Innere-Punkte-Methoden zur Lösung semidefiniter Programme: Vergleich verschiedener Suchrichtungen

Nils Meißner

Diplomarbeit
bei Professor M. Grötschel
an der Technischen Universität Berlin

7. Januar 2000

Kapitel 0

Einleitung

Eine Reihe von Autoren haben bereits die Verallgemeinerung von Innere-Punkte-Algorithmen der Linearen Optimierung in den Kontext der Semidefiniten Optimierung diskutiert. Ein wegweisendes Werk in diese Richtung ist von Nesterov und Nemirovkii, zusammengefaßt in ihrem Buch [22]. Sie zeigen darin, daß das Problem der Minimierung einer linearen Funktion über einer konvexen Menge in „polynomieller Zeit“ gelöst werden kann, solange eine sogenannte selbst-konkordante Barriere-Funktion für die konvexe Menge bekannt ist. Im besonderen zeigen Nesterov und Nemirovskii, daß lineare Programme, konvex-quadratische Programme mit konvex-quadratischen Nebenbedingungen und semidefinite Programme alle explizite leicht aufzustellende selbst-konkordante Barriere-Funktionen besitzen und damit in „polynomieller Zeit“ gelöst werden können.
Weiterhin erweitert Alizadeh [1] den Innere-Punkte-Algorithmus für Lineare Optimierung von Ye [39] zur Semidefiniten Optimierung und argumentiert, daß viele bekannte Innere-Punkte-Algorithmen der Linearen Optimierung ebenfalls in Algorithmen für Semidefinite Optimierung transformiert werden können. Seitdem haben viele Autoren Innere-Punkte-Algorithmen zum Lösen semidefiniter Probleme vorgeschlagen und implementiert, unter anderen Alizadeh, Haeberly und Overton [3], Helmberg, Rendl, Vanderbei und Wolkowicz [9], Jarre [11], Kojima, Shida und Shindoh [12], Kojima, Shindoh und Hara [13], Lin und Saigal [16, 15], Monteiro [18], Nesterov und Todd [24, 23], Potra und Sheng [26, 25, 28, 27], Sturm und Zhang [31], Vandenberghe und Boyd [35] und Zhang [40].
In dieser Arbeit werden einige Versionen von Innere-Punkte-Methoden beschrieben und verglichen. In Kapitel 1 werden zunächst die Grundlagen erörtert. In Kapitel 2 folgt dann die Beschreibung der Innere-Punkte-Methoden. Danach werden in Kapitel 3 verschiedene Suchrichtungen vorgestellt und verglichen. Es folgt eine Beschreibung der dieser Arbeit zugrundeliegenden Implementation in Kapitel 4. Das Kaptitel 5 besteht schließlich aus einer numerischen Auswertung der Implementation, gefolgt von einer Zusammenfassung in Kapitel 6.

Ich möchte mich an dieser Stelle bei Dr. Christoph Helmberg für seine ermunternde Unterstützung bedanken. Weiterhin danke ich Professor M. Grötschel und dem Konrad-Zuse Zentrum Berlin dafür, daß sie mir diese Arbeit ermöglicht haben und für die hervorragenden Arbeitsbedingungen. Für moralischen Anschub danke ich außerdem Professor G. Ziegler.

Kapitel 1

Grundlagen

In diesem Kapitel werden die Grundlagen für alle weiteren Kapitel besprochen. Dabei handelt es sich in Abschnitt 1.1 um Grundlagen aus der linearen Algebra. Es werden definite und semidefinite Matrizen definiert und charakterisiert. Außerdem wird festgestellt, daß die Menge der semidefiniten Matrizen ein abgeschlossener, spitzer und selbstdualer Kegel ist. Danach werden einige Eigenschaften des Kronecker-Produkts besprochen. Zuletzt folgt ein wenig über Differentiation von Matrizen.
In Abschnitt 1.2 folgen dann die Grundlagen der Semidefiniten Optimierung. Es wird eine Standardform für semidefinite Probleme und das duale Problem vorgestellt. Weiterhin wird eine hinreichende Bedingung für starke Dualität angegeben.

1.1 Grundlagen aus der Linearen Algebra

1.1.1 Symmetrische und positiv (semi-)definite Matrizen

Zuerst soll ein kurzer Überblick über die aus der linearen Algebra benötigten Hilfsmittel gegeben werden. Hierbei zitieren wir grundlegende Definitionen und Sätze aus Graham [8] und Horn und Johnson [10]. Es bezeichne $M_{m,n}$ die Menge der reellen $m \times n$ Matrizen. Die Menge $M_{m,n}$ kann als Vektorraum $\mathbf{R}^{m \cdot n}$ interpretiert werden. In diesem Vektorraum definieren wir das Innere Produkt zwischen zwei Elementen A und B wie folgt. Zuerst benötigen wir die Spur einer quadratischen Matrix.

Definition 1.1 (Spur) *[8] Die Spur* $\mathrm{tr}(.)$ *ist die Summe der Diagonalelemente einer quadratischen Matrix.*

$$\mathrm{tr}(A) := \sum_{i=1}^{n} a_{ii}, \quad A \in M_{n,n}$$

Definition 1.2 (Inneres Produkt) *[8] Für zwei Matrizen $A, B \in M_{m,n}$ ist das Innere Produkt zwischen ihnen*

$$\langle A, B \rangle := \mathrm{tr}(B^T A) = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} b_{ij}.$$

Definition 1.3 (Frobeniusnorm) *[10] Die Norm, die mit diesem Inneren Produkt assoziiert ist, ist die Frobeniusnorm*

$$\|A\|_F := \sqrt{\langle A, A \rangle}.$$

Anstatt der Matrizen aus $M_{m,n}$ werden bei der Semidefiniten Optimierung hauptsächlich symmetrische Matrizen verwendet. Die Menge der symmetrischen Matrizen soll mit S_n bezeichnet werden, welche wir als Vektorraum in $\mathbf{R}^{\binom{n+1}{2}}$ betrachten. Für S_n kann das gleiche innere Produkt genutzt werden. Also für $A, B \in S_n$ sei

$$\langle A, B \rangle = \mathrm{tr}(B^T A) = \mathrm{tr}(AB).$$

Eine weitere Klasse von Matrizen, auf die später noch eingegangen wird, sind die schiefsymmetrischen Matrizen. Eine schiefsymmetrische Matrix A erfüllt $A = -A^T$ und ist damit eine Art „Gegenstück“ einer symmetrischen Matrix. Die Diagonale der Matrix muß offensichtlich Null sein. Die Dimension des Vektorraums der schiefsymmetrischen Matrizen ist $\binom{n}{2}$. In der Menge M_n sind die linearen Halbräume der symmetrischen und der schiefsymmetrischen Matrizen orthogonal. Für $A \in S_n$ und eine beliebige schiefsymmetrische Matrix $B \in M_n$ gilt

$$\mathrm{tr}(AB) = \sum_{i<j} a_{ij} b_{ij} + \sum_{i>j} a_{ji}(-b_{ji}) = 0.$$

Zusammen spannen die symmetrischen und die schiefsymmetrischen Matrizen M_n auf. Eine Matrix $A \in M_n$ kann in ihren symmetrischen Teil $(A + A^T)/2$ und ihren schiefsymmetrischen Teil $(A - A^T)/2$ aufgespalten werden

$$A = \frac{A + A^T}{2} + \frac{A - A^T}{2}.$$

Wir befassen uns nun mit positiv semidefiniten Matrizen.

Definition 1.4 (Positive Definitheit) *[10]*
$A \in M_n$ ist positiv semidefinit $:\Leftrightarrow x^T A x \geq 0 \quad \forall x \in \mathbf{R}^n$.
Ist $A \in S_n$ schreiben wir auch kurz $A \in S_n^+$ oder $A \succeq 0$.
$A \in M_n$ ist positiv definit $:\Leftrightarrow x^T A x > 0 \quad \forall x \in \mathbf{R}^n \backslash \{0\}$.
Ist $A \in S_n$ schreiben wir auch kurz $A \in S_n^{++}$ oder $A \succ 0$.

Wir verwenden die positive Definitheit fast ausschließlich im Zusammenhang mit symmetrischen Matrizen. Im weiteren seien deshalb alle positiv definiten Matrizen symmetrisch, wenn nichts gegenteiliges geschrieben wird.
Es gibt einige äquivalente Charakterisierungen für positiv definite und semidefinite Matrizen.

Satz 1.5 (Charakterisierungen positiv definiter Matrizen) *[10]*
Für $A \in S_n$ sind die folgenden Aussagen äquivalent:

1. *A ist positiv definit.*

2. *Für alle Eigenwerte* λ_i $(i = 1, ..., n)$ *von* A *gilt* $\lambda_i > 0$.
3. $\exists C \in M_n$ *mit* $\operatorname{rang}(C) = n$, *so daß* $A = C^T C$.
4. *Für eine geschachtelte Sequenz* $\langle A_i \rangle_{i=1}^n$ *von Hauptuntermatrizen von* A *gilt* $\det(A_i) > 0$.

A ist genau dann positiv definit, wenn auch A^{-1} positiv definit ist, da die Eigenwerte von A^{-1} die Reziproken der Eigenwerte von A sind.

Satz 1.6 (Charakterisierungen positiv semidefiniter Matrizen) *[10]*
Für $A \in S_n$ *sind die folgenden Aussagen äquivalent:*

1. *A ist positiv semidefinit.*
2. *Für alle Eigenwerte* λ_i $(i = 1, ..., n)$ *von* A *gilt* $\lambda_i \geq 0$.
3. $\exists C \in M_{m,n}$ *mit* $\operatorname{rang}(C) = \operatorname{rang}(A)$, *so daß* $A = C^T C$.

Für positiv semidefinite Matrizen ist die Determinante jeder Hauptuntermatrix nicht negativ. Allerdings ist die Existenz einer geschachtelten Sequenz von Hauptuntermatrizen mit nichtnegativen Determinanten nicht mehr hinreichend für die positive Semidefinitheit. Um eine weitere hinreichende Bedingung für positive Definitheit zu erhalten, brauchen wir zunächst folgende Definition.

Definition 1.7 (Strikte diagonale Dominanz) *[10]*
$A \in M_n$ *is strikt diagonal dominant* $:\Longleftrightarrow |a_{ii}| > \sum_{j=1, i \neq j}^{n} |a_{ij}| \quad \forall i = 1, \ldots, n$.

Proposition 1.8 *[10] Ist* $A \in S_n$ *strikt diagonal dominant und sind alle Diagonalelemente positiv, so ist* A *positiv definit.*

Die diagonale Dominanz ist zum einen einfach zu überprüfen und zum anderen sehr nützlich um zulässige Startpunkte für semidefinite Probleme zu konstruieren. Es ist sehr bedeutend, daß eine positiv semidefinite Matrix A in der Form $C^T C$ geschrieben werden kann. Die Faktorisierung ist nicht eindeutig und kann auf unterschiedliche Arten berechnet werden. Die Cholesky Faktorisierung ist die wahrscheinlich bekannteste. Durch sie erhält man eine Dreiecksmatrix C. Ein anderer Ansatz basiert auf der Eigenwertzerlegung von A,

$$A = P \Lambda_A P^T = (P\sqrt{\Lambda_A})(P\sqrt{\Lambda_A})^T.$$

Eigenwertzerlegung wird auch für das folgende Theorem genutzt.

Theorem 1.9 (Wurzel einer positiv semidefiniten Matrix) *[10] Sei* $A \in S_n$ *positiv semidefinit und* $k \geq 1$. *Dann existiert eine eindeutige positiv semidefinite Matrix* B *mit* $B^k = A$. *Weiterhin erfüllt diese Matrix* B *die Gleichungen* $AB = BA$ *und* $\operatorname{rang}(A) = \operatorname{rang}(B)$.

Es folgt ein weiteres Theorem über positiv definite Matrizen.

Theorem 1.10 *[10] Sei* $A \in S_n^{++}$ *und* $C \in M_n$. *Dann ist* $C^T AC$ *positiv semidefinit. Ist* C *regulär ist* $C^T AC$ *sogar positiv definit.*

1.1.2 Kegel der semidefiniten Matrizen

In diesem Abschnitt soll die Menge der positiv semidefiniten Matrizen als Untermenge der symmetrischen Matrizen näher betrachtet werden.

Definition 1.11 (Kegel) *[37]*
Eine Menge C ist ein Kegel $:\Longleftrightarrow x, y \in C \Longrightarrow \alpha(x+y) \in C \quad \forall \alpha \geq 0$.
Ein Kegel C ist spitz $:\Longleftrightarrow C \cap (-C) = \{0\}$.

Kegel sind gemäß Definition konvexe Mengen. Ein Kegel C heißt abgeschlossen, wenn die Menge C abgeschlossen ist.
Auf der Grundlage dieser Definition läßt sich die folgende Behauptung zu der Menge der positiv semidefiniten Matrizen aufstellen.

Proposition 1.12 *[37]*
S_n^+ ist ein volldimensionaler, abgeschlossener spitzer Kegel im $\mathbf{R}^{\binom{n+1}{2}}$.

Die Menge der positiv definiten Matrizen ist kein Kegel, da $0 \notin S_n^{++}$. Es ist einfach zu ersehen, daß S_n^{++} das Innere des Kegels S_n^+ ist. Die positiv semidefiniten Matrizen, welche wenigstens einen Eigenwert besitzen, der Null ist, bilden den Rand von S_n^+.
Eine wichtige Eigenschaft des Kegels der semidefiniten Matrizen ist seine Selbstdualität. Dazu definieren wir zuerst den polaren (dualen) Kegel.

Definition 1.13 (Polarer Kegel) *Für einen Kegel C ist der polare Kegel C^* die Menge* $\{y : \langle x, y\rangle \geq 0 \quad \forall x \in C\}$.

Zuerst eine weitere Charakterisierung positiv definiter Matrizen.

Theorem 1.14 (Fejer's Spur Theorem) *[10]*
A ist positiv semidefinit $\Longleftrightarrow \langle A, B\rangle \geq 0 \quad \forall B \in S_n^+$.

Als Folgerung dieses Theorems erhalten wir die Selbstdualität.

Korollar 1.15 $S_n^+ = S_n^{+*}$.

Der Kegel der semidefiniten Matrizen induziert eine Halbordnung auf der Menge der symmetrischen Matrizen.

Definition 1.16 (Löwner Halbordnung) *[10]*
Für $A, B \in S_n$ gilt $A \succeq B$ $\quad :\Longleftrightarrow \quad (A-B) \in S_n^+$.
Für $A, B \in S_n$ gilt $A \succ B$ $\quad :\Longleftrightarrow \quad (A-B) \in S_n^{++}$.

Das ist der Ursprung der Notation $A \succeq 0$ $(A \succ 0)$ für $A \in S_n^+$ (S_n^{++}). Der folgende Satz gibt eine Charakterisierung für die positive Definitheit einer Matrix durch die positive Definitheit des Schur-Komplements hinsichtlich einer Blockaufteilung der Matrix.

Satz 1.17 (Schur-Komplement) *[10] Sei $A \in S_m$, $C \in S_n$ und $B \in M_{m,n}$. Dann gilt*

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \succ 0 \qquad \Longleftrightarrow \qquad A \succ 0 \quad \text{und} \quad C \succ B^T A^{-1} B.$$

1.1.3 Kronecker-Produkt

Das Kronecker-Produkt erleichtert die Handhabung von Matrixgleichungen. Für eine ausführlich Besprechung des Kronecker-Produkts siehe Graham [8]. Es ist eine Abbildung $\otimes : M_{m,n} \times M_{k,l} \to M_{mk,nl}$, welche definiert ist durch

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}.$$

Es ist aus der Definition zu ersehen, daß wenn A und B symmetrisch sind, auch $A \otimes B$ symmetrisch ist. In Verbindung mit dem Kronecker-Produkt benötigen wir einen Operator, der eine Matrix $A \in M_{m,n}$ in einen Vektor im $\mathbf{R}^{mn}$ transformiert. Dabei werden die Spalten von A aufeinander gestapelt:

$$\mathrm{vec(A)} = \begin{bmatrix} A_{\cdot,1} \\ \vdots \\ A_{\cdot,n} \end{bmatrix}.$$

Die im folgenden aufgeführten Eigenschaften des Kronecker-Produkts und des Vektoroperators werden uns noch von Nutzen sein.

Theorem 1.18 *[8] Es seien A, B, C und D Matrizen von entsprechender Größe. Wenn notwendig seien A und B invertierbar. Dann gilt*

$$\begin{aligned} (A \otimes B)^T &= (A^T \otimes B^T) && (1.1) \\ (A \otimes B)(C \otimes D) &= (AC) \otimes (BD) && (1.2) \\ \mathrm{vec(ABC)} &= (C^T \otimes A)\mathrm{vec(B)} && (1.3) \\ \mathrm{vec(AB + BC)} &= (I \otimes A + C^T \otimes I)\mathrm{vec(B)} && (1.4) \\ (A \otimes B)^{-1} &= A^{-1} \otimes B^{-1} && (1.5) \\ \mathrm{vec}(A)^T \mathrm{vec}(B) &= \langle A, B \rangle && (1.6) \end{aligned}$$

Alle Eigenschaften können durch einfaches Nachrechnen gezeigt werden. Sind die Eigenwerte und Eigenvektoren zweier Matrizen A und B bekannt, ist es einfach, die Eigenwerte und Eigenvektoren von $A \otimes B$ zu konstruieren.

Theorem 1.19 *[8]*
$A(B)$ habe die Eigenwerte $\lambda_i(\mu_i)$ mit zugehörigen Eigenvektoren $x_i(y_i) \in \mathbf{C}^n$. Dann sind die Eigenwerte von $A \otimes B$ gegeben durch $\lambda_i \mu_j$ mit Eigenvektoren $x_i \otimes y_j$.

Von diesem Theorem können wir zwei Folgerungen ableiten.

Korollar 1.20 *Sind A und B positiv definit, so ist auch $A \otimes B$ positiv definit.*

Korollar 1.21 *Sind A und B positiv semidefinit, so ist auch $A \otimes B$ positiv semidefinit.*

Analog wollen wir ein symmetrisiertes Kronecker-Produkt einführen. Wir folgen dabei der Darstellung von Todd, Toh und Tütüncü [32]. Zunächst benötigen wir einen speziellen Vektoroperator für symmetrische Matrizen.

Definition 1.22 *Sei $B \in S_n$, dann definieren wir den symmetrischen Vektoroperator* $\mathrm{svec}(B)$ *als*

$$\mathrm{svec}(B) := (b_{11}, \sqrt{2}b_{21}, \ldots, \sqrt{2}b_{n1}, b_{22}, \sqrt{2}b_{32}, \ldots, \sqrt{2}b_{n2}, \ldots, b_{nn})^T.$$

Der Faktor $\sqrt{2}$ wird eingeführt, um Isometrie bezüglich der Normen, induziert durch die inneren Produkte in S_n und $\mathbf{R}^{n(n+1)/2}$, zu bewahren. Das symmetrisierte Kronecker-Produkt zweier (nicht notwendig symmetrischer) Matrizen A und B, ist eine quadratische Matrix der Ordnung $n(n+1)/2$.

