Otto-von-Guericke-Universität Magdeburg

Fakultät für Elektrotechnik und Informationstechnik
Institut für Automatisierungstechnik

Masterarbeit

Meine Master Thesis

Heinemann, Hannes
22. Juli 2015

Erstprüfer:	Rolf Findeisen
Zweitprüfer:	Erik B.
Betreuer:	Pablo

Aufgabenstellung

Thema

Zeitraum: 06 - 11

Das ist meine Aufgabenstellung

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

MPC ..... Modellprädiktive Regelung

Symbolverzeichnis

1 Einleitung

1.1 TODO

1.1.1 Format

- Zitierstil
- Papierformat
- Schriftgröße, Schriftart ...
- Zeilenabstand

1.1.2 Inhalt

- Beispiele helfen um den Algorithmus und das Bilden der Matrizen zu verstehen, falls noch Seiten gefüllt werden müssen ;)
- "Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization" hier findet man in kapitel 5 ein paar hinweise zu der Regularization
- "Primal Barrier Methods For Linear Programming" Kapitel 3: schwierigkeiten bei der Wahl des StartKappa und Startwerte, wie man es günstig wählt Bei einem guten Startwert, kann der Barrierparameter entsprechend klein gewählt werden, ohne das Probleme bei der Konvergenz gegen ein Optimum auftreten + S.29 Tested kappa [0.0001, 1] multiplied with (c'x)/n Value of Costfunction durch dimension to have the value of barrier function in same order as the costvalue + lienear modifications to the barrier function + where to stop + The Nullspace Methods + Hinweise zu Cholesky factorization

- Ausprobieren Kappa kleiner zu wählen, wenn in ermittelter Suchrichtung keine Verbesserung des Funktionswertes möglich ist
- Tabelle mit den Ergebnissen der gelösten Tests

[MM97] Modellprädiktive Regelung (MPC)

2 Grundlagen

- generalized inequalities
- Originalalgorithmus von Wang und Boyd [WB10]

Tääast

3 Algorithmus

3.1 Erweiterung für nichtlineare Constraints

Um mit dem effizienten Algorithmus aus [WB10] auch kompliziertere/komplexere [TODO] Probleme, in Kapitel 3.1.2 Second Order Cone Problems (SOCP), bzw in Kapitel 3.1.1 Quadratic Programs/Problems mit Quadratic Constraints (QCQP) zu lösen, wurde der Algorithmus wie gleich folgt erweitert. Dabei wurde speziell darauf geachtet nicht die Struktur der entstehenden Matrizen zu verändern, sodass diese auch weiterhin ausgenutzt werden kann. Allerdings sind für ein SOCP bzw QCQP nun die Matrizen für die Ungleichungsnebenbedingungen nicht mehr konstant sondern hängen von $z(k)$ ab, sodass sie in jedem MPC Schritt angepasst werden müssen, was einen erhöhten Rechenaufwand bedeutet.

3.1.1 Erweiterung für Quadratic Problems mit quadratic Constraints

Einführendes zu QCQPs

QCQP-Formulierung

Zusäliche Ungleichungsnebenbedingung sieht wie folgt aus:

$$x^T \Gamma x + \beta^T x \leq \alpha \tag{3.1}$$

Mit $x = z$ kommen somit zu den $lT + l_f$ ursprünglichen mit den linearen Ungleichungsnebenbedingungen assoziierten Funktionen zusätzliche p Funktionen für die logarithmic barrier function hinzu

$$-f_j(z) = \alpha_j - z^T \Gamma_j z - \beta_j^T z, \quad j = 1 \dots p \tag{3.2}$$

Für den Algorithmus wird nun weiterhin die Ableitung (Gradient und Hessian) der logarithmic barrier function $\phi(z)$ benötigt, die aus unter anderem $\nabla f_k(z)$ und $\nabla^2 f_k(z)$

gebildet werden.

$$\nabla f_j(z) = 2z^T\Gamma_j + \beta_j^T \tag{3.3}$$

$$\nabla^2 f_j(z) = 2\Gamma_j \tag{3.4}$$

Daraus ergibt sich der Gradient der barrier function $\phi(z)$ zu

$$\nabla\phi(z) = \sum_{k=1}^{lT+f_f+p} \frac{1}{\begin{bmatrix} h_i \\ \alpha_j \end{bmatrix}_k - \begin{bmatrix} p_i^T \\ \beta_j^T + z^T\Gamma_j \end{bmatrix}_k z} \begin{bmatrix} p_i^T \\ \beta_j^T + 2z^T\Gamma_j \end{bmatrix}_k \tag{3.5}$$

