Nichtlineare Optimierung
(Abschnitt des B.Sc. Moduls Optimierung)

Moritz Diehl Florian Messerer

2. Februar 2023

Präambel

Bei dem vorliegenden Dokument handelt es sich um das Skript zum Vorlesungsabschnitt "Nichtlineare Optimierung" des B.Sc. Moduls "Optimierung". Dieser Abschnitt umfasst die Vorlesungen 6 und 7 des Gesamtmoduls, gehalten von Prof. Moritz Diehl, sowie eine Übung (Übung 5), betreut von Florian Messerer. Die beiden Vorlesungen wurden bereits im Januar 2020 gehalten. Eine Aufzeichnung kann auf der Kursseite[1] gefunden werden. Dort sind außerdem die Übungen zur Verfügung gestellt. Die durch ein Sternchen gekennzeichnete Sektion 7.2 wurde in den Vorlesungen nicht behandelt, ist aber zur Vollständigkeit enthalten (allerdings nicht prüfungsrelevant).
Eine Vertiefung der hier angeschnittenen Materialien wird in der von Prof. Moritz Diehl gehaltenen M.Sc. Vorlesung *Numerical Optimization*[2] unterrichtet. Als weiterführende Litatur können wir insbesondere das Lehrbuch *Numerical Optimization* von Jorge Nocedal und Stephen Wright [1] empfehlen, sowie *Convex Optimization* von Stephen Boyd [2]. Als Anwendungsbeispiele tauchen in diesem Skript Optimalsteuerungsprobleme auf. Diese sind der Fokus der M.Sc. Vorlesung *Numerical Optimal Control*.[3]

Notation

Für eine vektorwertige Funktion $g : \mathbb{R}^n \to \mathbb{R}^m$, $w \mapsto g(w)$, benutzen wir die Konvention, dass der Gradient $\nabla g(w)$ als transponierte Jakobimatrix definiert ist, $\nabla g(w) := \frac{\partial g}{\partial w}(w)^\top$, also

$$\frac{\partial g}{\partial w}(w) = \begin{bmatrix} \frac{\partial g_1}{\partial w_1} & \dots & \frac{\partial g_1}{\partial w_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial w_1} & \dots & \frac{\partial g_m}{\partial w_n} \end{bmatrix}, \quad \nabla g(w) = \frac{\partial g}{\partial w}(w)^\top = \begin{bmatrix} \frac{\partial g_1}{\partial w_1} & \dots & \frac{\partial g_m}{\partial w_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_1}{\partial w_n} & \dots & \frac{\partial g_m}{\partial w_n} \end{bmatrix}. \tag{1}$$

Für zwei Vektoren $x \in \mathbb{R}^n$ und $y \in \mathbb{R}^m$ benutzen wir die Notation $(x, y) = [x^\top, y^\top]^\top$ um die vertikale Konkatenation von x und y auszudrücken. Also wenn $z = (x, y)$, haben wir, dass $z \in \mathbb{R}^{n+m}$.

[1] `https://www.syscop.de/teaching/ws2022/optimierung-bsc`
[2] Unter `https://www.syscop.de/teaching/ws2020/numerical-optimization` findet sich eine Onlinekursvariante zum Selbststudium.
[3] `https://www.syscop.de/teaching/ss2020/numerical-optimal-control-online`

6 Nichtlineare Optimierung mit Gleichungsbeschränkungen

Ein allgemeines nichtlineares Programm (NLP) mit Gleichungsbeschränkungen lässt sich schreiben als

$$\min_{x \in \mathbb{R}^n} \quad f(x) \tag{2a}$$

$$\text{s.t.} \quad c(x) = 0. \tag{2b}$$

Hierbei ist $f\colon \mathbb{R}^n \to \mathbb{R}$ die Zielfunktion und $c\colon \mathbb{R}^n \to \mathbb{R}^m$ die Nebenbedingungsfunktion, in der die m Nebenbedingungen gesammelt sind. Wir nehmen beide Funktionen als (mindestens) zweimal kontinuierlich differenzierbar an, was oft als $f, c \in C^2$ geschrieben wird.
Dieses NLP is konvex, wenn f konvex und c affin ist (also wenn sich c als $c(x) = Ax + b$ schreiben lässt).
Die Lagrangefunktion von (2) definieren wir als