Definition 1.23 (Symmetrisiertes Kronecker-Produkt) *Die Wirkung des symmetrisierten Kronecker-Produktes auf einen Vektor $h := \mathrm{svec}(H)$ (mit $H \in S_n$) ist gegeben durch*

$$(A \otimes_s B)\mathrm{svec}(H) := \frac{1}{2}\mathrm{svec}(BHA^T + AHB^T).$$

Nun folgen, analog zum Kronecker-Produkt, einige Eigenschaften des symmetrisierten Kronecker-Produkts und des symmetrischen Vektoroperators.

Theorem 1.24 *[32] Es seien A, B, C und D Matrizen von entsprechender Größe. Wenn notwendig sei A invertierbar. Dann gilt*

$$\begin{aligned}
A \otimes_s B &= B \otimes_s A && (1.7)\\
(A \otimes_s B)^T &= (A^T \otimes_s B^T) && (1.8)\\
(A \otimes_s B)(C \otimes_s D) &= \frac{1}{2}(AC \otimes_s BD + AD \otimes_s BC) && (1.9)\\
(A \otimes_s B)(C \otimes_s C) &= (AC \otimes_s BC) && (1.10)\\
(C \otimes_s C)(A \otimes_s B) &= (CA \otimes_s CB) && (1.11)\\
(A \otimes_s A)^{-1} &= A^{-1} \otimes_s A^{-1} && (1.12)\\
\mathrm{svec}(A)^T\mathrm{svec}(B) &= \langle A, B\rangle && (1.13)
\end{aligned}$$

Aus der Gleichung (1.8) folgt, daß $A \otimes_s B$ symmetrisch ist, wenn A und B symmetrisch sind.

Theorem 1.25 *[32] Die Matrix A habe die Eigenwerte $\lambda_i (i = 1, \ldots, n)$ mit zugehörigen Eigenvektoren x_i. Dann sind $\lambda_i\lambda_j (i, j = 1, \ldots, n)$ die Eigenwerte von $A \otimes_s A$ mit den zugehörigen Eigenvektoren* $\mathrm{svec}(\frac{1}{2}(x_i x_j^T + x_j x_i^T))$.

Theorem 1.26 *[32] Es seien A und B zwei kommutierende Matrizen mit einer gemeinsamen Basis von Eigenvektoren $\{x_i\}$ und zugehörigen Eigenwerten λ_i und μ_i. Dann hat $A \otimes B$ die Eigenwerte $\frac{1}{2}(\lambda_i\mu_j + \lambda_j\mu_i)$ mit zugehörigen Eigenvektoren* $\mathrm{svec}(\frac{1}{2}(x_i x_j^T + x_j x_i^T))$.

Theorem 1.27 *[32] Es seien A und B zwei symmetrische positiv definite Matrizen. Dann ist $A \otimes_s B$ symmetrisch positiv definit.*

Wie Todd, Toh und Tütüncü [32] gezeigt haben, können die Eigenschaften des symmetrisierten Kronecker-Produkts verifiziert werden, indem man es zum „normalen“ Kronecker-Produkt und seinen Eigenschaften in Beziehung setzt. Um dies zu tun, betrachten wir die $n(n+1)/2 \times n^2$ dimensionale Matrix U für die gilt, $U\mathrm{vec}(H) = \mathrm{svec}(H)$ und $U^T\mathrm{svec}(H) = \mathrm{vec}(H)$ für alle symmetrischen $n \times n$ Matrizen H.
Wenn wir die Reihen von U mit (1,1),(2,1),...,(n,1),(2,2),(3,2),...,(n,2),(3,3),...,(n,n) und die Spalten von U mit (1,1),(2,1),...,(n,1),(1,2),...,(n,2),(1,3),...,(n,n) beschriften, dann gilt:

$$U_{(i,j),(k,l)} = \begin{cases} 1 & \text{wenn } i = j = k = l, \\ 1/\sqrt{2} & \text{wenn } i = k \neq j = l \text{ oder } i = l \neq j = k, \\ 0 & \text{sonst.} \end{cases}$$

Zum Beispiel gilt für $n = 3$

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/\sqrt{2} & 0 & 0 & 0 & 1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Man prüft leicht nach, daß $UU^T = I_{n(n+1)/2}$. Und obwohl $U^TU \neq I_{n^2}$ ist, erhalten wir $U^TU\mathrm{vec}(H) = \mathrm{vec}(H)$ für jede Matrix $H \in S_n$. Mit der Matrix U kann $A \otimes_s B$ wie folgt mit Hilfe des Kronecker-Produkts ausgedrückt werden:

$$A \otimes_s B = \frac{1}{2}U(A \otimes B + B \otimes A)U^T.$$

1.1.4 Matrix Differentialrechnung

Wir werden nun Matrixfunktionen, von Matrizen abhängige Funktionen und deren Ableitungen betrachten. Für näheres zu diesem Unterabschnitt sei auf das Buch von Graham [8] verwiesen. Wir benutzen den vec-Operator um Ableitungen nach Matrixvariablen in Ableitungen nach Vektorvariablen umzuformen. Es sei x ein Vektor im $\mathbf{R}^n$ und sei $F : \mathbf{R}^n \to \mathbf{R}^m; x \mapsto [f_1(x), \ldots, f_n(x)]^T$ eine differenzierbare Funktion. Dann ist der Gradient (oder die Ableitung) von $F(x)$ nach x die $n \times m$ Funktion

$$\nabla_x F = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \cdots & \frac{\partial f_m(x)}{\partial x_1} \\ \vdots & & \vdots \\ \frac{\partial f_1(x)}{\partial x_n} & \cdots & \frac{\partial f_m(x)}{\partial x_n} \end{bmatrix}.$$

Die Linearisierung von F in x^0 ist

$$F(x^0 + \Delta x) \approx F(x^0) + \left[\nabla_x F(x^0)\right]^T \Delta x.$$

Satz 1.28 (Eigenschaften der Ableitung)

$$\begin{aligned} \nabla_x(Ax) &= A^T && (1.14) \\ \nabla_x z(y(x)) &= \nabla_x y \cdot \nabla_y z \qquad \textit{(Kettenregel)} && (1.15) \end{aligned}$$

Wir illustrieren die Anwendung von (1.14) auf Matrizen für die Produkte AX und XA mit $A, X \in S_n$. Für AX haben wir nach (1.3) $\mathrm{vec(AX)} = (\mathrm{I} \otimes \mathrm{A})\mathrm{vec(X)}$. Damit ist

$$\nabla_X \mathrm{vec(AX)} = (\mathrm{I} \otimes \mathrm{A}) \quad \text{und analog} \quad \nabla_X \mathrm{vec(XA)} = (\mathrm{A} \otimes \mathrm{I}).$$

Um annehmbare Representierungen für die Linearisierungen zu erhalten, ist zu beachten, daß $(I \otimes A)\mathrm{vec}(\Delta \mathrm{X}) = \mathrm{vec}(\mathrm{A}\Delta\mathrm{X})$ und daher

$$[\nabla_X \mathrm{vec(AX)}]^T \mathrm{vec}(\Delta\mathrm{X}) = \mathrm{vec}(\mathrm{A}\Delta\mathrm{X})$$

und

$$[\nabla_X \mathrm{vec(XA)}]^T \mathrm{vec}(\Delta\mathrm{X}) = \mathrm{vec}(\Delta\mathrm{XA}).$$

Wir schreiben intuitiv $A\Delta X$ und ΔXA für die Linearisierungen von AX und XA.
In den Innere-Punkte-Verfahren von Kapitel2 werden wir die Funktion

$$-\log\det(X) = -\log\prod_{i=1}^{n}\lambda_i(X) = -\sum_{i=1}^{n}\log\lambda_i(X)$$

als Barriere-Funktion benötigen. Da $\det(X)$ eine kontinuierliche Funktion ist, welche positiv für positiv definite Matrizen und Null für singuläre semidefinite Matrizen ist, wächst, für eine Folge $\langle X_n \rangle$ von Matrizen aus S_n^{++}, die sich dem Rand von S_n^+ nähert, die Barriere-Funktion nach Unendlich. Weiterhin ist $-\log\det(X)$ streng konvex.

Lemma 1.29 *[10]* $\log\det(X)$ *ist streng konkav auf der Menge der positiv definiten Matrizen.*

Um die Optimalitäts-Bedingung erster Ordnung für das Barriere-Problem in Abschnitt 2.1 aufzustellen, brauchen wir die Ableitung von $\log\det X$. Für skalare Matrixfunktionen ist es gebräuchlich, den Gradienten in Form einer Matrix zu repräsentieren.

Theorem 1.30 *[8] Für eine nichtsinguläre Matrix* $X \in M_n$ *gilt*

$$\nabla_X \det(X) = X^{-T} \cdot \det(X).$$

Wir erhalten die Ableitung von $\log\det(X)$ für positiv definite Matrizen X durch die Kettenregel

$$\nabla_X \log\det(X) = \frac{1}{|\det(X)|}\nabla_X \det(X) = X^{-1}.$$

Im Prinzip ist X^{-1} ein Gradientenvektor und keine Matrix. Sie sollte daher in der Form $\langle X^{-1}, \Delta X \rangle = \mathrm{tr}(X^{-1}\Delta X)$ benutzt werden.

1.2 Grundlagen der Semidefiniten Programmierung

In diesem Abschnitt werden die Grundlagen der Semidefiniten Programmierung geklärt. Es wird eine Standardformulierung für ein semidefinites Problem eingeführt, und es werden Fragen der Dualität und Komplexität besprochen.
Man kann semidefinite Probleme in unterschiedlicher Form darstellen. Hier soll die folgende Standardform benutzt werden:

$$\begin{array}{ll} \max & \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}(X) = b \\ & X \succeq 0. \end{array} \tag{1.16}$$

Hierbei sind $C, X \in S_n$, $b \in \mathbf{R}^m$ und $\mathcal{A}(.) : S_n \to \mathbf{R}^m$ ist der lineare Operator

$$\mathcal{A}(X) = \begin{bmatrix} \langle A_1, X\rangle \\ \vdots \\ \langle A_m, X\rangle \end{bmatrix}, \quad A_i \in S_n \quad (i = 1, \ldots, m). \tag{1.17}$$

Die A_i seien ohne Beschränkung der Allgemeinheit linear unabhängig. Werden die Bedingungen A_i so gewählt, daß X als diagonal erzwungen wird, so erhält man Lineare Optimierung als Spezialfall der Semidefiniten Optimierung.
Es gibt eine ausführliche Dualitätstheorie für die Semidefinite Optimierung, annähernd analog zu der der Linearen Optimierung (siehe z.B. Wolkowicz [37] oder Alizadeh [2]). Um das duale Problem von (1.16) zu erhalten, benötigen wir zuerst den zu $\mathcal{A}(.)$ adjungierten Operator $\mathcal{A}^T(.) : \mathbf{R}^m \to S_n$,

$$\mathcal{A}^T(y) = \sum_{i=1}^{m} y_i A_i. \tag{1.18}$$

Das duale Problem ist dann

$$\begin{array}{ll} \min & b^T y \\ \text{s.t.} & Z = \mathcal{A}^T(y) - C \succeq 0 \\ & y \in \mathbf{R}^m, \end{array} \tag{1.19}$$

wobei $Z \in S_n$ die duale Schlupfvariable ist. Wenn X primal und y, Z dual zulässige Lösungen sind, folgt

$$\langle C, X\rangle = \langle \mathcal{A}^T(y), X\rangle - \langle Z, X\rangle = \langle y, \mathcal{A}(X)\rangle - \langle Z, X\rangle = b^T y - \langle Z, X\rangle \leq b^T y.$$

Das heißt, für zwei primal bzw. dual zulässige Lösungen ist der primale Zielfunktionswert kleiner oder gleich dem dualen Zielfunktionswert. Dieses wird auch schwache Dualität genannt. Damit ist $\langle Z, X\rangle = \operatorname{tr}(XZ)$ die Differenz der primalen und dualen Lösung. Wenn entweder das primale oder das duale Problem unbeschränkt ist, ist das jeweils andere offensichtlich unzulässig. Ist $\langle Z, X\rangle$ gleich Null, dann ist dieses primal-duale Paar eine optimale Lösung. Umgekehrt ist dies, im Gegensatz zu linearer Optimierung, nicht der Fall.

Beispiel 1.31 *[36] Betrachten wir das folgende primale semidefinite Problem*

$$\begin{array}{ll} \max & -x_{12} \\ \text{s.t.} & \begin{bmatrix} 0 & x_{12} & 0 \\ x_{12} & x_{22} & 0 \\ 0 & 0 & 1+x_{12} \end{bmatrix} \succeq 0. \end{array}$$

Um das duale Problem zu erhalten, schreiben wir das Problem zunächst in Standardform auf,

$$\begin{array}{lll} \max & \left\langle \begin{bmatrix} 0 & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, X \right\rangle & \\ \text{s.t.} & \left\langle \begin{bmatrix} 0 & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, X \right\rangle & = 1, \\ & \left\langle \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, X \right\rangle & = 0, \\ & \left\langle \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, X \right\rangle & = 0, \\ & \left\langle \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, X \right\rangle & = 0, \\ & X \succeq 0. & \end{array}$$

Nennen wir die Matrizen in absteigender Reihenfolge $A_1, \ldots, A_4$ und sind $y_1, \ldots, y_4$ die zugehörigen dualen Variablen, ergibt sich für das duale Problem

$$\min y_1 \quad \text{s.t. } Z = y_1 A_1 + y_2 A_2 + y_3 A_3 + y_4 A_4 - C \succeq 0.$$

Dieses Problem läßt sich umschreiben in

$$\begin{array}{ll} \min & y_1 \\ \text{s.t.} & Z = \begin{bmatrix} y_2 & \frac{1-y_1}{2} & y_3 \\ \frac{1-y_1}{2} & 0 & y_4 \\ y_3 & y_4 & y_1 \end{bmatrix} \succeq 0. \end{array}$$

Eine notwendige Bedingung für die positive Semidefinitheit der primalen Matrix ist, daß x_{12} Null ist, weil $x_{11} = 0$. Ebenso erhalten wir für das duale Problem aus $z_{22} = 0$, daß $z_{12} = 0$ und damit $y_1 = 1$. Es ist leicht zu sehen, daß es wirklich Lösungen mit diesen Werten gibt. Die Lücke zwischen jedem Paar primaler und dualer Optimallösungen ist Eins.

Starke Dualität (d.h., wenn das eine Problem eine endliche Optimallösung besitzt, hat das andere ebenfalls eine endliche Optimallösung mit dem gleichen Zielfunktionswert) erhalten wir unter der zusätzlichen (hinreichenden aber nicht notwendigen) Annahme, daß entweder (1.16) oder (1.19) einen strikt zulässigen Punkt besitzt.

Definition 1.32 (Strikte Zulässigkeit)
Ein Punkt X ist strikt zulässig für (1.16), wenn er für (1.16) zulässig ist und $X \succ 0$ erfüllt. Ein Paar (y, Z) ist strikt zulässig für (1.19), wenn es für (1.19) zulässig ist und $Z \succ 0$ erfüllt.

Theorem 1.33 (Starke Dualität) *[22, 36]*
Es seien

$$p^* \;=\; \sup\{\langle C, X\rangle : \mathcal{A}(X) = b, X \succeq 0\}$$

und

$$d^* \;=\; \inf\{\langle b, y\rangle : \mathcal{A}^T(y) - Z = C, Z \succeq 0\}.$$

(i) Ist (1.16) strikt zulässig, dann ist $p^* = d^*$ und dieser Wert wird für (1.19) angenommen,falls p^* endlich ist.

(ii) Ist (1.19) strikt zulässig, dann ist $p^* = d^*$ und dieser Wert wird für (1.16) angenommen, falls d^* endlich ist.

(iii) Ist sowohl (1.16) als auch (1.19) strikt zulässig, dann ist $p^* = d^*$ und wird für beide Probleme angenommen.

Gibt es keine dual strikt zulässige Lösung kann es passieren, daß die primale Optimallösung nicht angenommen wird.

Beispiel 1.34 *Betrachten wir folgendes Problem*

$$\begin{array}{ll} \max & -x_{11} \\ \text{s.t.} & \begin{bmatrix} x_{11} & 1 \\ 1 & x_{22} \end{bmatrix} \succeq 0 \end{array} \quad \begin{array}{ll} \min & -2y_1 \\ \text{s.t.} & \begin{bmatrix} 1 & -y_1 \\ -y_1 & 0 \end{bmatrix} \succeq 0. \end{array}$$

Das primale Problem hat die strikt zulässige Lösung $(x_{11} = 2, x_{22} = 2)$ und tatsächlich wird die duale Optimallösung 0 für $y_1 = 0$ angenommen. Diese ist zusätzlich die einzig dual zulässige Lösung. Durch die strikte Zulässigkeit des primalen Problems wissen wir, daß das Supremum der Zielfunktionswerte aller primal zulässigen Lösungen Null sein muß. Um zu testen, ob dieser Wert auch angenommen wird, berechnen wir die Menge der zulässigen Lösungen explizit. Wegen der Semidefinitheitsbedingung müssen die Determinanten der Hauptuntermatrizen nichtnegativ sein,

$$x_{11} \geq 0, \qquad x_{22} \geq 0, \qquad x_{11}x_{22} - 1 \geq 0.$$

Dies ergibt die untere Grenze $x_{11} \geq \frac{1}{x_{22}}$, die für $x_{22} \to \infty$ Null ist. Der primale Optimalwert wird also nicht angenommen. Es ist zu beachten, daß das duale Problem keine strikt zulässige Lösung hat.

Kapitel 2

Innere-Punkte-Verfahren

In diesem Kapitel werden die Grundlagen Innerer-Punkte Verfahren zum Lösen semidefiniter Probleme besprochen. In Abschnitt 2.1 werden die Eigenschaften des Zentralen Pfads diskutiert und das Grundgerüst für Innere-Punkte Algorithmen besprochen. Im Gegensatz zur Linearen Optimierung ist der Grenzwert des zentralen Pfads bei der semidefiniten Optimierung nicht unbedingt ein strikt komplementäres Paar optimaler Lösungen, da solch eine Lösung nicht immer existiert. Aber wir werden feststellen, daß der Grenzwert des Zentralen Pfads immer eine „maximal komplementäre“ Lösung ist. Die im Innere-Punkte-Verfahren verwendete perturbierte Komplementaritätsbedingung eines semidefiniten Problems lautet $XZ = \nu I$. Da X und Z Matrizen sind, gilt im allgemeinen $XZ \neq ZX$; daher muß bei der Linearisierung auf Symmetrie geachtet werden. Im nächsten Abschnitt wird der Prediktor-Korrektor Ansatz besprochen. Im dritten Abschnitt dieses Kapitels werden wir eine schiefsymmetrische Einbettung für semidefinite Probleme betrachten. Diese Einbettung garantiert einen zulässigen zentrierten Startpunkt. Aus der optimalen Lösung des eingebetteten Problems läßt sich die optimale Lösung des Originalproblems ableiten, sofern sowohl das primale als auch das duale Problem zulässig ist und es keine Dualitätslücke gibt. Hat eines der Probleme einen verbessernden Strahl und ist daher in Richtung der Zielfunktion unbeschränkt, wird dies entdeckt. Die Einbettung bringt allerdings keine nützliche Information, wenn es eine Dualitätslücke gibt oder die Optimallösung für eines der Probleme nicht angenommen wird.