Mit

$$\hat{P}(z) = \begin{bmatrix} P \\ \beta_1^T + z^T\Gamma_1 \\ \vdots \\ \beta_p^T + z^T\Gamma_p \end{bmatrix}, \quad \hat{h} = \begin{bmatrix} h \\ \alpha_1^T \\ \vdots \\ \alpha_p^T \end{bmatrix} \tag{3.6}$$

lässt sich das auch einfacher schreiben:

$$\nabla\phi(z) = \hat{P}^T(2z)\hat{d} \tag{3.7}$$

Wobei

$$\hat{d}_k = \frac{1}{\hat{h}_k - \hat{p}_k(z)z} \tag{3.8}$$

und $\hat{p}_k(2z)$ die Zeilen in $\hat{P}(2z)$ sein sollen. Um Φ zu bilden wird noch die zweite Ableitung von der barrier function benötigt

$$\nabla^2\phi(z) = \hat{P}(2z)\mathrm{diag}(\hat{d})^2\hat{P}(2z) + \sum_{j=1}^{p}\left(\hat{d}_{(lT+l_f)+j}2\Gamma_j\right) \tag{3.9}$$

Die Berechnungen der Ableitungen haben also analog Struktur wie [WB10] bis auf den zusätzlichen Term, der bei der Hessian hinzukommt. Aber da dieser Term in der Implementierung nicht durch die Multiplikation einer großen Matrix *Gamma* mit z sonder identischer Teilmatrizen $Gamma_k$ mit den T x_k ist sicher gestellt das keine größeren Blöcke als $n \times n$ an den richtigen Stellen erzeugt werden. Das bedeutet, dass auch hier die Struktur der Matrix Φ nicht verändert wird.

3.1.2 Erweiterung für Second Order Cone Problems

Bei der Ungleichungsnebenbedingungen für die Second Order Cone Constraints lässt sich die Berechnung leider nicht so Überschaubar darstellen, da hier Funktion und Ableitung nicht mehr so schön ähnlich sind. Die Anpassungen müssen daher wie folgt aussehen.

SOCP Formulierung

Zusäliche Ungleichungsnebenbedingung sieht wie folgt aus [TODO: woher [BV04]]:

$$\|Ax + b\|_2 \leq c^T x + d \tag{3.10}$$

Als generalized inequality nimmt Gleichung 3.10 leicht andere Form an:

$$\|Ax + b\|_2^2 \leq \left(c^T x + d\right)^2 \tag{3.11}$$

Im folgenden lässt sich die Ungleichungsnebenbedingung so leichter umformen und ableiten, speziell hebt sich so später ein Wurzelterm auf. Mit $x = z$ ergeben sich die zusätzlichen j Funktionen für die logarithmic barrier function somit zu

$$-f_j\left(x\right) = \left(c_j^T x + d_j\right)^2 - \|A_j x + b_j\|_2^2 \tag{3.12}$$

Alle k barrier function Funktionen lassen sich zu

$$-f_k\left(z\right) = -\begin{bmatrix} f_i \\ f_j \end{bmatrix}_k = \begin{bmatrix} h_i \\ 0 \end{bmatrix}_k - \begin{bmatrix} p_i^T z \\ \left(\|A_j z + b_j\|_2^2 - \left(c_j^T z + d_j\right)^2\right) \end{bmatrix}_k \tag{3.13}$$

zusammenfassen. Dabei lässt sich f_j auch noch auflösen zu

$$f_j(z) = d_j^2 - b_j^T b_j - \left[-\left(c_j^T z + 2d_j\right) c_j^T + \left(z^T A_j^T + 2b_j^T\right) A_j\right] z \tag{3.14}$$

Für den Algorithmus wird nun weiterhin die Ableitung (Gradient und Hessian) der logarithmic barrier function $\phi(z)$ benötigt, die aus unter anderem $\nabla f_k(z)$ und $\nabla^2 f_k(z)$ gebildet werden.

$$\nabla f_k\left(z\right) = \begin{bmatrix} p_i^T \\ \left(-2\left(\left(c_j^T z + d_j\right) c_j - A_j^T\left(A_j z + b_j\right)\right)\right)^T \end{bmatrix}_k \tag{3.15}$$