$$\mathcal{L}(x, \lambda) = f(x) - \lambda^\top c(x), \tag{3}$$

mit Lagrangemultiplikator $\lambda \in \mathbb{R}^m$, sodass jeder Eintrag λ_i genau einer Nebenbedingung $c_i(x) = 0$, $i = 1, \dots, m$, zugeordnet ist.
Die notwendigen Optimalitätsbedingungen erster Ordnung von (2), auch KKT-Bedingungen genannt, sind gegeben durch

$$\nabla_x \mathcal{L}(x, \lambda) = 0, \qquad (n \text{ Zeilen}) \tag{4a}$$

$$c(x) = 0. \qquad (m \text{ Zeilen}) \tag{4b}$$

An jedem Minimierer x^* von (2) gibt es ein λ, sodass diese Bedingungen halten. Für konvexe Probleme sind diese Bedingungen auch hinreichend, d.h., wenn wir (x, λ) gefunden haben, an dem die Bedingungen halten und das Problem konvex ist, wissen wir, dass es sich um ein globales Minimum handelt.
Da es sich bei (4) um ein System von $(n + m)$ Gleichungen in $(n + m)$ Variablen handelt, ist es im Prinzip eindeutig lösbar.

6.1 Algorithmische (oder automatische) Differentiation (AD)

Die Funktionen f, c und $\mathcal{L}$ bestehen aus (vielen) Elementaroperationen, wie z.B. Addition, Multiplikation, sin, exp, etc., deren Ableitungen wir kennen. Dies wird von AD ausgenutzt, um erste und höhere Ableitungen von Funktionen effizient und präzise zu berechnen. So können wir z.B. den Gradienten von f zu nur einem dreifachen der Kosten der Auswertung von f selbst berechnen, $\text{cost}(\nabla f) \approx 3\,\text{cost}(f)$. Es existieren viele AD-Tools, z.B. CasADi [3].

6.2 Quadratische Programme (QP)

Das NLP (2) ist ein Quadratisches Programm (QP), wenn

- Die Zielfunktion f linear-quadratisch ist, $f(x) = \frac{1}{2} x^\top Q x + g^\top x$
- Die Nebenbedingungsfunktion c affin ist, $c(x) = Ax + b$.

Ein allgemeines gleichungsbeschränktes QP ist also gegeben durch

$$\min_{x \in \mathbb{R}^n} \quad \tfrac{1}{2} x^\top Q x + g^\top x \tag{5a}$$

$$\text{s.t.} \quad Ax + b = 0. \tag{5b}$$

Bemerkung 6.1. Die Matrizen Q und A sowie die Vektoren g und b können leicht mit AD-Tools durch Ableiten von f und g an der Stelle $x = 0$ generiert werden, da

$$f(x) = f(0) + \overbrace{\nabla f(0)^\top}^{g^\top} x + \tfrac{1}{2} x^\top \overbrace{\nabla^2 f(0)}^{Q} x, \tag{6a}$$

$$c(x) = \underbrace{c(0)}_{b} + \underbrace{\nabla c(x)^\top}_{A} x. \tag{6b}$$

Hier sieht man auch, dass Q symmetrisch ist, da es sich um die Hessematrix von f handelt.

Das QP ist konvex, wenn Q positiv semidefinit ist: $Q \succeq 0$ (alle Eigenwerte von Q sind nichtnegativ). Das QP kann über die zugegörigen KKT-Bedingungen gelöst werden:

$$\nabla_x \mathcal{L}(x, \lambda) = g + Qx - A^\top \lambda \overset{!}{=} 0, \tag{7a}$$

$$c(x) = b + Ax \overset{!}{=} 0. \tag{7b}$$

Dies ist ein lineares System in (x, λ),

$$\begin{bmatrix} g \\ b \end{bmatrix} + \underbrace{\begin{bmatrix} Q & -A^\top \\ A & 0 \end{bmatrix}}_{\text{“KKT-Matrix”}} \begin{bmatrix} x \\ \lambda \end{bmatrix} = 0, \tag{8}$$

und die Lösung ist gegeben durch

$$\begin{bmatrix} x \\ \lambda \end{bmatrix} = - \begin{bmatrix} Q & -A^\top \\ A & 0 \end{bmatrix}^{-1} \begin{bmatrix} g \\ b \end{bmatrix}. \tag{9}$$