2.1 Barriere Problem und Zentraler Pfad

Während dieses Abschnitts soll folgende Annahme gelten.

Annahme 2.1 *Es existiert ein strikt zulässiges X^0 für (1.16) und ein strikt zulässiges Paar (y^0, Z^0) für (1.19).*

In Abschnitt 2.5 werden wir sehen, daß dies mit Hilfe der schiefsymmetrischen Einbettung für vernünftig gestellte Probleme immer zu erreichen ist. Wie in der linearen Optimierung beginnen wir mit dem Barriere-Problem. Für den Kegel der positiv semidefiniten Matrizen ist $-\log\det(\cdot)$ die kanonische Barriere-Funktion. Das Barriere-Problem lautet

$$\begin{array}{ll} \max & \langle C, X\rangle + \nu \log\det(X) \\ \text{s.t.} & \mathcal{A}(X) = b \\ & X \succ 0. \end{array}$$

ν ist der Barriere-Parameter und kontrolliert den Einfluß der Barriere-Funktion. Die Kostenfunktion ist nach Lemma 1.29 streng konkav. Zusammen mit Annahme 2.1 folgt daraus die Existenz und Eindeutigkeit der Optimallösung. Um ein unrestringiertes Problem zu erhalten, definieren wir für jedes $\nu > 0$ die zugehörige Langrange-Funktion

$$\mathcal{L}_\nu(X, y) := \langle C, X\rangle + \nu \log\det(X) + \langle y, b - \mathcal{A}(X)\rangle.$$

Der Sattelpunkt dieser Sattelfunktion liefert die Optimallösung des Barriere-Problems (siehe Rockafellar [29]). Die notwendigen Bedingungen erster Ordnung für diesen Sattelpunkt, auch Karush-Kuhn-Tucker-Bedingungen genannt, sind

$$\begin{array}{rcl} \nabla_X \mathcal{L}_\nu &=& C + \nu X^{-1} - \mathcal{A}^T(y) = 0 \\ \nabla_y \mathcal{L}_\nu &=& b - \mathcal{A}(X) = 0. \end{array}$$

In einer primal-dualen Formulierung setzen wir $Z = \nu X^{-1}$ und schreiben die KKT-Bedingungen folgendermaßen um:

$$\begin{array}{rcl} \mathcal{A}(X) &=& b, \quad X \succ 0 \\ \mathcal{A}^T(y) - Z &=& C, \quad Z \succ 0 \\ XZ &=& \nu I. \end{array} \tag{2.1}$$

Die erste Zeile beschreibt die primale Zulässigkeit, die zweite Zeile beschreibt die duale Zulässigkeit und die dritte Zeile ist eine perturbierte Komplementaritätsbedingung. Die Lösung dieses Systems ist eindeutig in X und Z aber nicht notwendiger Weise in y. Wir nennen die Lösung von (2.1) für ein beliebiges aber festes ν (X_ν, Z_ν). X_ν ist die eindeutige Optimallösung des primalen Barriere-Problems und Z_ν ist die eindeutige Optimallösung des analogen dualen Barriere-Problems. X_ν und Z_ν sind primal bzw. dual zulässige Punkte des Originalproblems mit einer Differenz der Zielfunktionswerte von $\langle Z_\nu, X_\nu\rangle = n\nu$. Die Menge der Lösungen (X_ν, Z_ν) mit $\nu > 0$ ist eine glatte Kurve, welche zentraler Pfad genannt wird [13].

Lemma 2.2 *[6] Für eine Folge $\langle \nu_k\rangle_{k\in\mathbf{N}} \to 0, \nu_k > 0\ \forall k \in \mathbf{N}$ konvergieren die korrespondierenden Lösungen (X_{ν_k}, Z_{ν_k}) gegen ein Paar optimaler Lösungen (X^*, Z^*) von (1.16) und (1.19).*

Definition 2.3 (Maximale und Strikte Komplementarität) *[6]*
Ein Paar optimaler Lösungen (X^*, Z^*) ist maximal komplementär, wenn X^* und Z^* unter allen optimalen Lösungen maximalen Rang haben.
Ein optimales Paar (X^*, Z^*) ist strikt komplementär, wenn
$\operatorname{rang}(X^*) + \operatorname{rang}(Z^*) = n$.

Lemma 2.4 *[6] Für eine Folge $\langle \nu_k \rangle_{k\in\mathbf{N}} \to 0, \nu_k > 0 \ \forall k \in \mathbf{N}$ konvergieren die korrespondierenden Lösungen (X_{ν_k}, Z_{ν_k}) gegen ein Paar maximal komplementärer, optimaler Lösungen (X^*, Z^*) von (1.16) und (1.19).*

Angenommen, wir versuchen, ausgehend vom Punkt (X, y, Z), den Punkt $(X_{\sigma\mu}, y_{\sigma\mu}, Z_{\sigma\mu})$ auf dem zentralen Pfad für den Wert $\nu = \sigma\mu$ mittels eines Schrittes $(\Delta X, \Delta y, \Delta Z)$ zu ereichen, wobei $\mu = \langle X, Z\rangle/n$ und $\sigma \in [0, 1]$, dann haben wir das System

$$\begin{array}{rcl} \mathcal{A}(\Delta X) & = & -(\mathcal{A}(X) - b) \\ \mathcal{A}^T(\Delta y) - \Delta Z & = & -(\mathcal{A}^T(y) - Z - C) \\ \Delta X Z + X \Delta Z & = & \sigma\mu I - XZ - \Delta X \Delta Z \end{array} \tag{2.2}$$

zu lösen. Linearisieren wir, wie im Newton-Verfahren, die dritte Gleichung, ergibt sich

$$\begin{array}{rcl} \mathcal{A}(\Delta X) & = & -(\mathcal{A}(X) - b) \\ \mathcal{A}^T(\Delta y) - \Delta Z & = & -(\mathcal{A}^T(y) - Z - C) \\ \Delta X Z + X \Delta Z & = & \sigma\mu I - XZ. \end{array} \tag{2.3}$$

Allerdings kommutieren X und Z in der Linearisierung der Komplementaritäts-Bedingung nicht ($XZ \neq ZX$). Im Allgemeinen können wir nicht erwarten, daß symmetrische ΔX und ΔZ existieren, die (2.3) lösen. Wird das System mit quadratischen Matrizen $\Delta X, \Delta Z \in M_n$ gelöst, ergibt sich wegen der zweiten Gleichung ein symmetrisches ΔZ aber (im Allgemeinen) ein unsymmetrisches ΔX. Da die nächste Iterierte $X + \alpha\Delta X$, wobei α die Schrittweite ist, symmetrisch positiv definit sein muß, ist dies ein erhebliches Problem. Es wurden verschiedene Ansätze vorgeschlagen, um diese Schwierigkeit zu überwinden. Daraus ergaben sich verschiedene Suchrichtungen. Alle in dieser Arbeit besprochenen Ansätze lassen sich durch die Einführung eines Symmetrisierungsoperators

$$H_P(M) := \frac{1}{2}(PMP^{-1} + (PMP^{-1})^T) = (P \otimes_s P^{-T})$$

mit einer nichtsingulären Matrix $P \in S_n$, verallgemeinern, wie Zhang [40] gezeigt hat. Zhang hat auch beobachtet, daß

$$H_P(M) = \tau I \Longleftrightarrow M = \tau I.$$

Durch das Symmetrisieren der Komplementaritätsbedingung erhalten wir

$$H_P(XZ + \Delta X Z + X \Delta Z) = \sigma\mu I. \tag{2.4}$$

Damit ergibt sich das verallgemeinerte System

$$\begin{array}{rcl} \mathcal{A}(\Delta X) & = & -(\mathcal{A}(X) - b) \\ \mathcal{A}^T(\Delta y) - \Delta Z & = & -(\mathcal{A}^T(y) - Z - C) \\ H_P(XZ + \Delta X Z + X \Delta Z) & = & \sigma\mu I. \end{array} \tag{2.5}$$

Wählt man $P = I$ so erhält man die in [3] vorgestellte AHO-Richtung. Für $P = Z^{\frac{1}{2}}$ ist die Suchrichtung äquivalent mit der HKM-Richtung aus [9, 13, 18]. Die NT-Richtung aus [24] ergibt sich mit der Wahl von P, so daß $P^T P = W = X^{\frac{1}{2}}(X^{\frac{1}{2}} Z X^{\frac{1}{2}})^{-\frac{1}{2}} X^{\frac{1}{2}}$.
Diese drei Suchrichtungen sind die zur Zeit in praktischen Implementationen populärsten. Die algorithmische Grundstruktur ist für alle diese Methoden die Gleiche.

Algorithmus 2.5 *Input:* $\mathcal{A}, b, C$ *und ein Startpunkt* (X^0, y^0, Z^0) *mit* $X^0 \succ 0$ *und* $Z^0 \succ 0$.

1. *Wähle* $\mu = \frac{\langle X,Z \rangle}{n}$ *und* $\sigma \in [0,1]$.
2. *Berechne* $(\Delta X, \Delta y, \Delta Z)$ *durch lösen einer Version von (2.5).*
3. *Wähle ein* α_p *und ein* α_d*, so daß* $X + \alpha_p \Delta X$ *und* $Z + \alpha_d \Delta Z$ *positiv definit bleiben.*
4. *Setze* $(X, y, Z) := (X + \alpha_p \Delta X, y + \alpha_d \Delta y, Z + \alpha_d \Delta Z)$.
5. *Stoppe, wenn primale Zulässigkeit, duale Zulässigkeit und* $\langle X, Z \rangle$ *klein genug sind, sonst gehe zu 1.*

Es wurde in verschiedenen Quellen (z.B. [21, 20, 19]) bewiesen, daß bei geeigneter Wahl von σ, α_p und α_d eine polynomielle Iterationskomplexität erreicht werden kann.
So benötigt zum Beispiel die primal-duale zulässige Innere-Punkte-Methode aus [19] zum erreichen einer Genauigkeit von $\langle X, Z \rangle \leq \varepsilon$ $O(\sqrt{n} \log \frac{\langle X_0, Z_0 \rangle}{\varepsilon})$ Iterationen.

2.2 Kronecker-Blockstruktur

Um das zu lösende System etwas übersichtlicher zu gestalten, nutzen wir das Kronecker-Produkt und den Vektor-Operator aus Abschnitt 1.1.3. Dabei folgen wir den Darstellungen von Alizadeh et al. [3] und Todd et al. [32]. Zuerst müssen wir den Operator $\mathcal{A}$ als Matrix uminterpretieren.

$$\mathbf{A} = \begin{bmatrix} \operatorname{vec}(A_1)^T \\ \vdots \\ \operatorname{vec}(A_m)^T \end{bmatrix}, \quad A_i \in S_n \quad (i = 1, \ldots, m)$$

Da die A_i linear unabhängig sind, hat $\mathbf{A}$ vollen Zeilenrang. Nun führen wir noch die Matrizen $E := \frac{1}{2}(P^{-T} Z \otimes P + P \otimes P^{-T} Z)$, $F := \frac{1}{2}(PX \otimes P^{-T} + P^{-T} \otimes PX)$ und $R := H_P(XZ) - \sigma\mu I$ ein. Damit erhalten wir das System (2.5) in Blockstruktur

$$\begin{bmatrix} 0 & \mathbf{A}^T & -\mathbf{I} \\ \mathbf{A} & 0 & 0 \\ E & 0 & F \end{bmatrix} \begin{bmatrix} \operatorname{vec}(\Delta X) \\ \Delta y \\ \operatorname{vec}(\Delta Z) \end{bmatrix} = \begin{bmatrix} \operatorname{vec}(C) - \mathbf{A}^T y + \operatorname{vec}(Z) \\ b - \mathbf{A}\operatorname{vec}(X) \\ \operatorname{vec}(-R) \end{bmatrix}. \tag{2.6}$$

$\mathbf{I}$ ist hierbei die Identitätsmatrix der Dimension $n^2 \times n^2$.
Die Matrizen E und F sind nicht singulär, wenn X und Z positiv definit sind. Dies folgt aus den Gleichungen (siehe (1.2))

$$E = \frac{1}{2}(P^{-T}ZP^{-1} \otimes I + I \otimes P^{-T}ZP^{-1})(P \otimes P) \tag{2.7}$$

und

$$F = \frac{1}{2}(PXP^T \otimes I + I \otimes PXP^T)(P^{-T} \otimes P^{-T}). \tag{2.8}$$

Jeweils sind beide Faktoren nicht singulär (Theorem 1.19 und (1.5)). Der erste ist nach Korollar 1.20 sogar positiv definit. Nun kann das System (2.6) sehr übersichtlich mit Hilfe von Gauss-Block-Elimination gelöst werden. Indem wir den dritten Block mit F^{-1} multiplizieren und zum ersten Block addieren, erhalten wir

$$\begin{bmatrix} F^{-1}E & \mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix} \begin{bmatrix} \mathrm{vec}(\Delta X) \\ \Delta y \end{bmatrix} = \begin{bmatrix} \mathrm{vec}(C) - \mathbf{A}^T y + \mathrm{vec}(Z) - F^{-1}\mathrm{vec}(R) \\ b - \mathbf{A}\mathrm{vec}(X) \end{bmatrix}. \tag{2.9}$$

Im zweiten Schritt der Gauss-Block-Elimination multiplizieren wir die erste Zeile mit $\mathbf{A}E^{-1}F$ und subtrahieren sie von der zweiten. Dies ergibt die Schur-Komplement-Gleichung

$$-(\mathbf{A}E^{-1}F\mathbf{A}^T)\Delta y = b - \mathbf{A}\mathrm{vec}(X) - \mathbf{A}E^{-1}F(\mathrm{vec}(C) - \mathbf{A}^T y + \mathrm{vec}(Z)) + \mathbf{A}E^{-1}\mathrm{vec}(R).$$

Indem wir diese Gleichung mit -1 multiplizieren, erhalten wir unser endgültiges System

$$(\mathbf{A}E^{-1}F\mathbf{A}^T)\Delta y = -b + \mathbf{A}(\mathrm{vec}(X) + E^{-1}F(\mathrm{vec}(C) - \mathbf{A}^T y + \mathrm{vec}(Z)) - E^{-1}\mathrm{vec}(R)). \tag{2.10}$$

Wie dieses System für unterschiedliche Wahlen von P günstig aufgestellt wird, beschreibt das nächste Kapitel über die verschiedenen Suchrichtungen. Es ist ein Gleichungssystem von m Gleichungen in den m Variablen Δy. Ist $m = O(n)$, wie es in der Praxis oft der Fall ist, ist es weniger kostspielig (2.10) aufzustellen und zu lösen, als (2.6) oder (2.9) direkt zu lösen [3].
Ist Δy ermittelt, ergibt sich ΔZ durch die erste Block-Gleichung in (2.6)

$$\Delta Z = \mathcal{A}^T(y + \Delta y) - C - Z.$$

Nun setzen wir ΔZ in die dritte Block-Gleichung ein und erhalten

$$\mathrm{vec}(\Delta X) = -E^{-1}F\mathrm{vec}(\Delta Z) - E^{-1}\mathrm{vec}(R).$$

Mit dem ebenfalls in Abschnitt 1.1.3 eingeführten symmetrisierten Kronecker-Produkt, läßt sich das System (2.5) mit

$$E_s := P \otimes_s P^{-T}Z, \quad F_s := PX \otimes_s P^{-T},$$

$$\begin{array}{rcl} R_d & := & \mathrm{svec}(C) - \mathbf{A}_s^T y + \mathrm{svec}(Z), \\ r_p & := & b - \mathbf{A}_s \mathrm{svec}(X), \\ R_c & := & \mathrm{svec}(\sigma\mu I - H_P(XZ)), \\ \mathbf{A}_s^T & := & [\mathrm{svec}(A_1) \cdots \mathrm{svec}(A_m)]\,. \end{array}$$

auch wie folgt formulieren:

$$\begin{bmatrix} 0 & \mathbf{A}_s^T & -\mathcal{I} \\ \mathbf{A}_s & 0 & 0 \\ E_s & 0 & F_s \end{bmatrix} \begin{bmatrix} \mathrm{svec}(\Delta X) \\ \Delta y \\ \mathrm{svec}(\Delta Z) \end{bmatrix} = \begin{bmatrix} R_d \\ r_p \\ R_c \end{bmatrix}. \tag{2.11}$$

Hierbei ist $\mathcal{I}$ die Identitätsmatrix der Ordnung $n(n+1)/2$. Auch hier gilt, daß die Matrizen E_s und F_s nicht singulär sind, wenn X und Z positiv definit sind. Aus der Eigenschaft (1.10) des symmetrisierten Kronecker-Produkts folgen die Gleichungen

$$E_s = P \otimes_s P^{-T} Z = (I \otimes_s P^{-T} Z P^{-1})(P \otimes_s P) \tag{2.12}$$

und

$$F_s = PX \otimes_s P^{-T} = (PXP^T \otimes_s I)(P^{-T} \otimes_s P^{-T}). \tag{2.13}$$

Nun sind E_s und F_s nicht singulär, da beide Faktoren in (2.12) und (2.13) nicht singulär sind (siehe Eigenschaft (1.12) und Theorem 1.27).

2.3 Eindeutigkeit der Suchrichtung

Es ist interessant zu betrachten, unter welchen Umständen das System (2.5) eine eindeutige Lösung mit symmetrischen ΔX und ΔZ hat. In [30] haben Kojima, Shida und Shindo die Existenz und Eindeutigkeit der Lösungen des Systems (2.5) für einige Auswahlen der Matrix P bewiesen. Sie zeigten, daß das System für $P = X^{-\frac{1}{2}}$ immer eine eindeutige Lösung hat, solange $X \succ 0$ und $Z \succ 0$. Für $P = I$ ist dies nicht so. In diesem Fall wurde die zusätzliche Bedingung $XZ + ZX \succeq 0$ hergeleitet. Diese Bedingung ist hinreichend, aber nicht notwendig. Todd, Toh und Tütüncü [32] haben dieses Resultat wie folgt verallgemeinert.