Das Transponieren der unteren Zeile ergibt sich dadurch, dass man zu P eine Spalte anhängt also zu P^T die transformierte dieser Spalte. Damit lässt sich auch hier $\nabla f_k(z)$ wie schon bei die quadratic constraint schreiben.

$$\nabla\phi\left(z\right) = \hat{P}^T(2z)\hat{d} \tag{3.16}$$

Mit

$$\nabla^2 f_k\left(z\right) = \begin{bmatrix} 0 \\ -2\left(c_j c_j^T - A_j^T A_j\right) \end{bmatrix}_k \tag{3.17}$$

ergibt sich $\nabla^2\phi(z)$ zu

$$\nabla^2\phi(z) = \hat{P}(2z)\mathrm{diag}(\hat{d})^2\hat{P}(2z) + \sum_{j=1}^{p}\left(\hat{d}_{(lT+l_f)+j} - 2\left(c_j c_j^T - A_j^T A_j\right)\right) \tag{3.18}$$

[TODO richtiger Index für d]

3.1.3 Komplette Formulierung

Ungleichungsnebenbedingungen

$$\hat{P}(z)z \leq \hat{h} \tag{3.19}$$

mit

$$\hat{P}(z) = \begin{bmatrix} P \\ \beta_1^T + z^T\Gamma_1 \\ \vdots \\ \beta_p^T + z^T\Gamma_p \\ -\left(c_1^T z + 2d_1\right)c_1^T + \left(z^T A_1^T + 2b_1^T\right)A_1 \\ \vdots \\ -\left(c_q^T z + 2d_q\right)c_q^T + \left(z^T A_q^T + 2b_q^T\right)A_q \end{bmatrix}, \quad \hat{h} = \begin{bmatrix} h \\ \alpha_1^T \\ \vdots \\ \alpha_p^T \\ d_1^2 - b_1^T b_1 \\ \vdots \\ d_q^2 - b_q^T b_q \end{bmatrix} \tag{3.20}$$

Daraus ergibt sich die Logarithmic barrier function zu

$$\phi(z) = \sum_{i=1}^{lT+l_f+p+q} -\log\left(\hat{h}_i - \hat{p}_i^T(z)z\right) \tag{3.21}$$

Wobei $\hat{p}_i^T(z)$ die Zeilen aus $\hat{P}(z)$ sind.
Bei der Berechnung des Residuums wird der Gradient der logarithmic barrier function benötigt

$$\nabla\phi(z) = \hat{P}^T(2z)\hat{d} \tag{3.22}$$

mit

$$\hat{d}_i = \frac{1}{\hat{h}_i - \hat{p}_i^T(z)z} \tag{3.23}$$

Dabei muss $\hat{P}^T(z)$ mit $2z$ aufgerufen werden wie oben beschrieben.
Hessian der der Logarithmic barrier function für Berechnung des Φ

$$\begin{aligned}\nabla^2\phi(z) =& \hat{P}(2z)\mathrm{diag}(\hat{d})^2\hat{P}(2z) \\ &+ \sum_{i=lT+lf+1}^{lT+lf+p}\left(\hat{d}_i 2\Gamma_i\right) + \sum_{j=lT+lf+p+1}^{lT+lf+p+q}\left(\hat{d}_j - 2\left(c_j c_j^T - A_j^T A_j\right)\right)\end{aligned} \tag{3.24}$$

3.2 Anpassung für test cases

Der implementierte Algorithmus wie in [Paper:, Section:] beschrieben kann auch verwendet werden, um Optimierungsprobleme zu lösen, die ihren Ursprung nicht in der Anwendung von MPC haben. Dazu sind keine wirklichen Anpassungen des Algorithmus notwendig. Da der Algorithmus allerdings die Struktur der bei MPC auftretenden Matrizen ausnutzt, muss der jeweilige test case so "transformiert" werden, dass dieser eine ähnliche Struktur aufweist.