Beispiel 6.2 (Optimalsteuerungsproblem mit quadratischer Zielfunktion und linearer Dynamik)**.** Wir betrachten einen einfachen Oszillator in Form eines Masse-Feder-Systems, also eine Masse, die an einer Feder befestigt ist. Die Masse liegt auf einer Oberfläche und hat Position $p \in \mathbb{R}$. Die andere Seite der Feder hat Position $a \in \mathbb{R}$, welche wir frei wählen und beliebig schnell ändern können. Fassen wir Position p und deren Geschwindigkeit v im Zustandsvektor $s = (p, v)$ zusammen, können wir die Dynamik des Systems durch die gewöhnliche Differentialgleichung (ordinary differential equation, ODE)

$$\dot{s} = \begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v \\ k_{\text{Feder}}(a - p) \end{bmatrix} =: g^{\text{lin}}(s, a) \tag{10}$$

beschreiben, mit Federkonstante $k_{\text{Feder}} = 1$. Zur Vereinfachung haben wir hier alle Einheiten ignoriert. Wir betrachten das System über die Zeitdauer $T = 4$ und diskretisieren diese in $N = 30$ Zeitschritte. Auf dem resultierenden diskreten Zeitgitter verändert sich das System als

$$s_{i+1} = G_h^{\text{lin}}(s_i, a_i). \tag{11}$$

Hierbei ist s_i der Zustand zum diskreten Zeitpunkt i, $i = 0, \ldots, N$, und $h = \frac{T}{N}$ die Länge der Zeitschritte. Die Steuerung a_i, $i = 0, \ldots, N-1$, wird für die Dauer eines Zeitschritts konstant gehalten. Die Integratorfunktion $G_h^{\text{lin}}(s_i, a_i)$ erhalten wir, in dem wir zehn Schritte des Runge-Kutta-Verfahrens 4. Ordnung (RK4)[4] auf das System anwenden. Dabei bleibt die Linearität

[4] `https://de.wikipedia.org/wiki/Klassisches_Runge-Kutta-Verfahren`

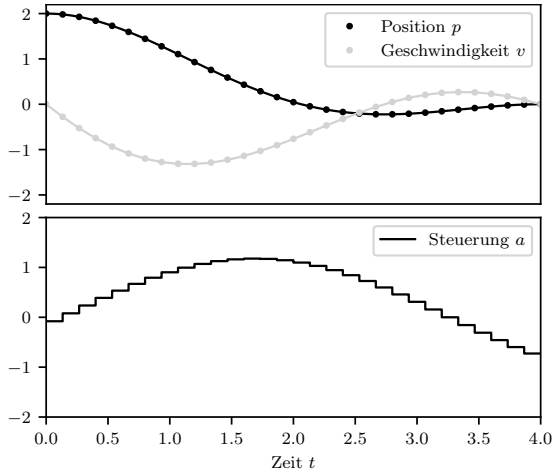

Abbildung 1: Lösung des Optimalsteuerungsproblems (12) aus Beispiel 6.2.

von (10) erhalten. Das heißt, die Funktion $G_h^{\text{lin}}(s_i, a_i)$ ist linear in (s_i, a_i). Unser Ziel ist es nun, das System, angefangen in der Startposition $\bar{s}_0 = (2, 0)$, im Ursprung zu stabilisieren, sodass $s_N = (0, 0)$. Dabei wollen wir insbesondere den Steuerungsaufwand klein halten, aber haben auch eine leichte Präferenz dafür, dass die Position möglichst früh bei 0 ist. Alles zusammen drücken wir als das folgende Optimalsteuerungsproblem aus:

$$\min_{\substack{s_0,\dots,s_N\in\mathbb{R}^2,\\ a_0,\dots,a_{N-1}\in\mathbb{R}}} \quad \sum_{i=0}^{N-1} a_i^2 + 0.01 p_i^2 \tag{12a}$$

$$\text{s.t.} \qquad s_0 = \bar{s}_0, \tag{12b}$$

$$s_{i+1} = G_h^{\text{lin}}(s_i, a_i), \quad i = 0, \dots, N-1, \tag{12c}$$

$$s_N = 0. \tag{12d}$$

Um dieses Problem in die NLP-Standardform (2) zu bringen, sammeln wir alle Entscheidungsvariablen in $x = (s_0, a_0, \dots, a_{N-1}, s_N)$. Die Zielfunktion bleibt gegeben als $f(x) = \sum_{i=0}^{N-1} a_i^2 + 0.01 p_i^2$. Da die allgemeine Nebenbedingung $c(x) = 0$ ist, müssen wir in den Nebenbedingungen von (1) alle Terme auf eine Seite bringen, was wir durch

$$c(x) = \begin{bmatrix} s_0 - \bar{s}_0 \\ s_1 - G_h^{\text{lin}}(s_0, a_0) \\ \vdots \\ s_N - G_h^{\text{lin}}(s_{N-1}, a_{N-1}) \\ s_N \end{bmatrix} \tag{13}$$

erreichen. Wir stellen fest, dass $f(x)$ quadratisch ist und $c(x)$ affin, sodass es sich bei (12) um ein QP handelt. Da $f(x)$ außerdem konvex ist (und die Nebenbedingungen affin, wie bei jedem QP), handelt es sich um ein konvexes QP. Die Lösung ist definiert durch (8) bzw. (9), und in Abb. 1 dargestellt.