Theorem 2.6 *[32] Es seien X und Z positiv definit. Dann hat das System (2.11) (bzw. (2.6)) eine eindeutige Lösung $(\Delta X, \Delta y, \Delta Z) \in S_n \times \mathbf{R}^m \times S_n$, wenn $E_s^{-1} F_s$ (bzw. $E^{-1}F$) positiv definit ist (aber nicht notwendiger Weise symmetrisch). Im besonderen gilt diese Bedingung, wenn X und Z positiv definit sind und $H_P(XZ)$ positiv semidefinit ist.*

Beweis: Für diesen Beweis impliziert positive Definitheit nicht die Symmetrie einer Matrix; sind Matrizen symmetrisch, werden wir es explizit angeben.
Es ist ausreichend zu zeigen, daß das 3×3 Blocksystem

$$\begin{bmatrix} 0 & \mathbf{A}_s^T & -\mathcal{I} \\ \mathbf{A}_s & 0 & 0 \\ E_s & 0 & F_s \end{bmatrix} \begin{bmatrix} \mathrm{svec}(\Delta X) \\ \Delta y \\ \mathrm{svec}(\Delta Z) \end{bmatrix} = 0 \tag{2.14}$$

nur die triviale Lösung besitzt.
Da E nach (2.12) invertierbar ist, können wir Gauß-Block-Elimination benutzen, um (2.14) auf die Schurkomplement-Gleichung

$$(\mathbf{A}_s E_s^{-1} F_s \mathbf{A}_s^T)\Delta y = 0 \tag{2.15}$$

zu reduzieren. Da $E_s^{-1}F_s$ nach Annahme positiv definit ist und $\mathbf{A}_s$ vollen Zeilenrang hat, ist $\mathbf{A}_s E_s^{-1} F_s \mathbf{A}_s^T$ auch positiv definit. Daher ist $\Delta y = 0$ die einzige Lösung für (2.14).
Nun ergibt sich aus der ersten Blockgleichung in (2.14) $\operatorname{svec}(\Delta Z) = (\mathbf{A}_s^T \Delta y) = 0$ und damit $\Delta Z = 0$. Aus der dritten Blockgleichung folgt schließlich $\Delta X = -E_s^{-1}F_s\Delta Z = 0$. Dies zeigt, daß die Lösung zu (2.14) existiert und eindeutig ist.
Um zu zeigen, daß die Bedingung „$H_P(XZ)$ ist positiv semidefinit" impliziert, daß $E_s^{-1}F_s$ positiv definit ist, betrachten wir einen beliebigen $n(n+1)/2$-dimensionalen Vektor g, der nicht der Nullvektor ist. Wir wollen zeigen, daß $g^T E_s^{-1} F_s g > 0$. Sei $k := E_s^{-T} g$ und $K \in S_n$ so definiert, daß $\operatorname{svec}(K) = k$. Somit ist k auch nicht der Nullvektor. Wir erhalten mit den Rechenregel (1.9),(1.13), dem Theorem 1.27 und der Definition des symmetrisierten Kronecker-Produktes

$$\begin{aligned}
g^T E_s^{-1} F_s g &= k^T F_s E_s^T k = k^T (PX \otimes_s P^{-T})(P^T \otimes_s ZP^{-1})k \\
&= \tfrac{1}{2} k^T (PXP^T \otimes_s P^{-T}ZP^{-1})k + \tfrac{1}{2} k^T (PXZP^{-1} \otimes_s I)k \\
&> \tfrac{1}{2} k^T (PXZP^{-1} \otimes_s I)k \\
&= \tfrac{1}{4} (\operatorname{svec}(K))^T \operatorname{svec}(PXZP^{-1}K + KP^{-T}ZXP^T) \\
&= \tfrac{1}{4} \operatorname{tr}\left[K(PXZP^{-1} + P^{-T}ZXP^T)K\right] \\
&= \tfrac{1}{2} \operatorname{tr}\left[KH_P(XZ)K\right] \geq 0.
\end{aligned}$$

Die strikte Ungleichung ist gegeben, weil $PXP^T \otimes_s P^{-T}ZP^{-1}$ positiv definit ist, eine Konsequenz aus der Tatsache, daß PXP^T und $P^{-T}ZP^{-1}$ symmetrisch positiv definit sind (Theorem 1.27). Die letzte Ungleichung folgt aus der Annahme, daß $H_P(XZ)$ positiv semidefinit ist. Damit ist die Behauptung bewiesen. ■

Das folgende Theorem beschreibt eine Klasse von Matrizen P, die die hinreichende Bedingung von Theorem (2.6) erfüllen.

Theorem 2.7 *[32] X und Z seien positiv definite Matrizen und P sei eine invertierbare Matrix. E_s und F_s seien definiert wie in (2.11). Dann sind folgende Aussagen äquivalent:*

(1) PXP^T und $P^{-T}ZP^{-1}$ kommutieren;

(2) $PXZP^{-1}$ ist symmetrisch;

(3) $F_sE_s^T$ ist symmetrisch;

(4) $E_s^{-1}F_s$ ist symmetrisch.

Aussage (1) impliziert, daß $H_P(XZ)$ positiv definit ist. Folglich impliziert jede dieser Aussagen, daß sowohl $E_s^{-1}F_s$ als auch $F_sE_s^T$ positiv definit sind.

Beweis: Die Äquivalenz von (1) und (2) ergibt sich unmittelbar: PXP^T und $P^{-T}ZP^{-1}$ kommutieren dann und nur dann wenn

$$PXZP^{-1} = (PXP^T)(P^{-T}ZP^{-1}) = (P^{-T}ZP^{-1})(PXP^T) = P^{-T}ZXP^T.$$

Andererseits ist

$$F_sE_s^T = (PX \otimes_s P^{-T})(P^T \otimes_s ZP^{-1}) = \frac{1}{2}(PXP^T \otimes_s P^{-T}ZP^{-1} + PXZP^{-1} \otimes_s I).$$

Der erste Term der rechten Seite ist symmetrisch, da PXP^T und $P^{-T}ZP^{-1}$ es sind. Daher ist $F_sE_s^T$ dann und nur dann symmetrisch, wenn es der zweite Term ist. Aber dies ist dann und nur dann der Fall, wenn $PXZP^{-1}$ symmetrisch ist. Damit ist die Äquivalenz von (2) und (3) gezeigt. Die Äquivalenz von (3) und (4) schließlich ist offensichtlich: $E_s^{-1}F_s = E_s^{-1}(F_sE_s^T)E_s^{-T}$ ist dann und nur dann symmetrisch, wenn $F_sE_s^T$ es ist.
Nehmen wir nun an, daß (1) (und damit auch (2)) gilt. Dann gilt

$$H_P(XZ) = PXZP^{-1} = (PXP^T)(P^{-T}ZP^{-1}),$$

welches ein Produkt zweier kommutierender positiv definiter Matrizen ist. Dies impliziert die positive Definitheit von $H_P(XZ)$ (siehe [10],Theorem 1.3.12). Also ist nach Theorem 2.6 auch $E_s^{-1}F_s$ und damit auch $F_sE_s^T$ positiv definit. ■

2.4 Prediktor-Korrektor-Ansatz

Die meisten Suchrichtungen erlauben die Anwendung des Prediktor-Korrektor-Ansatzes von Mehrotra [17]. Hierbei wird in einem ersten Schritt eine Suchrichtung für $\sigma = 0$, die sogenannte affine Richtung berechnet.

$$\begin{aligned} \mathcal{A}(\Delta\hat{X}) &= -(\mathcal{A}(X) - b) \\ \mathcal{A}^T(\Delta\hat{y}) - \Delta\hat{Z} &= -(\mathcal{A}^T(y) - Z - C) \\ H_P(\Delta\hat{X}Z + X\Delta\hat{Z}) &= -H_P(XZ). \end{aligned}$$

In einem zweiten Schritt wird die Information dieses vorbestimmten Punktes in eine neue rechte Seite integriert und eine zweite Suchrichtung für irgendein σ bestimmt. Dies ist die sogenannte zentrierende Richtung.

$$\begin{aligned} \mathcal{A}(\Delta\check{X}) &= 0 \\ \mathcal{A}^T(\Delta\check{y}) - \Delta\check{Z} &= 0 \\ H_P(\Delta\check{X}Z + X\Delta\check{Z}) &= \sigma\mu I - H_P(\Delta\hat{X}\Delta\hat{Z}). \end{aligned}$$

Danach werden die beiden Richtungen addiert, um die endgültige Suchrichtung zu erhalten.

$$(\Delta X, \Delta y, \Delta Z) = (\Delta\hat{X} + \Delta\check{X}, \Delta\hat{y} + \Delta\check{y}, \Delta\hat{Z} + \Delta\check{Z}).$$

In einigen Versionen des Prediktor-Korrektor-Ansatzes wird die eigentliche Schrittrichtung allerdings auch gleich im zweiten Schritt mit dem System

$$\begin{array}{rcl} \mathcal{A}(\Delta X) & = & -(\mathcal{A}(X) - b) \\ \mathcal{A}^T(\Delta y) - \Delta Z & = & -(\mathcal{A}^T(y) - Z - C) \\ H_P(\Delta X Z + X \Delta Z) & = & \sigma\mu I - H_P(XZ + \Delta\hat{X}\Delta\hat{Z}). \end{array}$$

ermittelt. Diese Suchrichtung ist normalerweise von weit besserer Qualität als die jeweilige direkte Suchrichtung. Da nur die rechte Seite verändert wird, ist der zusätzlich benötigte Rechenaufwand nicht übermäßig groß.

2.5 Schiefsymmetrische Einbettung

In diesem wird kurz der Ansatz der schiefsymmetrischen Einbettung erklärt. Für mehr Informationen über schiefsymmetrische Einbettung seien Potra und Sheng [26], Xu, Hung und Ye [38] und de Klerk, Roos und Terlaky [5] zur Lektüre empfohlen.
Wir beginnen mit dem Homogenisieren des KKT-Systems, indem wir zwei künstliche Variablen τ und ρ einführen.

$$\begin{array}{cccccc} & \mathcal{A}(X) & -\tau b & & & = 0 \\ -\mathcal{A}^T(y) & & +\tau C & +Z & & = 0 \\ b^T y & -\langle C, X\rangle & & & +\rho & = 0 \\ & X \succeq 0, & \tau \geq 0, & Z \succeq 0, & \rho \geq 0. & \end{array}$$

Dieses System ist zulässig, da es von der Null-Lösung erfüllt wird. Nehmen wir an, es gibt eine Lösung (X, y, Z, τ, ρ) mit $\tau > 0$. Dann erhalten wir durch das Dividieren von X, y und Z durch τ primal und dual zulässige Lösungen. Weil die dritte Gleichung äquivalent zu $b^T y \leq \langle C, X\rangle$ ist, welches das Gegenteil der schwachen Dualität ist, sind diese Lösungen sogar optimal. Andererseits wissen wir, daß bei zulässigen primalen und dualen Lösungen $b^T y < \langle C, X\rangle$ unmöglich ist. Also muß die Schlupfvariable ρ bei jeder zulässigen Lösung gleich Null sein. Dieses System hat folglich nie einen strikt zulässigen Punkt. Um einen strikt zulässigen Startpunkt zu erhalten, müssen wir zwei weitere Variablen ϑ und η zusammen mit einigen neuen Konstanten einführen. Dadurch ergibt sich ein neues semidefinites Problem.
min $\quad \vartheta\beta \qquad$ s.t.

$$\begin{array}{cccccccc} & \mathcal{A}(X) & -\tau b & +\vartheta\bar{b} & & & & = & 0 \\ -\mathcal{A}^T(y) & & +\tau C & -\vartheta\bar{C} & +Z & & & = & 0 \\ b^T y & -\langle C, X\rangle & & +\vartheta\alpha & & +\rho & & = & 0 \\ -\bar{b}^T y & +\langle\bar{C}, X\rangle & -\tau\alpha & & & & +\eta & = & \beta \\ & X \succeq 0, & \tau \geq 0, & \vartheta \geq 0, & Z \succeq 0, & \rho \geq 0, & \eta \geq 0, & & \end{array}$$

mit

$$\begin{aligned}\bar{b} &:= -\mathcal{A}(I) + b\\ \bar{C} &:= C + I\\ \alpha &:= \langle C, I\rangle - 1\\ \beta &:= n + 2.\end{aligned}$$

$\bar{b}, \bar{C}, \alpha$ und β sind so gewählt, daß der Punkt $y^0 = 0, X^0 = Z^0 = I, \tau^0 = \vartheta^0 = \rho^0 = \eta^0 = 1$ ein strikt zulässiger Punkt des semidefiniten Problems ist. Außerdem ist nach ein paar Umstellungen zu erkennen, daß das Duale dieses Problems exakt dasselbe Problem ist. Es ist also selbstdual. Insbesondere ist die zu X komplementäre semidefinite Variable Z und umgekehrt. Die anderen komplementären Variablenpaare sind τ und ρ sowie ϑ und η. Da die Bedingungen identisch sind, kann der gleiche Startpunkt gewählt werden. Die primalen und dualen Variablen werden in genau der gleichen Weise verändert, daher ist es ausreichend, nur einen Satz Variablen zu verwalten. Für das KKT-System der primal-dualen Inneren-Punkte-Methoden benötigen wir nur noch die Komplementaritäts-Bedingungen,

$$\begin{aligned}XZ &= \sigma\mu I\\ \tau\rho &= \sigma\mu\\ \vartheta\eta &= \sigma\mu.\end{aligned}$$

Der zusätzliche Rechenaufwand ist relativ klein. Der oben angegebene strikt zulässige Punkt erfüllt diese Komplementaritäts-Bedingungen für $\sigma\mu = \sigma\mu(X^0, Z^0) = 1$. Daher ist es ein Punkt auf dem zentralen Pfad. Offensichtlich erhält man eine optimale Lösung, indem man $\eta = \beta$ und alle anderen Variablen gleich Null setzt. Es macht trotzdem Sinn, dieses System zu lösen, da der Innere-Punkte-Algorithmus eine maximal komplementäre Optimallösung liefert.

Satz 2.8 *[5] Es sei $(X^*, y^*, Z^*, \rho^*, \eta^*, \tau^*, \vartheta^*)$ eine maximal komplementäre Optimallösung der schiefsymmetrischen Einbettung. Dann sind folgende Fälle zu unterscheiden:*

(i) $\tau > 0 \iff$ *Sowohl (1.16) als auch (1.19) sind zulässig, die Dualitätslücke ist Null und die Optimalwerte beider Probleme werden angenommen.*

(ii) $\tau = 0 \wedge \rho > 0 \iff$ *Wenigstens eines der Probleme hat einen verbessernden Strahl, d.h. ist in Richtung der Zielfunktion unbeschränkt.*

(iii) $\tau = 0 \wedge \rho = 0 \iff$ *Entweder sind beide Probleme zulässig mit strikt positiver Dualitätslücke oder der Optimalwert wird nicht von beiden Problemen angenommen oder mindestens eines ist nicht zulässig aber keines der Probleme hat einen verbessernden Strahl.*

Kapitel 3

Verschiedene Suchrichtungen im Vergleich

In diesem Kapitel werden wir einige der am häufigsten genutzten Suchrichtungen vorstellen und miteinander vergleichen. Um die Schurkomplement-Gleichung (2.10) zu lösen, muß zuerst jeweils die Systemmatrix $M = \mathbf{A}E^{-1}F\mathbf{A}^T$ aufgestellt werden. Aus den Eigenschaften des Vektoroperators und der Matrizenmultiplikation ergibt sich für die einzelnen Koeffizienten

$$M_{ij} = \text{vec}(A_i)^T E^{-1} F\text{vec}(A_j) \;\; (i,j = 1,\ldots,m).$$

Dabei ist es von Vorteil, zuerst für $j = 1,\ldots,m$ die Terme $E^{-1}F\text{vec}(A_j)$ zu berechnen. Es ist also wichtig, Ausdrücke der Art $E^{-1}F\text{vec}(H)$ mit $H \in S_n$ effizient ausrechnen zu können.

3.1 Die AHO-Richtung

Diese Suchrichtung wurde von Alizadeh, Haeberly und Overton [3] vorgestellt. Bei der AHO-Richtung ist die Matrix P des Symmetrisierungsoperators gleich der Einheitsmatrix I. Die linearisierte Komplementaritätsbedingung lautet damit

$$\Delta XZ + X\Delta Z + Z\Delta X + \Delta ZX = \sigma\mu I - XZ - ZX.$$

Für die Matrizen E und F aus der Blockstruktur (2.6) ergeben sich daraus die Kronecker-Produkte

$$E = \frac{1}{2}(I \otimes Z + Z \otimes I)$$

und

$$F = \frac{1}{2}(X \otimes I + I \otimes X).$$

In symmetrisierten Kroneckerprodukten ausgedrückt, ergibt dies $E_s = I \otimes_s Z$ und $F_s = X \otimes_s I$. Ausdrücke der Form $\text{vec}(Y) = E^{-1}F\text{vec}(H)$, wobei $H \in S_n$, bestimmt man für die AHO-Richtung günstig mittels der $E\text{vec}(Y) = F\text{vec}(H)$ entsprechenden Lyapunov-Gleichung, die für die symmetrische Matrixvariable Y zu lösen ist,

$$YZ \quad + \quad ZY \quad = \quad HX \quad + \quad XH. \tag{3.1}$$

Um diese Gleichung zu lösen, benötigen wir die Spektral-Faktorisierung (Eigenwertzerlegung) von $Z = Q\Omega Q^T$, wobei Q orthogonal und Ω eine aus den Eigenwerten von Z bestehende Diagonalmatrix ($\Omega = \text{Diag}(\omega_1, \ldots, \omega_n)$) ist. Dann wird (3.1) zu

$$\tilde{Y}\Omega + \Omega\tilde{Y} = Q^T(HX + XH)Q$$

reduziert, wobei $\tilde{Y} = Q^TYQ$. Da Ω diagonal ist, lösen wir diese Gleichung durch

$$\tilde{y}_{ij} = \frac{(Q^T(HX + XH)Q)_{ij}}{\omega_i + \omega_j}.$$

Wir erhalten dann $Y = Q\tilde{Y}Q^T$. Nach einmaligem Berechnen der Spektral-Faktorisierung von Z kann also $B_k = E^{-1}F\text{vec}A_k$ durch Lösen der Gleichung (3.1) für $H = A_k, \quad k = 1, \ldots, m$ aufstellt werden. Anschließend bildet man $M_{ij} = tr(A_iB_k)$ für $i, j = 1, \ldots, m$. Da $E^{-1}F$ nicht symmetrisch ist, ist die Schurkomplement-Matrix M im Allgemeien auch nicht symmetrisch.

