1.06×10^{-10}	1.47×10^{-15}

Random Hard Instances II

- Dual Slater's condition (almost) fails
- $n = 50, m = 1000$, algorithm without crossover, block diagonal preconditioner

	(BD)	SeDuMi	CSDP	SDPA	SDPT3
Iteration	26.00	17.10	13.80	15.20	20.40
RelZXnorm	9.90×10^{-16}	6.14×10^{-07}	4.61×10^{-07}	2.05×10^{-07}	2.13×10^{-08}
Relmineig	1.05×10^{-17}	1.06×10^{-15}	1.60×10^{-12}	1.06×10^{-10}	1.47×10^{-15}
DIMACS1	2.78×10^{-13}	1.07×10^{-10}	1.10×10^{-12}	1.17×10^{-13}	8.41×10^{-11}
DIMACS2	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}
DIMACS3	1.42×10^{-12}	3.56×10^{-14}	5.21×10^{-09}	6.83×10^{-08}	2.59×10^{-13}
DIMACS4	0.00×10^{00}	1.48×10^{-14}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}
DIMACS5	1.09×10^{-14}	2.77×10^{-12}	1.38×10^{-09}	2.02×10^{-14}	1.26×10^{-11}
DIMACS6	3.44×10^{-15}	3.16×10^{-12}	8.54×10^{-10}	4.43×10^{-08}	1.82×10^{-12}

Random Hard Instances II

- Dual Slater's condition (almost) fails
- $n = 50, m = 1000$, algorithm without crossover, block diagonal preconditioner

	(BD)	SeDuMi	CSDP	SDPA	SDPT3
Iteration	26.00	17.10	13.80	15.20	20.40
RelZXnorm	9.90×10^{-16}	6.14×10^{-07}	4.61×10^{-07}	2.05×10^{-07}	2.13×10^{-08}
Relmineig	1.05×10^{-17}	1.06×10^{-15}	1.60×10^{-12}	1.06×10^{-10}	1.47×10^{-15}
DIMACS1	2.78×10^{-13}	1.07×10^{-10}	1.10×10^{-12}	1.17×10^{-13}	8.41×10^{-11}
DIMACS2	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}
DIMACS3	1.42×10^{-12}	3.56×10^{-14}	5.21×10^{-09}	6.83×10^{-08}	2.59×10^{-13}
DIMACS4	0.00×10^{00}	1.48×10^{-14}	0.00×10^{00}	0.00×10^{00}	0.00×10^{00}
DIMACS5	1.09×10^{-14}	2.77×10^{-12}	1.38×10^{-09}	2.02×10^{-14}	1.26×10^{-11}
DIMACS6	3.44×10^{-15}	3.16×10^{-12}	8.54×10^{-10}	4.43×10^{-08}	1.82×10^{-12}
Time	58.04	75.02	76.42	32.32	48.98

Lovász Theta Function Problem

Lovász Theta Function Problem

- Random graph, $n = |\mathcal{V}| = 100$, $|\mathcal{E}| = n(n-1)/4$

Lovász Theta Function Problem

- Random graph, $n = |\mathcal{V}| = 100$, $|\mathcal{E}| = n(n-1)/4$
- Most difficult instances to solve (Gruber and Rendl (2002))

Lovász Theta Function Problem

- Random graph, $n = |\mathcal{V}| = 100$, $|\mathcal{E}| = n(n-1)/4$
- Most difficult instances to solve (Gruber and Rendl (2002))
- Specialized version of the code, multiple block diagonal preconditioner

Lovász Theta Function Problem

- Random graph, $n = |\mathcal{V}| = 100$, $|\mathcal{E}| = n(n-1)/4$
- Most difficult instances to solve (Gruber and Rendl (2002))
- Specialized version of the code, multiple block diagonal preconditioner

	Iteration	RelZXnorm	Relmineig
Average	18.31	4.85×10^{-15}	2.47×10^{-13}
Best	16.00	6.42×10^{-17}	-1.28×10^{-15}
Worst	23.00	9.74×10^{-13}	-3.58×10^{-11}

Lovász Theta Function Problem

- Random graph, $n = |\mathcal{V}| = 100$, $|\mathcal{E}| = n(n-1)/4$
- Most difficult instances to solve (Gruber and Rendl (2002))
- Specialized version of the code, multiple block diagonal preconditioner

	Iteration	RelZXnorm	Relmineig
Average	18.31	4.85×10^{-15}	2.47×10^{-13}
Best	16.00	6.42×10^{-17}	-1.28×10^{-15}
Worst	23.00	9.74×10^{-13}	-3.58×10^{-11}

- Comparison with other versions

	(D)	(SD)	(SMBD)
Iteration	23.90	21.00	21.00
RelZXnorm	5.62×10^{-14}	3.26×10^{-14}	2.67×10^{-14}
Relmineig	2.23×10^{-12}	9.51×10^{-13}	6.74×10^{-13}
Time	214.99	126.74	158.09

Summary

- Matrix-free, path following method with Gauss-Newton approach

Summary

- Matrix-free, path following method with Gauss-Newton approach
  - Null space representation
  - Sparsity exploitation
  - Presolve and preconditioning
  - Crossover techniques

Summary

- Matrix-free, path following method with Gauss-Newton approach
  - Null space representation
  - Sparsity exploitation
  - Presolve and preconditioning
  - Crossover techniques
- Numerical tests

Summary

- Matrix-free, path following method with Gauss-Newton approach
  - Null space representation
  - Sparsity exploitation
  - Presolve and preconditioning
  - Crossover techniques
- Numerical tests
  - Improved accuracy in random (hard) instances
  - Simple diagonal preconditioner recommended
  - Preferred for problems with high number of constraints
  - Specialized algorithm

Summary

- Matrix-free, path following method with Gauss-Newton approach
  - Null space representation
  - Sparsity exploitation
  - Presolve and preconditioning
  - Crossover techniques
- Numerical tests
  - Improved accuracy in random (hard) instances
  - Simple diagonal preconditioner recommended
  - Preferred for problems with high number of constraints
  - Specialized algorithm
- Future direction

Summary

- Matrix-free, path following method with Gauss-Newton approach
  - Null space representation
  - Sparsity exploitation
  - Presolve and preconditioning
  - Crossover techniques
- Numerical tests
  - Improved accuracy in random (hard) instances
  - Simple diagonal preconditioner recommended
  - Preferred for problems with high number of constraints
  - Specialized algorithm
- Future direction
  - Computational time improvement
  - General algorithm for conic programs

Thank You!