6.3 Newton-Verfahren für NLP mit Gleichungsbeschränkungen

Unser Ziel ist das Lösen des nichtlinearen Gleichungssystems (4), also der notwendigen Optimalitätsbedingungen, um so eine mögliches Minimum von (2) zu erhalten. Zur Vollständigkeit wiederholen wir diese hier:

$$\nabla_x \mathcal{L}(x, \lambda) = 0, \tag{14a}$$
$$c(x) = 0. \tag{14b}$$

Wir sammeln alle Variablen in z und alle Gleichungen in F,

$$z := \begin{bmatrix} x \\ \lambda \end{bmatrix} \in \mathbb{R}^{n+m}, \quad F(z) := \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ c(x) \end{bmatrix} = \begin{bmatrix} \nabla f(x) - \nabla c(x)\lambda \\ c(x) \end{bmatrix} \in \mathbb{R}^{n+m}, \tag{15a}$$

und definieren J als die Jakobimatrix von F,

$$J(z) := \frac{\partial F}{\partial z}(z) = \begin{bmatrix} \nabla_x^2 \mathcal{L}(x, \lambda) & -\frac{\partial c}{\partial x}(x)^\top \\ \frac{\partial c}{\partial x}(x) & 0 \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}. \tag{15b}$$

Insgesamt können wir so (14) als

$$F(z) = 0 \tag{16}$$

zusammenfassen. Dieses nichtlineare Gleichungssystem lösen wir iterativ durch das *Newton-Verfahren*, also durch sequentielle Linearisierung von (16):

1. Starte mit Anfangsschätzwert $z_{[0]} \in \mathbb{R}^{n+m}$, setze $k = 0$.
2. Linearisiere $\boxed{F(z) = 0}$ an Stelle $z_{[k]}$ (Taylor-Reihe 1. Ordnung):

   $$\boxed{F(z_{[k]}) + J(z_{[k]})(z - z_{[k]}) = 0}$$

3. Löse das lineare System, um $z_{[k+1]}$ zu erhalten:

   $$\boxed{z_{[k+1]} = z_{[k]} - J(z_{[k]})^{-1} F(z_{[k]})}$$

4. Stop falls $\|F(z_{[k]})\| \leq \varepsilon$, mit Toleranz ε (z.B. $\varepsilon = 10^{-8}$).
5. $k \leftarrow k + 1$, gehe zu 2.

Beispiel 6.3 (Newton-Verfahren für ein gleichungsbeschränktes NLP)**.** Wir betrachten das NLP

$$\min_{x \in \mathbb{R}^2} \quad x_1 + x_2 \tag{17a}$$
$$\text{s.t.} \quad 2 - x_1^2 - x_2^2 = 0, \tag{17b}$$

also $n = 2$, $m = 1$, $f(x) = x_1 + x_2$, $c(x) = 2 - x_1^2 - x_2^2$ und $\lambda \in \mathbb{R}$. Siehe Abb. 2 für eine Visualisierung. Demnach ist in diesem Beispiel

$$z = \begin{bmatrix} x_1 \\ x_2 \\ \lambda \end{bmatrix}, \quad F(z) = \begin{bmatrix} 1 + 2x_1\lambda \\ 1 + 2x_2\lambda \\ 2 - x_1^2 - x_2^2 \end{bmatrix}, \quad J(z) = \begin{bmatrix} 2\lambda & 0 & 2x_1 \\ 0 & 2\lambda & 2x_2 \\ -2x_1 & -2x_2 & 0 \end{bmatrix}.$$