3.2 Die NT-Richtung

Die NT-Richtung wurde von Nesterov und Todd [24] entwickelt. Sie basiert im wesentlichen auf einer symmetrischen Skalierungsmatrix W. Diese Matrix erfüllt die Gleichung $W^{-1}XW^{-1} = Z$ oder äquivalent $WZW = X$ oder auch

$$W^{-1}X = ZW. \tag{3.2}$$

Man prüft leicht nach, daß

$$\begin{aligned} W &= X^{\frac{1}{2}}(X^{\frac{1}{2}}ZX^{\frac{1}{2}})^{-\frac{1}{2}}X^{\frac{1}{2}} \\ &= Z^{-\frac{1}{2}}(Z^{\frac{1}{2}}XZ^{\frac{1}{2}})^{\frac{1}{2}}Z^{-\frac{1}{2}} \end{aligned}$$

diese Voraussetzungen erfüllt. In der Tat ist W sogar eindeutig und der sogenannte metrisch geometrische Durchschnitt von X und Z^{-1} [32]. Die Matrix P des Symmetrisierungsoperators wird so gewählt, daß $P^TP = W^{-1}$ ist (Beispiele dafür sind $P = W^{-\frac{1}{2}}$ und $P = (X^{\frac{1}{2}}ZX^{\frac{1}{2}})^{\frac{1}{4}}X^{-\frac{1}{2}}$). Wir erhalten damit als linearisierte Komplementaritätsgleichung z.B. (für $P = W^{-\frac{1}{2}}$)

$$W^{-\frac{1}{2}}(\Delta XZ + X\Delta Z)W^{\frac{1}{2}} + W^{\frac{1}{2}}(Z\Delta X + \Delta ZX)W^{-\frac{1}{2}} = 2\sigma\mu I - W^{-\frac{1}{2}}XZW^{\frac{1}{2}} - W^{\frac{1}{2}}ZXW^{-\frac{1}{2}}$$

Diese Gleichung ist schwer nach ΔX aufzulösen. Wir wollen statt (2.5) lieber folgendes System lösen (siehe [32]),

$$\begin{aligned} \mathcal{A}(\Delta X) &= -(\mathcal{A}(X) - b) \\ \mathcal{A}^T(\Delta y) - \Delta Z &= -(\mathcal{A}^T(y) - Z - C) \\ W^{-1}\Delta X W^{-1} + \Delta Z &= \sigma\mu X^{-1} - Z. \end{aligned} \tag{3.3}$$

Dazu beweisen wir, daß diese Systeme dieselbe Lösung liefern.

Behauptung 3.1 *[32] Es sei P eine nichtsinguläre Matrix, so daß $P^T P = W^{-1}$. Dann sind die Lösungen für die Systeme (2.5) und (3.3) identisch.*

Beweis: Es genügt zu zeigen, daß die Gleichungen (2.4) und dritte Gleichung von (3.3) äquivalent sind. Durch Premultiplikation mit P^T und Postmultiplikation mit P^{-T} und $P^T P = W^{-1}$ kann (2.4) umgeschrieben werden in

$$W^{-1}(X\Delta Z + \Delta X Z)W + \Delta Z X + Z\Delta X = 2\sigma\mu I - 2ZX. \tag{3.4}$$

Die letzte Gleichung von (3.3) ist andererseits äquivalent zu jeder der beiden folgenden Gleichungen.

$$\begin{aligned} W^{-1}\Delta X W^{-1} X + \Delta Z X &= \sigma\mu I - ZX \\ Z\Delta X + ZW\Delta Z W &= \sigma\mu Z W X^{-1} W - ZWZW \end{aligned}$$

Um die erste Gleichung zu erhalten, haben wir die dritte Gleichung von (3.3) von rechts mit X multipliziert. Für die zweite Gleichung haben wir sie von links mit ZW und von rechts mit W multipliziert. Durch Addieren der beiden Gleichungen und Anwenden von sowohl (3.2) als auch seinen äquivalenten Varianten $ZWX^{-1}W = I$ und $WZW = X$ erhalten wir (3.4). Also erfüllt jede Lösung von (3.3) auch (2.5). Da (3.3) mindestens eine Lösung hat und (2.5) in unserem Fall eine eindeutige Lösung hat (siehe Theorem 2.6) sind die Systeme äquivalent. ■

Damit ergibt sich für die Elemente aus der Blockstruktur (2.6) $E = W^{-1} \otimes W^{-1}$, $F = I \otimes I$ und $R = Z - \sigma\mu X^{-1}$ und für die symmetrisierte Variante (2.11) $E_s = W^{-1} \otimes_s W^{-1}$ und $F_s = I \otimes_s I$. Wir berechnen Terme der Art $E^{-1}F\mathrm{vec}(H)$ nun einfach mit

$$E^{-1}F\mathrm{vec}(H) = E^{-1}\mathrm{vec}(H) = \mathrm{vec}(WHW).$$

Also errechnet man zuerst $B_j := WA_jW$ und anschließend $M_{ij} = \mathrm{tr}(A_i B_j)$ für $i, j = 1, \ldots, m$. Hierbei erhält man eine symmetrische Matrix M, da $E^{-1}F$ symmetrisch ist und damit gilt

$$M_{ij} = \mathrm{vec}(A_i)^T E^{-1} F \mathrm{vec}(A_j) = \mathrm{vec}(A_j)^T E^{-1} F \mathrm{vec}(A_i) = M_{ji}.$$

3.2.1 Berechnung von W

Wir wollen zeigen, daß W durch zwei Cholesky-Faktorisierungen und eine Singulärwertzerlegung berechnet werden kann. Wir folgen dabei den Ausführungen von Todd et al. [32]. Einen Algorithmus zur Berechnung der Singulärwertzerlegung findet man in dem Buch von Golub und van Loan [7]. Die Cholesky-Faktorisierungen von X und Z seien

$$X = LL^T \qquad \text{und} \qquad Z = NN^T$$

und $UDV^T = N^T L$ sei die Singulärwertzerlegung von $N^T L$, wobei U und V orthonormale Matrizen sind und D eine Diagonalmatrix ist. Wir definieren die (nach dem später im Text angegebenen Lemma 3.3 orthogonale) Matrix $Q = L^{-1}X^{\frac{1}{2}}$. Damit ist

$$X^{\frac{1}{2}}ZX^{\frac{1}{2}} = Q^T(L^T N)(N^T L)Q = (Q^T V)D^2(V^T Q).$$

Es sei bemerkt, daß VD^2V^T die Eigenwertzerlegung von $L^T NN^T L$ ist. Wir könnten statt der Singulärwertzerlegung also auch diese benutzen. Der Algorithmus wäre dann aber numerisch weniger stabil, da die Kondition der Matrix $L^T ZL$ das Quadrat der Kondition von $N^T L$ ist (siehe [32]).
Da $Q^T V$ orthogonal ist, erhalten wir

$$(X^{\frac{1}{2}}ZX^{\frac{1}{2}})^{-\frac{1}{2}} = (Q^T V)D^{-1}(V^T Q). \tag{3.5}$$

Aus (3.5) läßt sich W einfach berechnen:

$$W = LVD^{-1}V^T L^T = GG^T,$$

wobei

$$G := LVD^{-\frac{1}{2}}.$$

3.3 Die HKM-Richtung

Die HKM-Richtung wurde von Helmberg, Rendl, Vanderbei und Wolkowicz [9] und gleichzeitig unabhängig davon von Kojima, Shindoh und Hara [13] entwickelt. Später wurde sie von Monteiro [18] wiederentdeckt.
Helmberg et al. erhielten diese Richtung, indem sie das asymmetrische System (2.3) lösten und ein im Allgemeinen nicht symmetrisches ΔX danach symmetrisierten. Erst Monteiro hat die Richtung durch Symmetrisieren mit $Z^{\frac{1}{2}}$ erhalten, was Zhang [40] die Idee zum H_P-Schema (2.5) gegeben hat. Man erhält die Richtung also, wenn man für die Matrix des Symmetrisierungsoperators $P = Z^{\frac{1}{2}}$ wählt. Daraus ergibt sich die Komplementaritätsgleichung

$$Z^{\frac{1}{2}}\Delta XZ^{\frac{1}{2}} + \frac{1}{2}(Z^{\frac{1}{2}}X\Delta ZZ^{-\frac{1}{2}} - Z^{-\frac{1}{2}}\Delta ZXZ^{\frac{1}{2}}) = \sigma\mu I - Z^{\frac{1}{2}}XZ^{\frac{1}{2}}.$$

Wir können sie durch Pre- und Postmultiplikation mit $Z^{-\frac{1}{2}}$ zu

$$\Delta X + \frac{1}{2}(X\Delta Z Z^{-1} + Z^{-1}\Delta Z X) = \sigma\mu Z^{-1} - X \tag{3.6}$$

vereinfachen. Es ergeben sich damit für $E = I \otimes I$ und für $F = \frac{1}{2}(X \otimes Z^{-1} + Z^{-1} \otimes X)$ bzw. $E_s = I \otimes_s I$ und $F_s = X \otimes_s Z^{-1}$.

Diese dritte Gleichung unterscheidet sich kaum von der in System (2.3). Und in der Tat ist es egal, ob man (2.3) löst und ΔX anschließend symmetrisiert (wie im Algorithmus von Helmberg et al. [9]) oder das System mit der Komplementaritätsgleichung (3.6) löst. Um dies zu zeigen, nehmen wir an $\overline{\Delta X}$ löst das System (2.3), dann gilt

$$\overline{\Delta X} = \sigma\mu Z^{-1} - X - X\Delta Z Z^{-1}.$$

Damit ist

$$\overline{\Delta X}^T = \sigma\mu Z^{-1} - X - Z^{-1}\Delta Z X$$

und weiterhin für das symmetrisierte

$$\Delta X = \frac{\overline{\Delta X} + \overline{\Delta X}^T}{2} = \sigma\mu Z^{-1} - X - \frac{1}{2}(X\Delta Z Z^{-1} + Z^{-1}\Delta Z X).$$

In der Implementation, die in Kapitel 4 besprochen wird, benutzen wir die Variante von Helmberg et al., da diese weniger aufwendig ist.
Wie bei der NT-Richtung ist auch hier das berechnen von $E^{-1}F\mathrm{vec}(H)$ nun sehr einfach. Es gilt

$$E^{-1}F\mathrm{vec}(H) = \mathrm{vec}(XHZ^{-1}).$$

Also errechnet man $B_j := XA_jZ^{-1}$ und anschließend $M_{ij} = \mathrm{tr}(A_iB_j)$ für $i, j = 1, \ldots, m$. Auch hier erhält man ein symmetrisches M, da $E^{-1}F$ symmetrisch ist.

Tabelle 3.1: **Zusammenfassung der Suchrichtungen**

Richtung	AHO ($P = I$)	HKM ($P = Z^{1/2}$)	NT ($P^TP = W^{-1}$)
E	$1/2(I \otimes Z + Z \otimes I)$	$I \otimes I$	$W^{-1} \otimes W^{-1}$
F	$1/2(X \otimes I + I \otimes X)$	$1/2(X \otimes Z^{-1} + Z^{-1} \otimes X)$	$I \otimes I$
$E^{-1}F$	$(I\otimes Z+Z\otimes I)^{-1}(X\otimes I+I\otimes X)$	$1/2(X \otimes Z^{-1} + Z^{-1} \otimes X)$	$W \otimes W$
E_s	$I \otimes_s Z$	$I \otimes_s I$	$W^{-1} \otimes_s W^{-1}$
F_s	$X \otimes_s I$	$X \otimes_s Z^{-1}$	$I \otimes_s I$
$E_s^{-1}F_s$	$(I \otimes_s Z)^{-1}(X \otimes_s I)$	$X \otimes_s Z^{-1}$	$W \otimes_s W$
R_c	$\sigma\mu I - XZ - ZX$	$\sigma\mu Z^{-1} - X$	$\sigma\mu X^{-1} - Z$

3.4 Eigenschaften der Suchrichtungen

Wir untersuchen nun einige Eigenschaften in denen sich die Suchrichtungen unterscheiden. Wir folgen dabei der Darstellungen von Todd [33] und Todd et al. [32].

3.4.1 Eindeutigkeit

Wie wir in Abschnitt 2.3 gesehen haben, ist die Suchrichtung eindeutig, wenn $E_s^{-1}F_s$, für alle positiv definiten X und Z, selbst positiv definit ist, da die Schurmatrix $\mathbf{A}_s E_s^{-1} F_s \mathbf{A}_s$ in diesem Fall nicht singulär ist. Eine hinreichende Bedingung für die positive Definitheit von $E_s^{-1}F_s$ ist die Semidefinitheit von $H_P(XZ)$. Ist $P = X^{-\frac{1}{2}}$ oder allgemeiner $P = (X^{\frac{1}{2}}ZX^{\frac{1}{2}})^{\alpha}X^{-\frac{1}{2}}$ für eine beliebiges α, dann ist

$$\begin{aligned} H_P(XZ) &= \frac{1}{2}[(X^{\frac{1}{2}}ZX^{\frac{1}{2}})^{\alpha}X^{-\frac{1}{2}}XZX^{\frac{1}{2}}(X^{\frac{1}{2}}ZX^{\frac{1}{2}})^{-\alpha} \\ &+ ((X^{\frac{1}{2}}ZX^{\frac{1}{2}})^{\alpha}X^{-\frac{1}{2}}XZX^{\frac{1}{2}}(X^{\frac{1}{2}}ZX^{\frac{1}{2}})^{-\alpha})^T] \\ &= \frac{1}{2}[X^{\frac{1}{2}}ZX^{\frac{1}{2}} + (X^{\frac{1}{2}}ZX^{\frac{1}{2}})^T] \\ &= X^{\frac{1}{2}}ZX^{\frac{1}{2}}. \end{aligned}$$

$X^{\frac{1}{2}}ZX^{\frac{1}{2}}$ ist immer positiv definit, wenn X und Z es sind.
Analog ist für $P = (Z^{\frac{1}{2}}XZ^{\frac{1}{2}})$ der Symmetrisierungsoperator $H_P(XZ) = Z^{\frac{1}{2}}XZ^{\frac{1}{2}}$. Auch dies ist, für positiv definite X und Z, immer positiv definit. Wir sehen also, daß für die HKM- und die NT-Richtung die Eindeutigkeit gesichert ist.
Für die AHO-Richtung, also $P = I$, fällt die Bedingung aus Theorem 2.6 mit dem Ergebnis von Kojima et al. [30] zusammen. Die Richtung ist also eindeutig, wenn $XZ + ZX \succeq 0$. Es gibt allerdings Beispiele, in denen die Koeffizientenmatrix im System (2.11) singulär ist. Ein solches Beispiel ist gegeben durch (siehe Todd et al. [32])

$$\mathbf{A}_s = \begin{bmatrix} -1 & 2 & 0 \end{bmatrix}, X = \begin{bmatrix} 1 & \sqrt{2} \\ \sqrt{2} & 3 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & 11 \end{bmatrix}.$$

In diesem Fall ist die Schurkomplementmatrix $\mathbf{A}_s E_s^{-1} F_s \mathbf{A}_s$ die 1×1 Nullmatrix.

3.4.2 Primal-Duale Symmetrie

Die Wahl der Matrix P hängt bei vielen Suchrichtungen, wie zum Beispiel den hier vorgestellten HKM- und NT-Richtungen, von den aktuellen Iterierten X und Z ab. Wir können deshalb, um diese Abhängigkeit hervorzuheben auch $P(X, Z)$ schreiben.
Eine Suchrichtung heißt primal-dual symmetrisch, wenn sie und die zu ihr duale Richtung definiert durch $\tilde{P} = P(Z, X)^{-T}$ gleich sind. Also sind sowohl die AHO- als auch die NT-Richtung primal-dual symmetrisch, die HKM-Richtung allerdings nicht. Die durch $P = X^{-\frac{1}{2}}$ definierte Richtung wird auch als duale HKM-Richtung bezeichnet und hat ähnliche Eigenschaften wie die HKM-Richtung (siehe Todd [33]).

3.4.3 Skaleninvarianz

Wir betrachten eine Skalierung oder einen *Wechsel der Variablen* im Problem (1.16), bei dem X durch $\bar{X} := UXU^T$ ersetzt wird. Mit diesen Variablen wird (1.16) dann zu

$$\begin{array}{rl} \max_{\bar{X}} & \langle \bar{C}, \bar{X} \rangle \\ \text{s.t.} & \bar{\mathcal{A}}(\bar{X}) = b \\ & \bar{X} \succeq 0, \end{array} \tag{3.7}$$

wobei $\bar{C} := U^{-T}CU^{-1}$ ist und $\bar{\mathcal{A}}$ und $\bar{\mathcal{A}}^T$ durch $\{\bar{A}_i := U^{-T}A_iU^{-1}\}$ wie in (1.17) und (1.18) definiert sind. Das Duale dieses Problems ist

$$\begin{array}{rl} \min_{\bar{y}} & b^T\bar{y} \\ \text{s.t.} & \bar{Z} = \bar{\mathcal{A}}^T(\bar{y}) - \bar{C} \succeq 0 \\ & \bar{y} \in \mathbf{R}^m, \end{array} \tag{3.8}$$

welches aus (1.19) durch einen Wechsel der Variablen resultiert, bei dem y durch $\bar{y} := y$ und Z durch $\bar{Z} := U^{-T}ZU^{-1}$ ersetzt werden. Nun ist es einfach, den Begriff der Skaleninvarianz zu definieren.

Definition 3.2 *Eine Methode für das Bestimmen einer Suchrichtung heißt skaleninvariant, wenn die Richtung bei jeder Iterierten ausgewertet und anschließend mit einer nichtsingulären Matrix U skaliert dieselbe ist, als wenn man das skalierte Problem mit skalierter Iterierten nach dieser Methode löst, daß heißt wenn*

$$(\Delta\bar{X}, \Delta\bar{y}, \Delta\bar{Z}) = (U\Delta XU^T, \Delta y, U^{-T}\Delta ZU^{-1}).$$

Folgendes Lemma und Theorem zur Charakterisierung der Skaleninvarianz sind [32] entnommen.

Lemma 3.3 *[32] Sei $B = CC^T$ positiv definit mit $C \in M_n$. Dann ist $Q := C^{-1}B^{1/2}$ orthgonal*

Beweis: Wir sehen, daß $QQ^T = C^{-1}BC^{-T} = C^{-1}CC^TC^{-T} = I$. ■

Theorem 3.4 *[32] Es sei P aus X und Z so definiert, daß wenn X und $\bar{X} := UXU^T$ und Z in $\bar{Z} := U^{-T}ZU^{-1}$ transformiert werden, P nach $\bar{P}$ mit $\bar{P}^T\bar{P} = U^{-T}P^TPU^{-1}$ transformiert wird. Weiterhin sei $H_P(XZ)$ positiv definit. Dann ist die Suchrichtung, die das System (2.11) löst, skaleninvariant.*

An diesem Theorem sieht man, daß sowohl die HKM- als auch die NT-Richtung skaleninvariant sind (siehe Abschnitt 3.4.1). Die AHO-Richtung ist nicht skaleninvariant, da XZ zu $U^{-1}XZU$ transformiert wird, während ZX zu U^TZXU^{-T} transformiert wird.

Für eine noch ausführlichere Betrachtung von Suchrichtungen und deren Eigenschaften siehe Todd [33].