3.2.1 Allgemeine Beschreibung der test cases

Nach [MM97] haben die test cases folgende Form:

$$\begin{aligned}\min \quad & \hat{c}^T\hat{x} + \frac{1}{2}\hat{x}^T\hat{Q}\hat{x} \\ \text{s.t.} \quad & \hat{A}\hat{x} = \hat{b} \\ & \hat{l} \le \hat{x} \le \hat{u}\end{aligned} \tag{3.25}$$

Aber es existieren auch test cases mit weiteren Ungleichungsnebenbedingung der Form:

$$\hat{b}_{lower} \le \hat{A}\hat{x} \le \hat{b}_{upper} \tag{3.26}$$

Vereinheitlicht für 3.25 und 3.26 schreiben

$$\begin{aligned} \min \quad & \hat{c}^T\hat{x} + \frac{1}{2}\hat{x}^T\hat{Q}\hat{x} \\ \text{s.t.} \quad & \hat{b}_{lower} \leq \hat{A}\hat{x} \leq \hat{b}_{upper} \\ & \hat{l} \leq \hat{x} \leq \hat{u} \end{aligned} \tag{3.27}$$

Wobei sich für

$$\hat{b} = \hat{b}_{lower} = \hat{b}_{upper}$$

die Gleichungsnebenbedingungen

$$\hat{A}\hat{x} = \hat{b}$$

ergeben

3.2.2 Algorithmus mit Prädiktionshorizont gleich eins

Um die test cases lösen zu können, muss der Prädiktionshorizont $T = 1$ gewählt werden. Die Optimierungsvariable beschränkt sich damit auf

$$z = (u(t), x(t+T)) \in \mathbb{R}^{(m+n)}, \quad T = 1$$

Die strukturierten Matrizen im Algorithmus zum lösen des Optimierungsproblems

$$\begin{aligned} \min \quad & z^T H z + g^T z \\ \text{s.t.} \quad & Pz \leq h, \quad Cz = b \end{aligned}$$

reduzieren sich damit auf folgende Form:

$$H = \begin{bmatrix} R & 0 \\ 0 & Q_f \end{bmatrix}$$

$$P = \begin{bmatrix} F_u & 0 \\ 0 & F_f \end{bmatrix}$$

$$C = \begin{bmatrix} -B & I \end{bmatrix}$$

$$g = \begin{bmatrix} r + 2S^T x(t) \\ q \end{bmatrix}$$

$$h = \begin{bmatrix} f - F_x x(t) \\ f_f \end{bmatrix}$$

$$b = \begin{bmatrix} Ax(t) \end{bmatrix}$$

Um den Algorithmus nun mit den test cases nach [MM97] zu testen muss

$$H = \frac{1}{2}\hat{Q}, \quad g = \hat{c}, \text{nicht korrekt, einzelne Untermatrizen setzen} \tag{3.28}$$

gesetzt werden. Die Ungleichungsnebenbedingung

$$F_u u(t) + F_x x(t) + F_f x(t+1) \leq f = f_u + f_x \tag{3.29}$$

ergeben sich zu

$$F_u = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$
$$F_x = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$
$$F_f = \begin{bmatrix} 0 & 0 \end{bmatrix}$$
$$f = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$
$$f_f = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Als Gleichungsnebenbedingungen bleibt im implementierten Algorithmus

$$x(t+1) = Ax(t) + Bu(t) \tag{3.30}$$

Da allerdings x(t+1) auch zu dem Vektor der Optimierungsvariablen gehört muss

$$\hat{b} = -Ax(t) \quad \text{mit} \quad A = -I \tag{3.31}$$

gesetzt werden. Zusätzlich wird mit weiteren Ungleichungsnebenbedingung dafür gesorgt, dass $x(t+1)$ im Optimum nahe der Null liegt. Das bedeutet aber auch, dass in der Auswertung der Güte der eingehaltenen Gleichungsnebenbedingungen auch die Genauigkeit der zusätzlichen Ungleichungsnebenbedingung betrachtet werden muss.

Literaturverzeichnis

[BV04] Boyd, Stephen ; Vandenberghe, Lieven: *Convex Optimization*. University Press, 2004

[MM97] Maros, István ; Mészáros, Csaba: A Repository of Convex Quadratic Programming Problems. (1997)

[WB10] Wang, Yang ; Boyd, Stephen: Fast Model Predictive Control Using Online optimization. In: *IEEE Transactions on Control Systems Technology* 18 (2010), Nr. 2, S. 267–278

Selbstständigkeitserklärung

Hiermit versichere ich, Hannes Heinemann, dass ich die vorliegende Bachelorarbeit mit dem Thema „Umsetzung eines Kanalmodells für Petri-Netz modellierte Funksysteme“ selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

22. Juli 2015 Hannes Heinemann