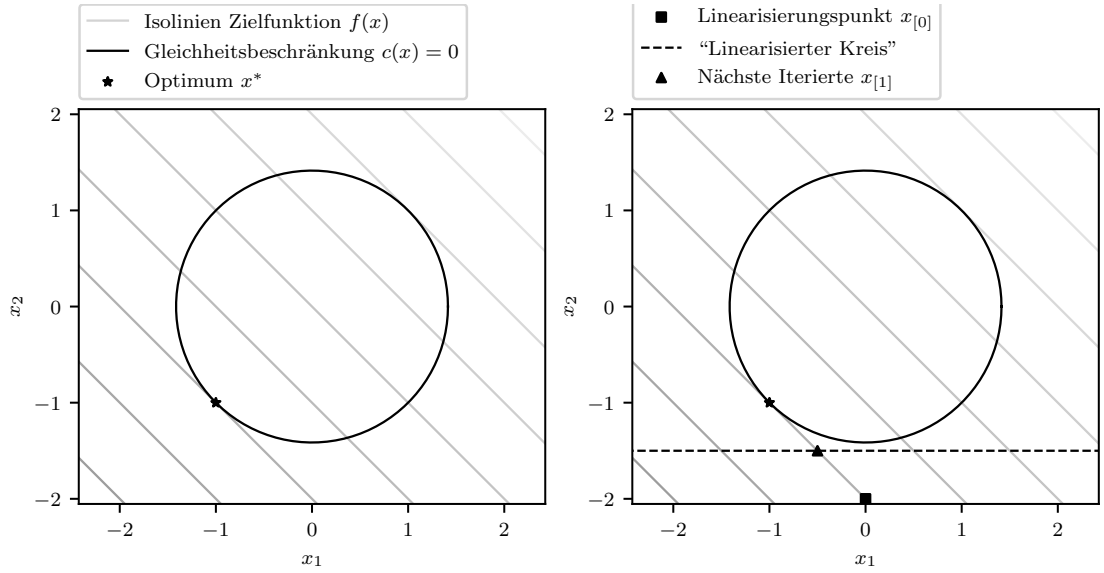

Abbildung 2: Visualisierung von Beispiel 6.3. Links: Skizze des betrachteten NLP. Rechts: Darstellung des Linearisierungsschrittes.

Wir starten unser Newton-Verfahren bei $z_{[0]} = (x_{[0]}, \lambda_{[0]}) = (0, -2, 1)$, und erhalten das an $z_{[0]}$ linearisierte Nullstellenproblem.

$$F(z_{[0]}) + J(z_{[0]})(z - z_{[0]}) = \begin{bmatrix} 1 \\ -3 \\ -2 \end{bmatrix} + \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & -4 \\ 0 & 4 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 + 2 \\ \lambda - 1 \end{bmatrix} = 0. \tag{18}$$

Dies ist ein lineares Gleichungssystem in $z = (x_1, x_2, \lambda)$, das wir nun lösen. Wir haben

$$\begin{aligned} &\text{erste Zeile:} && 1 + 2x_1 = 0 \Leftrightarrow x_1 = -\tfrac{1}{2} \\ &\text{letzte Zeile:} && -2 + 4(x_2 + 2) = 0 \Leftrightarrow x_2 = -\tfrac{3}{2} \\ &\text{mittlere Zeile:} && -3 + 2(x_2 + 2) - 4(\lambda - 1) = 0 \Leftrightarrow \lambda = \tfrac{5}{4} + \tfrac{1}{2}x_2 = \tfrac{1}{2}. \end{aligned}$$

Die Lösung des linearen Gleichungssystems – und damit unsere nächste Iterierte – ist also $z_{[1]} = (-\frac{1}{2}, -\frac{3}{2}, \frac{1}{2})$. Hier würden wir jetzt erneut linearisieren, um so $z_{[2]}$ zu erhalten, etc.

Beispiel 6.4 (Optimalsteuerungsproblem mit konvexer Zielfunktion und linearer Dynamik)**.** Wir betrachten wieder das Masse-Feder-System aus Beispiel 6.2. Wir wollen Steuerungen, die von 0 abweichen, noch stärker vermeiden. Dies erreichen wir, in dem wir die Zielfunktion zu

$$f(x) = \sum_{i=0}^{N-1} a_i^2 + a_i^4 + 0.01 p_i^2 \tag{19}$$

modifizieren, also zusätzliche Terme a_i^4 addieren. Der Rest des Problems bleibt gleich, also haben

wir jetzt

$$\min_{\substack{s_0,\dots,s_N\in\mathbb{R}^2,\\ a_0,\dots,a_{N-1}\in\mathbb{R}}} \quad \sum_{i=0}^{N-1} a_i^2 + a_i^4 + 0.01p_i^2 \tag{20a}$$