Tabelle 3.2: **Zusammenfassung der Eigenschaften**

Richtung		primal-duale Symmetrie	Skalen-invarianz	eindeutig definierte Richtung	Systemmatrix symmetrisch
AHO	$(P = I)$	ja	nein	nein	nein
HKM	$(P = Z^{1/2})$	nein	ja	ja	ja
NT	$(P^T P = W^{-1})$	ja	ja	ja	ja

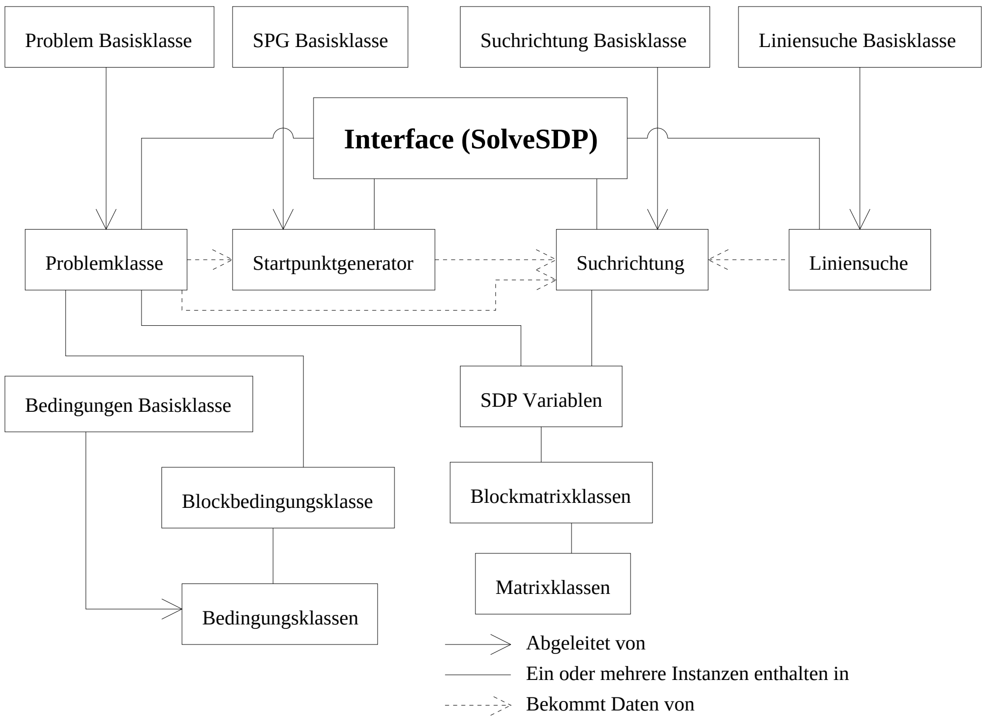
Quelle: Todd, Toh, Tütüncü [32]

Kapitel 4

Implementation

Die Implementation des Innere-Punkte-Verfahrens mit verschiedenen Suchrichtungen folgte einem objektorientierten Ansatz. Bei diesem wurde besonderes Augenmerk auf Flexibilität und Erweiterbarkeit gelegt. Die Programmierung erfolgte in C++. Als Grundlage dienten die bereits von C. Helmberg geschriebenen Matrixklassen. In diesem Kapitel werden die Klassenstruktur und die wichtigsten Klassen beschrieben.

4.1 Klassenstruktur



Die Abbildung zeigt die Struktur der C++-Klassen dieser Implementation. Die zentrale Klasse ist dabei das Interface, über welches der Benutzer auf das Problem, die Suchrichtun-

gen, die Parameter etc. zugreifen kann. Über das Interface kann der Benutzer auch eigene Suchrichtungen, Methoden zur Liniensuche oder Startpunktgeneratoren einbinden, solange diese von der entsprechenden Basisklasse abgeleitet sind.
Die Hauptschleife des Interfaces sieht folgendermaßen aus:

1. Setze Iterationszähler $i = 1$.

2. Solange kein Abruchkriterium erfüllt ist:

   (a) Berechne σ den Schritt $(\Delta X_i, \Delta y_i, \Delta Z_i)$ nach der festgelegten Methode.

   (b) Ermittle die Schrittlängen α_p und α_d (siehe Liniensuche Abschnitt 4.5).

   (c) Berechne die nächste Iterierte:
   $X_i = X_{i-1} + \alpha_p \Delta X_i$
   $y_i = y_{i-1} + \alpha_d \Delta y_i$
   $Z_i = Z_{i-1} + \alpha_d \Delta Z_i$

   (d) Setze $i = i + 1$.

3. Gehe zu 2.

Die Hauptschleife bricht ab,

- wenn die relative Lücke zwischen dem primalen und dem dualen Zielfunktionswert $\frac{\langle X_i, Z_i \rangle}{\langle C, X_i \rangle}$ kleiner als ein festgelegter Wert ε_{gap} ist
und das Maß für die Unzulässigkeit $\phi = \max\left(\frac{\|b - \mathcal{A}(X)\|}{1+\|b\|}, \frac{\|C + Z + \mathcal{A}^T(y)\|_F}{1+\|C\|_F}\right)$ kleiner als der Parameter ε_{inf} ist,

- wenn die Systemmatrix singulär wird,

- wenn eine der Schrittlängen kleiner als ein festgelegter Wert ε_{step} ist,

- wenn primale Zielfunktionwert über $f * \|\mathcal{A}^T(y) + Z\|_F$
oder der duale Zielfunktionswert über $f * \|\mathcal{A}(X)\|_F$ steigt
und daher der Verdacht auf Unzulässigkeit besteht, wobei f ein zuvor festgelegter Parameter ist oder

- wenn die maximal gewünschte Anzahl an Iterationen ereicht wird.

Ein einfaches Programm das mit dem Interface interagiert könnte dann zum Beispiel so strukturiert sein:

1. Lese das zu lösende Problem ein $(C, \mathcal{A}, b)$.

2. Lege die Methode für die Suchrichtung fest.

3. Lege die Iterations- und Abbruchparameter fest oder benutze die vorgegebenen Parameter.

4. Generiere Startpunkt (X_0, y_0, Z_0).

5. Starte die Hauptschleife.

Werden die Parameter nicht anderweitig bestimmt, gelten in der Implementation die Standardwerte:

$$\begin{aligned} \varepsilon_{gap} &= 10^{-6}, \\ \varepsilon_{inf} &= 5 * 10^{-6}, \\ \varepsilon_{step} &= 10^{-12}, \\ f &= 10^{8}. \end{aligned}$$

4.2 Blockmatrizen

Bisher haben wir nur Probleme mit einer einzigen semidefiniten Matrixvariable betrachtet. Nun stellt sich die Frage, warum man nicht auch Probleme mit mehreren solchen Variablen betrachten sollte. Für die meisten theoretischen Betrachtungen ist dies nicht notwendig. Tatsächlich kann jedes semidefinte Problem mit mehreren Variablen als äquivalentes Problem einer Variable formuliert werden, da

$$X_1 \succeq 0, X_2 \succeq 0, \ldots, X_k \succeq 0 \Longleftrightarrow \begin{bmatrix} X_1 & 0 & \cdots & 0 \\ 0 & X_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & X_k \end{bmatrix} \succeq 0.$$

Nun ist es auch einfach zu sehen, daß Lineare Optimierung ein Spezialfall der Semidefiniten Optimierung ist, wenn man jede einzelne Komponente $x_i \geq 0$ als positiv semidefinite 1×1 Matrix interpretiert.
In der Praxis haben viele Probleme diese oben angegebene Blockdiagonalstruktur. Um diese Struktur zu nutzen, existieren in unserer Implementation die Blockmatrixklassen. Dabei gibt es einerseits die Klasse der symmetrischen Blockmatrizen und andererseits die Klasse der unsymmetrischen Blockmatrizen. Beide bestehen aus einem Array entsprechender Matrizen und einer Reihe von Operationen. Die meisten dieser Operationen werden in einer Schleife an die einzelnen Matrizen weitergegeben.

4.3 Nebenbedingungen

Beim Aufstellen der Systemmatrix $\mathbf{A}E^{-1}F\mathbf{A}^T$ ist die Struktur der Nebenbedingungen A_i ein wichtiger Punkt. Bei vielen praktischen Problemen haben die A_i eine sehr einfache Struktur, die es auszunutzen gilt. Für die wichtigsten Strukturen existieren in der Implementation eigene Klassen. Die im Moment vorhandenen Klassen sind

- *Symcons* - für vollbesetzte Matrizen ohne besondere Struktur,

- *Sparscons* - für dünn besetzte Matrizen,

- *Vecvect* - für Matrizen der Form vv^T also des Ranges 1 und
- *Svecvect* - für Rang-1 Matrizen mit dünn besetzem Vektor v.

Diese Klassen sind alle von einer gemeinsamen Basisklasse abgeleitet.
Auch für die Nebenbedingungen wurde die oben beschriebene Blockstruktur vorgesehen. Wobei jeder Block eine andere Struktur haben kann.
Die Struktur wird beim aufstellen der Systemmatrix an zwei Stellen ausgenutzt. Zum einen beim Berechnen von $B_i = E^{-1}F\mathrm{vec}(A_i)$ und zum anderen beim Berechnen des inneren Produkts $\mathrm{tr}(A_iB_j)$.
Bei Nebenbedingungen mit *Sparscons*-Struktur wird dies durch spezielle Methoden zur Multiplikation und zur Berechnung des inneren Produkts mit dünn besetzten Matrizen gewährleistet.
Das Aufstellen von $E^{-1}F\mathrm{vec}(A_i)$ mit Nebenbedingungen vom Rang 1 erfolgt z.B. bei der HKM-Richtung durch

$$E^{-1}F\mathrm{vec}(A_i) = XA_iZ^{-1} = Xvv^TZ^{-1} = (Xv)(v^TZ^{-1}).$$

Man benötigt also statt zwei Matrixmultiplikationen, nur zwei Matrix-Vektor- und eine Vektor-Vektor-Multiplikation. Bei der NT-Richtung ist es entsprechend. Die AHO-Richtung nutzt die Struktur in der Lyapunov-Gleichung, beim Berechnen von $A_iX + XA_i$.
Beim inneren Produkt berechnen wir zuerst $w^T = v^TB$ und anschließend

$$\mathrm{tr}(vw^T) = \sum_{i=1}^{n} v_iw_i = w^Tv.$$

Wir benötigen also nur einen Matrix-Vektor-Multiplikation und das innere Produkt zweier Vektoren.
Bei Rang-1-Nebenbedingungen mit dünner Besetzung wird dies entsprechend ausgenutzt.

4.4 Die Problemklasse

Die Problemklasse enthält die Variablen, Kostenfunktion und Nebenbedingungen des semidefiniten Problems. Sie bietet verschiedene Methoden zum einlesen und auslesen des Problems und seiner einzelnen Bestandteile. Die Variablen X und Z und die Kostenmatrix C haben dabei die oben unter Blockmatrizen angegebene Struktur. Für einen eventuell vorhandenen linearen Anteil des Problems sind aus Effizienzgründen eigene Variablen, Kostenfunktion und Bedingungen hinzugefügt. In den implementierten Suchrichtungen wird der lineare Teil gesondert betrachtet, falls er vorhanden ist. Die benutzte Form semidefiniter Probleme ist in dieser Implementation also eigentlich

$$\begin{array}{rl} \max & \langle C, X\rangle + a^Ts \\ \text{s.t.} & \mathcal{A}(X) + Bs = b \\ & X \succeq 0, s \geq 0. \end{array} \tag{4.1}$$

Dabei sind die lineare Kostenfunktion a und die lineare Variable s in $\mathbf{R}^n$ und die Koeffizientenmatrix $B \in M_{n,m}$. Das entsprechende duale Problem ist

$$\begin{array}{ll} \min & b^T y \\ \text{s.t.} & Z = \mathcal{A}^T(y) - C \succeq 0 \\ & t = B^T y - a \geq 0 \\ & y \in \mathbf{R}^m. \end{array} \tag{4.2}$$

Die duale lineare Variable t ist ebenfalls in $\mathbf{R}^n$.
Die Hilfsvariablen für die schiefsymmetrische Einbettung aus Abschnitt 2.5 sind ebenfalls Bestandteil der Problemklasse. Sämtliche Variablen sind dabei in der Klasse der SDP-Variablen zusammengefasst.
Mit Hilfe der Methode *consistency()* wird festgestellt, ob die Anzahl und Größe der einzelnen Variablen, Koeffizienten und Blöcke der Variablen und Koeffizienten übereinstimmen.

4.5 Liniensuche

Für die Wahl von α, auch Liniensuche genannt, wurde ein einfacher Algorithmus implementiert.

Algorithmus 4.1 *Input: Derzeitige Iterierte* (X, y, Z) *und Schrittrichtung* $(\Delta X, \Delta y, \Delta Z)$.

1. *Setze* $\alpha_p = 1$ *und* $\alpha_d = 1$.
2. *Prüfe ob* $X + \alpha_p \Delta X$ *positiv definit ist.*
3. *Wenn nicht: Setze* $\alpha_p = 0,8 \cdot \alpha_p$ *und gehe zu 2. Sonst weiter.*
4. *Prüfe ob* $Z + \alpha_d \Delta Z$ *positiv definit ist.*
5. *Wenn nicht: Setze* $\alpha_d = 0,8 \cdot \alpha_d$ *und gehe zu 4 Sonst weiter.*
6. *Wenn* $alpha_p < 1$*: Setze* $\alpha_p = 0,95 \cdot \alpha_p$*. (Um nicht zu nah an den Rand zu kommen.)*
7. *Wenn* $alpha_d < 1$*: Setze* $\alpha_d = 0,95 \cdot \alpha_d$*. (Um nicht zu nah an den Rand zu kommen.)*

Das Testen der positiven Definitheit wird hierbei mittels einer LDL^T-Faktorisierung durchgeführt.

4.6 Startpunkt

Der implementierte Startpunktgenerator orientiert sich am Startpunkt den auch Toh, Todd und Tütüncü für ihr SDPT3 [34] wählen. Nehmen wir an, die Matrizen A_k und C seien blockdiagonal (siehe 4.2) mit l Blöcken der Dimensionen $n_1, n_2, \ldots, n_l$. Der i-te Block der

A_k bzw. C soll mit $A_k^{(i)}$ bzw. $C^{(i)}$ bezeichnet werden. Dann wird folgender Startpunkt gewählt:

$$X^0 = Diag(\xi_i I_{n_i}), \quad y^0 = 0, \quad Z^0 = Diag(\zeta_i I_{n_i}),$$

wobei $i = 1, \ldots, l$, I_{n_i} die Einheitsmatrix der Größe n_i bezeichnet und

$$\xi_i = n_i \max_{1 \le k \le m} \frac{1 + |b_k|}{1 + \|A_k^{(i)}\|_F}, \quad \zeta_i = \frac{1 + \max[\max_k\{\|A_k^{(i)}\|_F\}, \|C^{(i)}\|_F]}{\sqrt{n_i}}.$$

4.7 Löser

Für jede der Suchrichtungen wurden eigene Löserklassen implementiert, die auf einer gemeinsamen Basisklasse aufbauen. Es existieren jeweils zwei Klassen pro Suchrichtung, eine „normale“ und eine Prediktor-Korrektor Version. Für die Prediktor-Korrektor Versionen wurde der zweite in Abschnitt 2.4 beschriebene Ansatz benutzt. Jede Klasse enthält eine Instanz der SDP-Variablen, in der die Schrittrichtung gespeichert wird. Weiterhin bestehen die Klassen aus Methoden zum Berechnen von Termen der Form $E^{-1}F\text{vec}(H)$, zum Aufstellen der Systemmatrix M und der Hauptmethode, die die Schurkomplementgleichung aufstellt und das System löst. Die Suchrichtungen sind wie in Kaptitel 3 beschrieben implementiert.
Beim Aufstellen der Systemmatrix für die AHO-Richtung werden zunächst die A_i und X mit Q zu $\hat{A}_i = Q^T A_i Q$ und $\hat{X} = Q^T X Q$ transformiert und die Lyapunov-Gleichungen in der abgewandelten Form

$$\tilde{y}^i_{kl} = \frac{(\hat{A}_i \hat{X} + \hat{X} \hat{A}_i)_{kl}}{\omega_k + \omega_l}$$

gelöst. Anschließend wird die Systemmatrix $M_{ij} = \text{tr}(\hat{A}_i \tilde{Y}^j)$ gebildet. Haben die Nebenbedingungen eine dünn besetzte Struktur (siehe Abschnitt 4.3), so büßen sie sie durch die Transformation leider ein.

4.8 Wahl von σ

Die Wahl des Parameters σ folgt Todd et al. [34]. In den „normalen“ Varianten ist $\sigma = 0.5$ für die erste Iteration und $\sigma = 1 - 0.9 * \min(\alpha_p, alpha_d)$, mit den Schrittweiten α_p und α_d des jeweils vorherigen Schrittes, für alle weiteren Iterationen.
Im Falle der Prediktor-Korrektor-Variante ist σ im Prediktor-Schritt Null (siehe Abschnitt 2.4). Danach werden Prediktor-Schrittweiten $\hat{\alpha_p}$ und $\hat{alpha_d}$, wie in Abschnitt 4.5 beschrieben, bestimmt. Mit Hilfe dieser wird σ für den Korrektor-Schritt wie folgt berechnet:

$$\sigma = \min\left(1, \left[\frac{\langle X + \hat{\alpha_p}\Delta\hat{X}, Z + \hat{\alpha_d}\Delta\hat{Z}\rangle}{\langle X, Z\rangle}\right]^e\right),$$

dabei ist der Exponent e

$$e = \begin{cases} \max\left[expon, 3\min(\hat{\alpha_p}, \hat{\alpha_d})^2\right] & \text{if } \mu > 10^{-6}, \\ expon & \text{if } \mu \leq 10^{-6}. \end{cases}$$

Der Parameter *expon* wird, wie bei [34] für die AHO-Richtung auf 3 und für die anderen beiden Suchrichtungen auf 1 gesetzt.

Kapitel 5

Numerische Analyse

Die Implementation wurde mit verschiedenen Problemen getestet, zum einen einer Reihe von MAXCUT-Problemen für zufällige Graphen und zum anderen einer Auswahl von Problemen aus der SDPLIB von Borchers [4]. Anschließend wurden die Ergebnisse mit denen des MATLAB-Software-Paketes SDPT3 von Toh, Todd und Tütüncü [34] verglichen. Alle Berechnungen wurden auf einer SunSparc10 mit 96MB Ram durchgeführt. Es wurde iteriert, bis die relative Lücke zwischen primalem und dualem Zielfunktionswert den Wert $\varepsilon_{gap} = 10^{-6}$ unterschritt oder anderweitig abgebrochen wurde (siehe Abschnitt 4.1).

5.1 MAXCUT-Probleme

Die MAXCUT-Probleme wurden ausgewählt, da die Nebenbedingungen dieser Probleme eine sehr einfache Struktur haben. Sie sehen, als semidefinites Problem formuliert, folgendermaßen aus:

$$\begin{array}{rl} \max & \langle L, X \rangle \\ \text{s.t.} & \operatorname{diag}(X) = e/4, \\ & X \succeq 0, \end{array}$$

wobei $L = \operatorname{Diag}(Be) - B$. e ist der Vektor aus nur Einsen und B ist die Adjazenzmatrix eines Graphen. Die Operatoren diag(.) und Diag(.) wandeln die Diagonale eines Vektors in eine Matrix bzw. einen Vektor in eine Diagonalmatrix um.
Übersetzt man die Nebenbedingung in unsere Standardform, so erhält man $A_i = e_i e_i^T, i = 1, \ldots, n$. Dabei ist e_i der i-te Einheitsvektor.
Die Ergebnisse sind in den Tabellen 5.1 bis 5.6, eine Tabelle per untersuchter Richtung, zu sehen. In der ersten Spalte der Tabellen ist der Name des Problems angegeben. Nach jeweils fünf Problemen werden in der nächsten Zeile die Durchschnittswerte gezeigt, angedeutet durch das $\oslash$ vor dem Namen. In den Spalten 2 und 3 stehen die Dimension der Matrixvariablen n und die Anzahl der Nebenbedingungen m. Dann folgt, in der nächsten Spalte, die Anzahl der durchlaufenen Iterationen. Die 5. Spalte zeigt das abschließende $\mu = \langle X, Z \rangle / n$. Es folgen der primale und der duale Zielfunktionswert in den Spalten 6

und 7 und die primale und duale Unzulässigkeit in den Spalten 8 und 9. Die letzte Spalte gibt die für den Durchlauf benötigte CPU-Zeit in Sekunden an. In Tabelle 5.7 sind zur Übersicht die durchschnittlichen CPU-Zeiten der verschieden Suchrichtungen noch einmal zusammengetragen.
Man sieht in Tabelle 5.7, daß die HKM-Richtung die MAXCUT-Probleme deutlich am schnellsten löst und sich dort die Prediktor-Korrektor-Variante zeitlich in keiner Weise auszahlt, obwohl sie weniger Iterationen benötigt. Die P-K-Version der NT-Richtung braucht trotz weniger Iterationen ungefähr die gleiche Zeit, wie ihre „normale“ Version. Im Gegensatz dazu bringt die P-K-Variante der AHO-Richtung deutlich bessere Ergebnisse als die „normalen“ Version. Die AHO-Richtung ist allerdings auch die deutlich langsamste der drei Richtungen.
Bei dem Problem *mc200_01* brach die P-K-Version der AHO-Richtung wegen zu kurzer Schrittweiten ab (siehe †in Tabelle 5.6), es wurde deshalb bei der Durchschnittsberechnung außer acht gelassen (‡).
Normalerweise ist das Aufstellen der Systemmatrix mit einer Komplexität von $O(m^2n^2 + mn^3)$ der teuerste Schritt des Algorithmus (siehe [32]). Dies läßt sich bei den Ergebnissen der AHO-Richtung deutlich erkennen. Bei den anderen beiden Richtungen kann man eine kubische Abhängigkeit beobachten. Durch die angenehme Struktur der Nebenbedingungen tritt das Aufstellen der Systemmatrix in den Hintergrund und der Algorithmus wird von den Faktorisierungen der Systemmatrix und der Liniensuche (mit Komplexität $O(n^3)$) dominiert (siehe Tabelle 5.7).

Tabelle 5.1: **Maxcut-Probleme zufälliger Graphen mit HKM**

Problem	n	m	Iter.	μ	$\langle C, X\rangle$	$b^T y$	prim.Unz.	duale Unz.	CPU sek.
mc50_01	50	50	14	4.47e-06	380.993	380.993	4.7e-11	0	3.27
mc50_02	50	50	13	1.21e-06	382.899	382.899	8.3e-11	0	3.04
mc50_03	50	50	13	5.46e-06	383.269	383.269	7.3e-11	0	3.05
mc50_04	50	50	16	7.22e-06	385.136	385.136	1.4e-10	0	3.75
mc50_05	50	50	16	2.53e-06	386.755	386.756	7e-11	0	3.75
⊘mc50	50	50	14.4	-	-	-	-	-	3.37
mc100_01	100	100	15	1.03e-05	1463.32	1463.32	9.1e-11	0	25.10
mc100_02	100	100	15	2.34e-06	1467.95	1467.95	2.2e-10	0	26.78
mc100_03	100	100	17	4.77e-06	1457.86	1457.86	5.9e-11	0	32.79
mc100_04	100	100	15	3.94e-06	1460.41	1460.41	6.8e-11	0	26.60
mc100_05	100	100	17	6.16e-06	1461.03	1461.03	7.3e-10	0	29.59
⊘mc100	100	100	15.8	-	-	-	-	-	28.17
mc200_01	200	200	17	2.08e-05	5631.62	5631.63	4.1e-11	0	217.00
mc200_02	200	200	18	8.62e-06	5622.94	5622.94	1.2e-10	0	239.48
mc200_03	200	200	17	4.06e-06	5628.61	5628.61	1.8e-10	0	236.46
mc200_04	200	200	17	2.15e-06	3574.17	3574.17	9.8e-10	0	222.58
mc200_05	200	200	16	1.36e-05	3574.70	3574.71	9.5e-11	0	202.70
⊘mc200	200	200	17	-	-	-	-	-	223.64

Tabelle 5.2: **Maxcut-Probleme zufälliger Graphen mit HKMPC**

Problem	n	m	Iter.	μ	$\langle C, X\rangle$	$b^T y$	prim.Unz.	duale Unz.	CPU sek.
mc50_01	50	50	10	1.17e-06	380.993	380.993	6.2e-11	0	6.00
mc50_02	50	50	10	8.38e-07	382.899	382.899	1e-10	0	6.00
mc50_03	50	50	10	1.90e-06	383.269	383.269	1.1e-10	0	5.94
mc50_04	50	50	11	7.25e-06	385.136	385.136	1.8e-11	0	6.88
mc50_05	50	50	10	6.83e-06	386.755	386.756	1.1e-10	0	5.97
⊘mc50	50	50	10.2	-	-	-	-	-	6.16
mc100_01	100	100	10	2.38e-06	1463.32	1463.32	1.3e-10	0	47.46
mc100_02	100	100	10	7.44e-06	1467.95	1467.95	6.7e-11	0	49.83
mc100_03	100	100	10	3.94e-06	1457.86	1457.86	1.2e-10	0	53.60
mc100_04	100	100	10	3.07e-06	1460.41	1460.41	7e-11	0	48.30
mc100_05	100	100	10	1.08e-05	1461.03	1461.03	2.8e-11	0	48.31
⊘mc100	100	100	10	-	-	-	-	-	49.5
mc200_01	200	200	12	3.04e-06	5631.63	5631.63	3.2e-10	0	429.68
mc200_02	200	200	11	8.06e-06	5622.94	5622.94	1.2e-10	0	420.07
mc200_03	200	200	10	1.75e-05	5628.61	5628.61	4.7e-11	0	388.26
mc200_04	200	200	12	1.67e-06	3574.17	3574.17	8.8e-10	0	460.21
mc200_05	200	200	11	9.34e-06	3574.70	3574.71	7.1e-11	0	395.42
⊘mc200	200	200	11.2	-	-	-	-	-	418.73

Tabelle 5.3: **Maxcut-Probleme zufälliger Graphen mit NT**

Problem	n	m	Iter.	μ	$\langle C, X\rangle$	$b^T y$	prim.Unz.	duale Unz.	CPU sek.
mc50_01	50	50	14	9.45e-07	380.993	380.993	2.7e-10	0	26.08
mc50_02	50	50	13	1.21e-06	382.899	382.899	1.8e-10	0	24.65
mc50_03	50	50	13	1.55e-06	383.269	383.269	1.4e-10	0	24.64
mc50_04	50	50	16	3.41e-06	385.136	385.136	2.9e-10	0	29.36
mc50_05	50	50	16	2.52e-06	386.755	386.756	8.9e-11	0	29.13
⊘mc50	50	50	14.4	-	-	-	-	-	26.77
mc100_01	100	100	15	5.39e-06	1463.32	1463.32	9.5e-11	0	228.79
mc100_02	100	100	15	2.35e-06	1467.95	1467.95	2.2e-10	0	229.81
mc100_03	100	100	15	1.42e-05	1457.86	1457.86	2.8e-11	0	241.47
mc100_04	100	100	15	3.94e-06	1460.41	1460.41	9.8e-11	0	227.78
mc100_05	100	100	15	1.39e-05	1461.03	1461.03	5.4e-11	0	242.97
⊘mc100	100	100	15	-	-	-	-	-	234.16
mc200_01	200	200	17	4.78e-06	5631.63	5631.63	2.4e-10	0	1953.36
mc200_02	200	200	17	7.04e-06	5622.94	5622.94	2.1e-10	0	1994.57
mc200_03	200	200	16	5.56e-06	5628.61	5628.61	2.3e-10	0	1858.32
mc200_04	200	200	15	1.33e-05	3574.17	3574.17	1.2e-10	0	1786.59
mc200_05	200	200	16	3.40e-06	3574.70	3574.71	4.2e-10	0	1845.11
⊘mc200	200	200	16.2	-	-	-	-	-	1887.59

Tabelle 5.4: **Maxcut-Probleme zufälliger Graphen mit NTPC**

Problem	n	m	Iter.	μ	$\langle C, X\rangle$	$b^T y$	prim.Unz.	duale Unz.	CPU sek.
mc50_01	50	50	10	1.21e-06	380.993	380.993	1.3e-10	0	23.32
mc50_02	50	50	10	1.47e-06	382.899	382.899	1.4e-10	0	23.44
mc50_03	50	50	11	5.70e-06	383.269	383.269	7.4e-12	1.7e-14	24.98
mc50_04	50	50	12	1.43e-06	385.136	385.136	1.7e-10	0	26.92
mc50_05	50	50	11	1.55e-06	386.755	386.756	7.3e-11	0	25.07
⊘mc50	50	50	10.8	-	-	-	-	-	24.75
mc100_01	100	100	11	1.30e-05	1463.32	1463.32	6.5e-12	4.2e-14	205.94
mc100_02	100	100	11	1.06e-05	1467.95	1467.95	1.2e-11	4.3e-14	212.55
mc100_03	100	100	11	2.70e-06	1457.86	1457.86	2e-10	0	221.70
mc100_04	100	100	12	4.06e-06	1460.41	1460.41	2.8e-11	2.9e-14	227.36
mc100_05	100	100	11	8.90e-06	1461.03	1461.03	3.6e-11	3.9e-14	235.31
⊘mc100	100	100	11.2	-	-	-	-	-	220.57
mc200_01	200	200	12	6.79e-06	5631.63	5631.63	4e-11	8.3e-14	2040.15
mc200_02	200	200	12	6.72e-06	5622.94	5622.94	3.2e-11	1e-13	2197.66
mc200_03	200	200	11	2.67e-05	5628.61	5628.62	1.4e-11	9.8e-14	1595.89
mc200_04	200	200	11	1.14e-05	3574.17	3574.17	4.1e-10	0	2007.26
mc200_05	200	200	12	2.51e-06	3574.70	3574.71	3.2e-10	0	2006.60
⊘mc200	200	200	11.6	-	-	-	-	-	1969.51

Tabelle 5.5: **Maxcut-Probleme zufälliger Graphen mit AHO**

Problem	n	m	Iter.	μ	$\langle C, X\rangle$	$b^T y$	prim.Unz.	duale Unz.	CPU sek.
mc50_01	50	50	14	1.40e-06	380.993	380.993	1.2e-10	0	54.45
mc50_02	50	50	13	6.96e-06	382.899	382.899	2.4e-11	0	52.60
mc50_03	50	50	13	1.33e-06	383.269	383.269	2.1e-10	0	51.92
mc50_04	50	50	16	3.32e-06	385.136	385.136	2.8e-10	1.8e-14	62.54
mc50_05	50	50	16	2.50e-06	386.755	386.756	4e-11	0	65.12
⊘mc50	50	50	14.5	-	-	-	-	-	57.33
mc100_01	100	100	15	3.82e-06	1463.32	1463.32	1.1e-10	0	832.41
mc100_02	100	100	15	3.01e-06	1467.95	1467.95	8.9e-11	0	1175.84
mc100_03	100	100	15	1.13e-05	1457.86	1457.86	5.1e-11	0	869.72
mc100_04	100	100	15	4.65e-06	1460.41	1460.41	6.8e-11	0	970.72
mc100_05	100	100	15	4.06e-06	1461.03	1461.03	1.5e-10	0	954.46
⊘mc100	100	100	15	-	-	-	-	-	960.63
mc200_01	200	200	16	1.74e-05	5631.62	5631.63	5.9e-11	0	16339.54
mc200_02	200	200	16	1.52e-05	5622.94	5622.94	1e-10	0	14972.33
mc200_03	200	200	16	4.45e-06	5628.61	5628.61	4.3e-10	0	15506.90
mc200_04	200	200	16	4.29e-06	3574.17	3574.17	3.5e-10	0	15611.55
mc200_05	200	200	15	1.55e-05	3574.70	3574.71	7.3e-11	0	14460.90
⊘mc200	200	200	15.8	-	-	-	-	-	15378.24

Tabelle 5.6: **Maxcut-Probleme zufälliger Graphen mit AHOPC**

Problem	n	m	Iter.	μ	$\langle C,X\rangle$	$b^T y$	prim.Unz.	duale Unz.	CPU sek.
mc50_01	50	50	10	1.19e-07	380.993	380.993	1.7e-10	0	47.15
mc50_02	50	50	9	5.04e-07	382.899	382.899	1.4e-10	0	42.79
mc50_03	50	50	8	2.48e-06	383.269	383.269	2.3e-11	0	38.58
mc50_04	50	50	10	2.02e-06	385.136	385.136	1.7e-10	2e-14	46.33
mc50_05	50	50	12	2.75e-07	386.756	386.756	6.5e-11	0	55.00
⊘mc50	50	50	9.8	-	-	-	-	-	45.97
mc100_01	100	100	10	3.43e-07	1463.32	1463.32	2.9e-10	0	633.83
mc100_02	100	100	8	4.95e-06	1467.95	1467.95	8.8e-12	0	537.69
mc100_03	100	100	9	1.98e-06	1457.86	1457.86	3.7e-10	0	575.58
mc100_04	100	100	8	3.98e-06	1460.41	1460.41	1.5e-11	0	670.29
mc100_05	100	100	10	2.80e-07	1461.03	1461.03	3.1e-09	0	767.08
⊘mc100	100	100	9	-	-	-	-	-	636.89
mc200_01	200	200	28†	0.5455	5511.76	5655.88	0.14	1.8e-13	28687.71
mc200_02	200	200	10	6.95e-07	5622.94	5622.94	5.2e-10	0	12311.01
mc200_03	200	200	9	2.62e-06	5628.61	5628.61	7.9e-11	0	8823.85
mc200_04	200	200	9	1.36e-05	3574.17	3574.17	1.3e-12	0	9540.54
mc200_05	200	200	10	5.04e-07	3574.71	3574.71	3.8e-10	0	10386.92
⊘mc200‡	200	200	9.5	-	-	-	-	-	10265.58

Tabelle 5.7: **Vergleich der Richtungen bei MAXCUT-Problemen**

	CPU Zeit in Sekunden					
Problem	HKM	HKMPC	NT	NTPC	AHO	AHOPC
⊘mc50	3.37	6.16	26.77	24.75	57.33	45.97
⊘mc100	28.17	49.5	234.16	220.57	960.63	636.89
⊘mc200	223.64	418.73	1887.59	1969.51	15378.24	10265.58

5.2 Probleme aus der SDPLIB

Die Implementation wurde an einigen ausgewählten Problemen der SDPLIB [4] getestet. Dazu mußten sie zunächst aus dem SDPA-Sparse-Format in das benötigte Dateiformat konvertiert werden. Hierbei wurde für die Matrizen der Nebenbedingungen die dünn besetzte Struktur *Sparscons* gewählt.
Die Ergebnisse wurden in den Tabellen 5.8 bis 5.13 zusammengetragen. Für eine Beschreibung der Tabellen siehe Abschnitt 5.1.
Auch bei diesem Test war die HKM-Richtung die deutlich schnellste, allerdings hat sich diesmal die Prediktor-Korrektor-Version bei Problemen mit deutlich mehr Nebenbedingungen m als die Ordnung der Variablen n ausgezahlt. Die Probleme *qap5* und *qap6* wurden nur von der AHO-Richtung zur Zufriedenheit gelöst. Die beiden anderen Richtungen brachen ab, da sich die Kondition der Systemmatrix so stark verschlechtert hatte, daß sie nicht mehr faktorisiert werden konnte (siehe * in Tabellen 5.8 bis 5.11). Bei diesen Problemen scheint der Algorithmus mit der AHO-Richtung stabiler als mit den anderen Richtungen zu sein. Allerdings hatte die AHO-Richtung bei *qap6* einige Probleme mit primaler Unzulässigkeit (§in Tabelle 5.12 und 5.13). Auch beim Problem *control1* hat die AHO-Richtung ungewöhnlich viele Iterationen benötigt (#).