$$\text{s.t.} \qquad s_0 = \bar{s}_0, \tag{20b}$$

$$s_{i+1} = G_h^{\text{lin}}(s_i, a_i), \quad i = 0, \dots, N-1, \tag{20c}$$

$$s_N = 0. \tag{20d}$$

Da die Zielfunktion jetzt nicht mehr quadratisch ist, handelt es sich nicht mehr um ein QP. Allerdings ist $f(x)$ immer noch konvex, und die Gleichungsbeschränkungen affin. Also ist es ein konvexes NLP, welches wir mit dem Newton-Verfahren lösen. Die Definition von x und $c(x)$ bleibt gleich wie in Beispiel 6.4. Wir initialisieren bei $z_{[0]} = 0$ und verwenden als Konvergenzkriterium $\|F(z_{[k]})\|_\infty \leq 10^{-12}$, was in etwa der numerischen Präzision entspricht, die wir mit Standard-Floating-Point-Variablen erreichen können. Die resultierenden Iterationen sind in Abb. 3 visualisiert.

Beispiel 6.5 (Optimalsteuerungsproblem mit quadratischer Zielfunktion und nichtlinearer Dynamik)**.** Wir betrachten wieder das Optimalsteuerungsproblem aus Beispiel 6.2. Allerdings tauschen wir das Masse-Feder-System durch ein Pendel aus (eine Masse, die an einem festen Stab aufgehängt ist). Der Zustand ist nach wie vor $x = (p, v)$. Allerdings entspricht jetzt die Position p der horizontalen Position der Masse, und die Steuerung a der horizontalen Position der Pendelaufhängung, sodass sich bei $p = a$ die Masse exakt unter der Aufhängung befindet. Die Dynamik ist dann gegeben durch

$$\dot{s} = \begin{bmatrix} \dot{p} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ a_{\text{Schwerkraft}} \sin(\frac{a-p}{L}) \end{bmatrix} =: g^{\text{nonlin}}(s, a), \tag{21}$$

mit Pendellänge $L = 1$ und Schwerkraftbeschleunigung $a_{\text{Schwerkraft}}$. Wir merken an, dass es sich hierbei nur um ein angenähertes Modell handelt, bei dem der Fehler nur für geringe Auslenkungen $|a - p|$ vernachlässigbar wird. Im Gegensatz zum Masse-Feder-System ist die Dynamik jetzt nichtlinear. Dementsprechend ist auch die resultierende Integratorfunktion $s_{k+1} = G_h^{\text{nonlin}}(s_k, a_k)$ nichtlinear. Der Rest des Optimalsteuerungsproblems bleibt wie in (12), was in

$$\min_{\substack{s_0,\dots,s_N\in\mathbb{R}^2,\\ a_0,\dots,a_{N-1}\in\mathbb{R}}} \quad \sum_{i=0}^{N-1} a_i^2 + 0.01p_i^2 \tag{22a}$$

$$\text{s.t.} \qquad s_0 = \bar{s}_0, \tag{22b}$$

$$s_{i+1} = G_h^{\text{nonlin}}(s_i, a_i), \quad i = 0, \dots, N-1, \tag{22c}$$

$$s_N = 0, \tag{22d}$$

resultiert. Die Zielfunktion ist nach wie vor konvex. Allerdings sind jetzt die Gleichheitsbeschränkungen nichtlinear, weshalb die durch diese definierte zulässige Menge nicht konvex ist. Insgesamt ist also das gesamte NLP (22) nicht konvex. Wir können immer noch das Newton-Verfahren anwenden, um einen KKT-Punkt zu finden, haben aber erstmal keine Garantie, dass es sich bei diesem um ein lokales oder gar globales Optimum handelt (da es sich bloß um *notwendige* Optimalitätsbedingungen handelt, nicht um *hinreichende*). In der Praxis wird aber oft einfach davon ausgegangen, dass es sich zumindest um ein lokales Optimum handelt. Wir initialisieren das Newton-Verfahren wieder bei $z_{[0]} = 0$ und verwenden als Konvergenzkriterium $\|F(z_{[k]})\|_\infty \leq 10^{-12}$. Die resultierenden Iterationen sind in Abb. 4 visualisiert.