Tabelle 5.8: **Probleme aus der SDPLIB mit HKM**

Problem	n	m	Iter.	μ	$\langle C, X\rangle$	$b^T y$	prim.Unz.	duale Unz.	CPU sek.
truss1	13	6	13	4.51e-07	-9.00000	-9.00000	1.5e-09	0	0.14
truss4	19	12	15	1.41e-07	-9.01000	-9.01000	2.9e-09	0	0.26
control1	15	21	23	1.50e-07	17.7800	17.7800	6.8e-07	0	0.60
theta1	50	104	24	3.03e-07	23.0000	23.0000	4.3e-10	3e-16	93.3
theta2	100	498	27	1.65e-07	32.8792	32.8792	1.2e-12	9.5e-16	4152.36
qap5	26	136	13*	0.1045	-438.220	-435.904	9.5e-05	0	11.64
qap6	37	229	15*	0.0167	-381.747	-381.216	1.5e-05	1e-10	59.32

Tabelle 5.9: **Probleme aus der SDPLIB mit HKMPC**

Problem	n	m	Iter.	μ	$\langle C, X\rangle$	$b^T y$	prim.Unz.	duale Unz.	CPU sek.
truss1	13	6	11	4.66e-07	-9.00000	-9.00000	3e-10	0	0.18
truss4	19	12	13	3.22e-07	-9.01000	-9.01000	2.1e-09	0	0.28
control1	15	21	17	8.05e-07	17.7800	17.7800	1.2e-07	0	0.61
theta1	50	104	12	4.05e-07	23.0000	23.0000	1.8e-09	2.7e-16	45.36
theta2	100	498	12	1.33e-07	32.8792	32.8792	1e-10	1.2e-15	1698.13
qap5	26	136	14*	0.000627	-436.013	-436.000	1.9e-05	0	13.68
qap6	37	229	14*	0.01005	-381.610	-381.335	0.00012	3.3e-11	56.24

Tabelle 5.10: **Probleme aus der SDPLIB mit NT**

Problem	n	m	Iter.	μ	$\langle C, X\rangle$	$b^T y$	prim.Unz.	duale Unz.	CPU sek.
truss1	13	6	12	1.22e-07	-9.00000	-9.00000	4.2e-09	0	0.17
truss4	19	12	15	1.00e-07	-9.01000	-9.01000	1.4e-08	0	0.35
control1	15	21	23	3.20e-07	17.7800	17.7800	5e-07	0	1.06
theta1	50	104	24	9.4e-08	23.0000	23.0000	2.1e-10	3.4e-16	131.15
theta2	100	498	27	7.18e-08	32.8792	32.8792	1e-09	8.6e-16	5121.12
qap5	26	136	13*	0.10513	-438.237	-435.908	9.5e-05	0	14.54
qap6	37	229	14*	0.03814	-382.119	-380.858	3.7e-05	1.2e-10	67.28

Tabelle 5.11: **Probleme aus der SDPLIB mit NTPC**

Problem	n	m	Iter.	μ	$\langle C, X\rangle$	$b^T y$	prim.Unz.	duale Unz.	CPU sek.
truss1	13	6	11	1.98e-07	-9.00000	-9.00000	7.7e-09	0	0.24
truss4	19	12	13	3.92e-07	-9.01000	-9.01000	2.9e-09	0	0.46
control1	15	21	17	1.63e-07	17.7800	17.7800	3.8e-07	0	0.99
theta1	50	104	11	2.08e-07	23.0000	23.0000	1.3e-11	3.3e-16	67.86
theta2	100	498	13	1.68e-07	32.8792	32.8792	3.3e-10	1.9e-14	2782.96
qap5	26	136	15*	0.000133	-436.003	-436.000	3.2e-06	0	17.80
qap6	37	229	14*	0.00342	-381.485	-381.395	3e-05	0	68.33

Tabelle 5.12: **Probleme aus der SDPLIB mit AHO**

Problem	n	m	Iter.	μ	$\langle C, X\rangle$	$b^T y$	prim.Unz.	duale Unz.	CPU sek.
truss1	13	6	14	3.37e-07	-9.00000	-9.00000	4.3e-08	0	0.43
truss4	19	12	16	7.42e-08	-9.01000	-9.01000	3.5e-08	0	0.72
control1	15	21	80#	5.13e-07	17.7800	17.7800	9.8e-06	8.9e-11	6.54
theta1	50	104	21	1.59e-07	23.0000	23.0000	1.1e-09	2.6e-16	405.34
theta2	100	498	25	2.26e-07	32.8791	32.8792	1.4e-10	7.6e-16	35062.03
qap5	26	136	16	5.87e-06	-436.000	-436.000	3.2e-06	0	104.55
qap6	37	229	20	5.07e-06	-381.124	-381.294	0.00012§	6.1e-12	797.39

Tabelle 5.13: **Probleme aus der SDPLIB mit AHOPC**

Problem	n	m	Iter.	μ	$\langle C, X\rangle$	$b^T y$	prim.Unz.	duale Unz.	CPU sek.
truss1	13	6	10	2.63e-08	-9.00000	-9.00000	5.3e-08	0	0.45
truss4	19	12	12	1.12e-08	-9.01000	-9.01000	2.1e-08	0	0.66
control1	15	21	54#	5.25e-07	17.7800	17.7800	4.9e-04	0	5.63
theta1	50	104	10	8.90e-08	23.0000	23.0000	8.8e-08	2.7e-16	211.03
theta2	100	498	11	9.58e-09	32.8792	32.8792	3.9e-11	1.1e-15	18785.72
qap5	26	136	10	7.19e-06	-436.000	-436.000	2.7e-07	0	71.56
qap6	37	229	36	3.84e-06	-380.622	-381.077	0.0002§	0	1571.06

5.3 Vergleich mit SDPT3

Der Vergleich unserer Implementation mit dem MATLAB-Software-Paket SDPT3 ist in den Tabellen 5.14 bis 5.19 dargestellt. Jede Tabelle steht für eine der Suchrichtungen. In der ersten Spalte steht der Name des Problems, wobei bei den MAXCUT-Problemen die Durchschnittswerte von fünf Problemen angegeben sind, da nicht exakt die selben Probleme, sondern nur auf die gleiche Art generierte Probleme, getestet wurden. Die nächsten beiden Spalten geben die Anzahl der, zum Lösen der Problem, benötigten Iterationen unserer Implementation respektive von SDPT3 an. In den Spalten 4 und 5 steht die dafür benötigte CPU-Zeit in Sekunden. In den letzten beiden Spalten ist schließlich die durchschnittliche Zeit per Iteration angegeben.
Beim Vergleich hat sich ergeben, daß bei kleineren MAXCUT-Problemen die HKM- und die AHO-Richtung dieser Implementation deutlich schneller ist, was bei größeren Problemen nicht mehr der Fall ist. Dies mag eventuell darin begründet sein, daß MATLAB auf systemspezifische Routinen zur Berechnung großer Matrixoperationen zurückgreift, während die unserem Paket verwendeten Routinen durchweg selbst implementiert sind und auf keine systemabhängigen Komponenten zurückgreifen. Vermutlich sollte auch in unserer Implementation auf systemspezifische Routinen (z.B. BLAS siehe [14]) zurückgegriffen werden. Bei weiteren Tests hat sich bestätigt, daß das Faktorisieren unter MATLAB wesentlich schneller ist. Auch beim Aufstellen der Systemmatrix M ist SDPT3 schneller als unsere Implementation, wenn die Nebenbedingungen nicht, wie bei MAXCUT-Problemen eine sehr einfache Struktur haben.
Die NT-Richtung von SDPT3 ist, außer bei den eher kleinen Problemen truss1, truss4, control1 und qap5, deutlich besser als die unserer Implementation (siehe Tabelle 5.16 und Tabelle 5.17). Das deutet darauf hin, daß Toh, Todd und Tütüncü sie besser implementiert haben als es hier getan wurde. In dieser Hinsicht kann die Implementation noch verbessert werden.

Tabelle 5.14: **Vergleich dieser Implementation mit SDPT3 (HKM)**

	Anzahl der Iter.		CPU Zeit		CPU Zeit per Iter.	
Problem	Impl.	SDPT3	Impl.	SDPT3	Impl.	SDPT3
⌀mc50	14.4	16.6	3.37	11.6	0.23	0.70
⌀mc100	15.8	19	28.17	53.54	1.78	2.82
truss1	13	14	0.14	6.2	0.011	0.44
truss4	15	11	0.26	6.2	0.017	0.56
control1	23	22	0.60	16.5	0.04	0.75
theta1	24	23	93.3	25.2	3.89	1.10
theta2	27	27	4152.36	514.4	153.8	19.05
qap5	13*	15*	11.64	21.1	0.90	1.41
qap6	15*	19	59.32	55.5	3.95	2.92

Tabelle 5.15: **Vergleich dieser Implementation mit SDPT3 (HKMPC)**

Problem	Anzahl der Iter.		CPU Zeit		CPU Zeit per Iter.	
	Impl.	SDPT3	Impl.	SDPT3	Impl.	SDPT3
⌀mc50	10.2	10	6.16	11.3	0.6	1.13
⌀mc100	10	10.4	49.5	47.54	4.95	4.57
truss1	11	8	0.18	5.3	0.016	0.66
truss4	13	9	0.28	7.2	0.021	0.8
control1	17	16	0.61	15.2	0.036	0.95
theta1	12	10	45.36	15.8	3.78	1.58
theta2	12	11	1698.13	252.3	141.5	22.94
qap5	14*	10	13.68	16.6	0.98	1.66
qap6	14*	14	56.24	47.9	4.02	3.42

Tabelle 5.16: **Vergleich dieser Implementation mit SDPT3 (NT)**

Problem	Anzahl der Iter.		CPU Zeit		CPU Zeit per Iter.	
	Impl.	SDPT3	Impl.	SDPT3	Impl.	SDPT3
⌀mc50	14.4	15.6	26.77	16.2	1.86	1.04
⌀mc100	15	16	234.16	72.34	15.6	4.52
truss1	12	12	0.17	7.7	0.014	0.64
truss4	15	11	0.35	8.7	0.023	0.79
control1	23	22	1.06	18.5	0.046	0.84
theta1	24	22	131.15	32.2	5.46	1.46
theta2	27	25	5121.12	535.5	189.7	21.42
qap5	13*	17	14.54	25.3	0.35	1.49
qap6	13*	19	67.28	58.3	4.81	3.07

Tabelle 5.17: **Vergleich dieser Implementation mit SDPT3 (NTPC)**

Problem	Anzahl der Iter.		CPU Zeit		CPU Zeit per Iter.	
	Impl.	SDPT3	Impl.	SDPT3	Impl.	SDPT3
⌀mc50	10.8	10	24.75	16.36	2.29	1.64
⌀mc100	11.2	10.6	220.57	73.14	19.7	6.9
truss1	11	8	0.24	7.9	0.022	0.99
truss4	13	9	0.46	10.5	0.035	1.17
control1	17	16	0.99	18.4	0.058	1.15
theta1	11	11	67.86	22.9	6.17	2.08
theta2	13	11	2782.96	285.8	214.1	25.98
qap5	15*	10	17.80	18.9	1.19	1.89
qap6	14*	14	68.33	51.6	4.88	3.69

Tabelle 5.18: **Vergleich dieser Implementation mit SDPT3 (AHO)**

Problem	Anzahl der Iter.		CPU Zeit		CPU Zeit per Iter.	
	Impl.	SDPT3	Impl.	SDPT3	Impl.	SDPT3
⊘mc50	14.5	14	57.33	98.0	3.95	7.0
⊘mc100	15	15.2	960.63	929.46	64	61.15
truss1	14	11	0.43	9.8	0.03	0.89
truss4	16	11	0.72	14.7	0.045	1.34
control1	80	21	6.54	31.0	0.082	1.48
theta1	21	21	405.34	338.2	19.30	16.10
theta2	25	23	35062	10016.8	1402.5	435.51
qap5	16	17	104.55	172.8	6.53	10.16
qap6	20	19	797.39	470.8	39.87	24.78

Tabelle 5.19: **Vergleich dieser Implementation mit SDPT3 (AHOPC)**

Problem	Anzahl der Iter.		CPU Zeit		CPU Zeit per Iter.	
	Impl.	SDPT3	Impl.	SDPT3	Impl.	SDPT3
⊘mc50	9.8	9	45.97	67.06	4.69	7.45
⊘mc100	9	9	636.89	639.2	70.8	71.0
truss1	10	8	0.45	10.5	0.045	1.31
truss4	12	9	0.66	15.7	0.055	1.74
control1	54	16	5.63	28.8	0.10	1.80
theta1	10	10	211.03	166.4	21.10	16.64
theta2	11	11	18785.72	5895.7	1707.8	535.97
qap5	10	10	71.56	106.7	7.16	10.67
qap6	36	14	1571.06	350.0	43.64	25

Kapitel 6

Zusammenfassung

Abschließend ist zu sagen, daß an der Implementation wohl noch einiges zu verbessern ist. Zum einen ist das Einbauen von systemspezifischen Routinen anzuraten, um Dinge wie das Faktorisieren von Matrizen zu beschleunigen. Auch in Hinsicht der numerischen Stabilität könnten noch Anstrengungen unternommen werden.Außerdem wurde die schiefsymmetrische Einbettung (siehe Abschnitt 2.5) noch nicht implementiert und getestet.
Die Vorteile der Implementation liegen sicherlich in der guten Ausnutzung der Strukturen der Nebenbedingungen und der leichten Erweiterbarkeit, die das Hinzufügen von neuen Suchrichtungen und Nebenbedingungsstrukturen einfach macht.

Literaturverzeichnis

[1] Farid Alizadeh. *Combinatorial Optimization With Interior Point Methods And Semidefinite Matrices*. PhD thesis, Computer Science Department, University of Minnesota, 1991.

[2] Farid Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM J. Optim.*, 5(1):13–51, 1995.

[3] Farid Alizadeh, Jean-Pierre A. Haeberly, and Michael L. Overton. Primal-dual interior-point methods for semidefinite programming: Convergence rates, stability and numerical results. *SIAM J. Optim.*, 8(3):746–768, August 1998.

[4] Brian Borchers. CSDP, a C library for semidefinite programming. Technical report, May 1998.

[5] Etienne de Klerk, Cees Roos, and Támas Terlaky. Infeasible start, semidefinite programming algorithms via self-dual embeddings. Report 97-10, Faculty of Technical Mathematics and Informatics, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands, April 1997.

[6] Etienne de Klerk, Cees Roos, and Tamás Terlaky. Initialization in semidefinite programming via a self-dual skew-symmetric embedding. *Operations Research Letters*, 20:213–221, 1997.

[7] Gene H. Golub and Charles F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, 2^{nd} edition, 1989.

[8] Alexander Graham. *Kronecker Products and Matrix Calculus With Applications*. Mathematics and its Applications. Ellis Horwood Limited, Chichester, 1981.

[9] Christoph Helmberg, Franz Rendl, R. J. Vanderbei, and Henry Wolkowicz. An interior–point method for semidefinite programming. *SIAM J. Optim.*, 6(2):342–361, May 1996.

[10] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[11] Florian Jarre. An interior–point method for minimizing the maximum eigenvalue of a linear combination of matrices. *Siam J. Control and Optimization*, 31(5):1360–1377, September 1993.

[12] Masakazu Kojima, Masayuki Shida, and Susumu Shindoh. Local convergence of predictor-corrector infeasible-interior-point algorithms for SDPs and SDLCPs. *Math. Programming*, 80(2):129–160, January 1998.

[13] Masakazu Kojima, Susumu Shindoh, and Shinji Hara. Interior–point methods for the monotone linear complementarity problem in symmetric matrices. *SIAM J. Optim.*, 7(1):86–125, February 1997.

[14] C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh. Basic linear algebra subprograms for fortran usage. *ACM Trans. Math. Soft.*, 5:308–323, 1979.

[15] Chih-Jen Lin and Romesh Saigal. An infeasible start predictor corrector method for semi-definite linear programming. Technical report, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, Michigan 48109-2117, December 1995.

[16] Chih-Jen Lin and Romesh Saigal. A predictor-corrector method for semidefinite linear programming. Technical report, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, Michigan 48109-2117, October 1995. Revised December 1995.

[17] S. Mehrotra. On the implementation of a primal–dual interior point method. *SIAM J. Optim.*, 2:575–601, 1992.

[18] Renato D. C. Monteiro. Primal–dual path-following algorithms for semidefinite programming. *SIAM J. Optim.*, 7(3):663–678, August 1997.

[19] Renato D. C. Monteiro. Polynomial convergence of primal-dual algorithms for semidefinite programming based on Monteiro and Zhang family of directions. *SIAM J. Optim.*, 8(3):797–812, August 1998.

[20] Renato D. C. Monteiro and Takashi Tsuchiya. Polynomial convergence of a new family of primal-dual algorithms for semidefinite programming. *SIAM J. Optim.*, 9(3):551–577, 1999.

[21] Renato D. C. Monteiro and Yin Zhang. A unified analysis for a class of path-following primal-dual interior-point algorithms for semidefinite programming. *Math. Programming*, 81(3):281–299, May 1998.

[22] Yurii Nesterov and Arkadii Nemirovskii. *Interior–Point Polynomial Algorithms in Convex Programming.* SIAM Studies in Applied Mathematics, Philadelphia, 1994.

[23] Yurii Nesterov and Michael J. Todd. Primal-dual interior-point methods for self-scaled cones. Technical Report TR 1125, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York 14853, August 1995. To appear in SIAM J. Optim.

[24] Yurii Nesterov and Michael J. Todd. Self-scaled barriers and interior-point methods for convex programming. *Math. Oper. Res.*, 22:1–42, 1997.

[25] Florian A. Potra and Rongqin Sheng. Homogenous interior-point algorithms for semidefinite programming. Reports on Computational Mathematics 82, Department of Mathematics, University of Iowa, Iowa City, IA 52242, November 1995.

[26] Florian A. Potra and Rongqin Sheng. A superlinearly convergent primal-dual infeasible-interior-point algorithm for semidefinite programming. Reports on Computational Mathematics 78, Department of Mathematics, University of Iowa, Iowa City, IA 52242, October 1995.

[27] Florian A. Potra and Rongqin Sheng. Superlinear convergence of a predictor-corrector method for semidefinite programming without shrinking central path neighborhood. Reports on Computational Mathematics 91, Department of Mathematics, University of Iowa, Iowa City, IA 52242, August 1996.

[28] Florian A. Potra and Rongqin Sheng. Superlinear convergence of interior-point algorithms for semidefinite programming. Reports on Computational Mathematics 86, Department of Mathematics, University of Iowa, Iowa City, IA 52242, April 1996. Revised May 1996.

[29] R. Tyrrell Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, New Jersey, 1970.

[30] Masayuki Shida, Susumu Shindoh, and Masakazu Kojima. Existence of search directions in interior-point algorithms for the SDP and the monotone SDLCP. Research Report B–310, Department of Information Sciences, Tokyo Institute of Technology, January 1996.

[31] Jos F. Sturm and Shuzhong Zhang. Symmetric primal-dual path following algorithms for semidefinite programming. Report 9554/A, Econometric Institute, Erasmus University of Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands, 1995. Revised February 1996.

[32] Michael J. Todd, K. C. Toh, and R. H. Tütüncü. On the Nesterov-Todd direction in semidefinite programming. *SIAM J. Optim.*, 8(3):769–796, August 1998.

[33] Mike Todd. On search directions in interior-point methods for semidefinite programming. Technical Report TR1205, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853-3801, October 1997.

[34] K. C. Toh, Michael J. Todd, and R. H. Tütüncü. SDPT3 — a MATLAB software package for semidefinite programming. Technical report, Department of Mathematics, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260, August 1998.

[35] Lieven Vandenberghe and Stephen Boyd. A primal–dual potential reduction method for problems involving matrix inequalities. *Mathematical Programming, Series B*, 69(1):205–236, 1995.

[36] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, March 1996.

[37] Henry Wolkowicz. Some applications of optimization in matrix theory. *Linear Algebra and its Applications*, 40:101–118, 1981.

[38] Xiaojie Xu, Pi-Fang Hung, and Yinyu Ye. A simplified homogenous and self-dual linear programming algorithm and its implementation. *Annals of Operations Research*, 62:151–171, 1996.

[39] Yinyu Ye. A class of projective transformations for linear programming. *SICOM*, 19:457–466, 1990.

[40] Yin Zhang. On extending some primal–dual interior-point algorithms from linear programming to semidefinite programming. *SIAM J. Optim.*, 8(2):365–386, 1998.