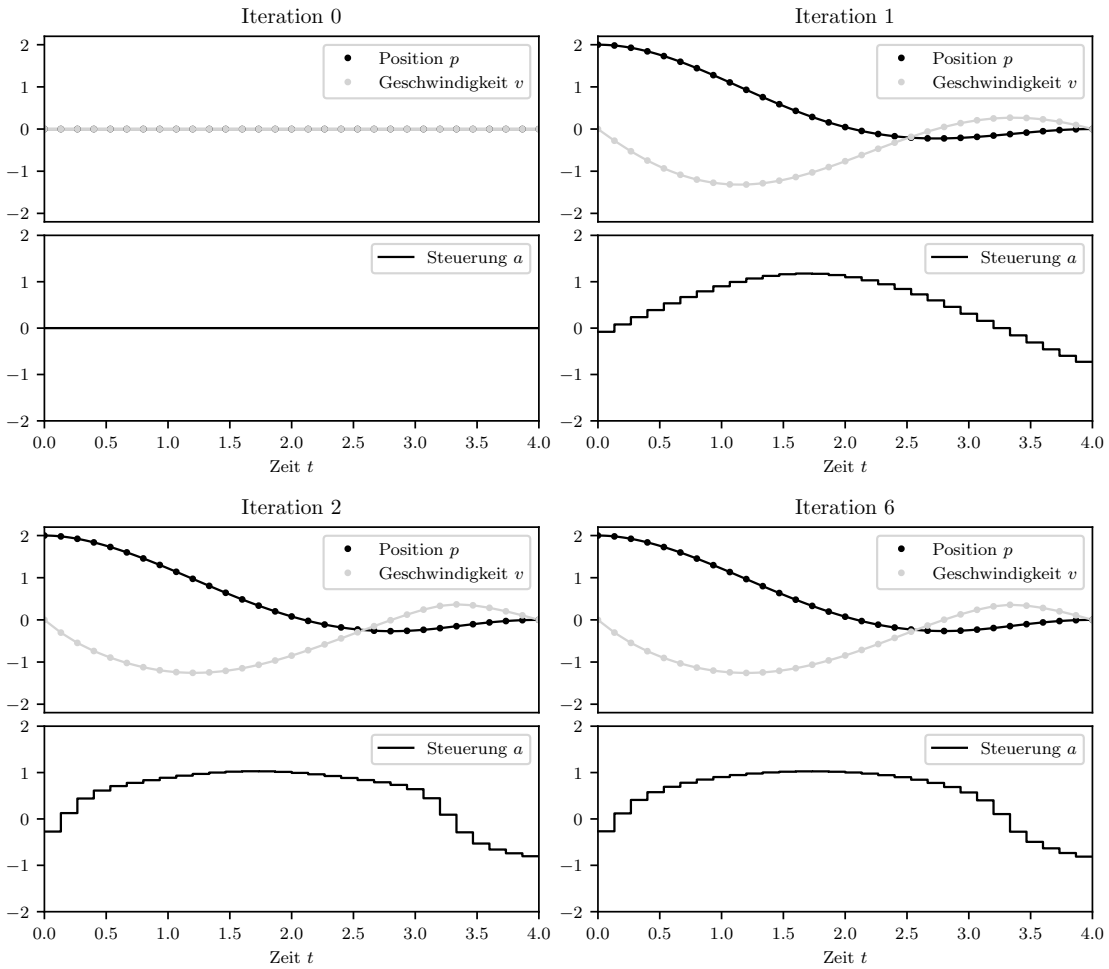

Abbildung 3: Die Iterationen des Newton-Verfahrens angewendet auf das konvexe Optimalsteuerungsproblem (20) aus Beispiel 6.4. Iteration 0 entspricht der Initialisierung des Verfahrens (Anfangsschätzwert). Nach nur einer Iteration kann man bereits die Form der späteren Lösung erkennen. Nach zwei Iteration ist die aktuelle Trajektorie nur noch schwer unterscheidbar von der Lösung. Nach sechs Iterationen ist das Verfahren konvergiert.

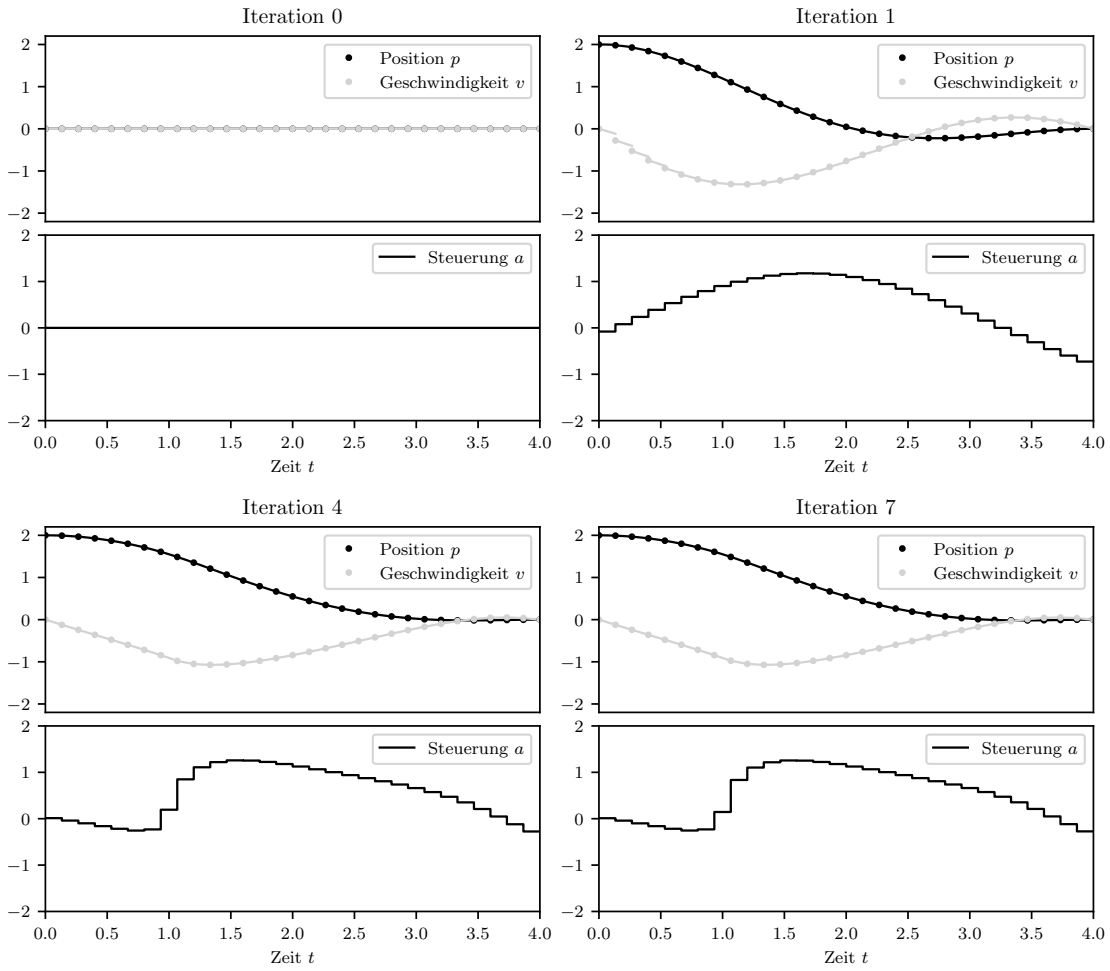

Abbildung 4: Die Iterationen des Newton-Verfahrens angewendet auf das nicht-konvexe Optimalsteuerungsproblem (22) aus Beispiel 6.5. Iteration 0 entspricht der Initialisierung des Verfahrens (Anfangsschätzwert). Nach nur einer Iteration kann man bereits grob die Form der späteren Lösung erkennen. Allerdings weist die Zustandstrajektorie noch Lücken auf, insbesondere im Bereich $t \in [0, 0.5]$. Dies kommt daher, dass die nichtlinearen Gleichheitsbeschränkungen während der Iterationen nicht notwendigerweise erfüllt sein müssen. Dies ist erst bei einer Lösung der Fall. Nach vier Iteration ist die aktuelle Trajektorie nur noch schwer unterscheidbar von der Lösung. Nach sieben Iterationen ist das Verfahren konvergiert.

Beispiel 6.6 (Optimalsteuerungsproblem mit nicht-quadratischer Zielfunktion und nichtlinearer Dynamik)**.** Wir betrachten wieder das Pendel aus Beispiel 6.5, addieren aber wie in Beispiel 6.4 zusätzliche Terme u_i^4 in die Zielfunktion, resultierend in

$$\min_{\substack{s_0,\dots,s_N\in\mathbb{R}^2,\\ a_0,\dots,a_{N-1}\in\mathbb{R}}} \quad \sum_{i=0}^{N-1} a_i^2 + a_i^4 + 0.01 p_i^2 \tag{23a}$$

$$\text{s.t.} \qquad s_0 = \bar{s}_0, \tag{23b}$$

$$s_{i+1} = G_h^{\text{nonlin}}(s_i, a_i), \quad i = 0, \dots, N-1, \tag{23c}$$

$$s_N = 0, \tag{23d}$$

Wie bereits (22) ist dieses Problem nicht-konvex, durch die zusätzlichen Terme aber nochmal deutlich nichtlinearer. Die resultierenden Iterationen des Newton-Verfahrens sind in Abb. 5 dargestellt.

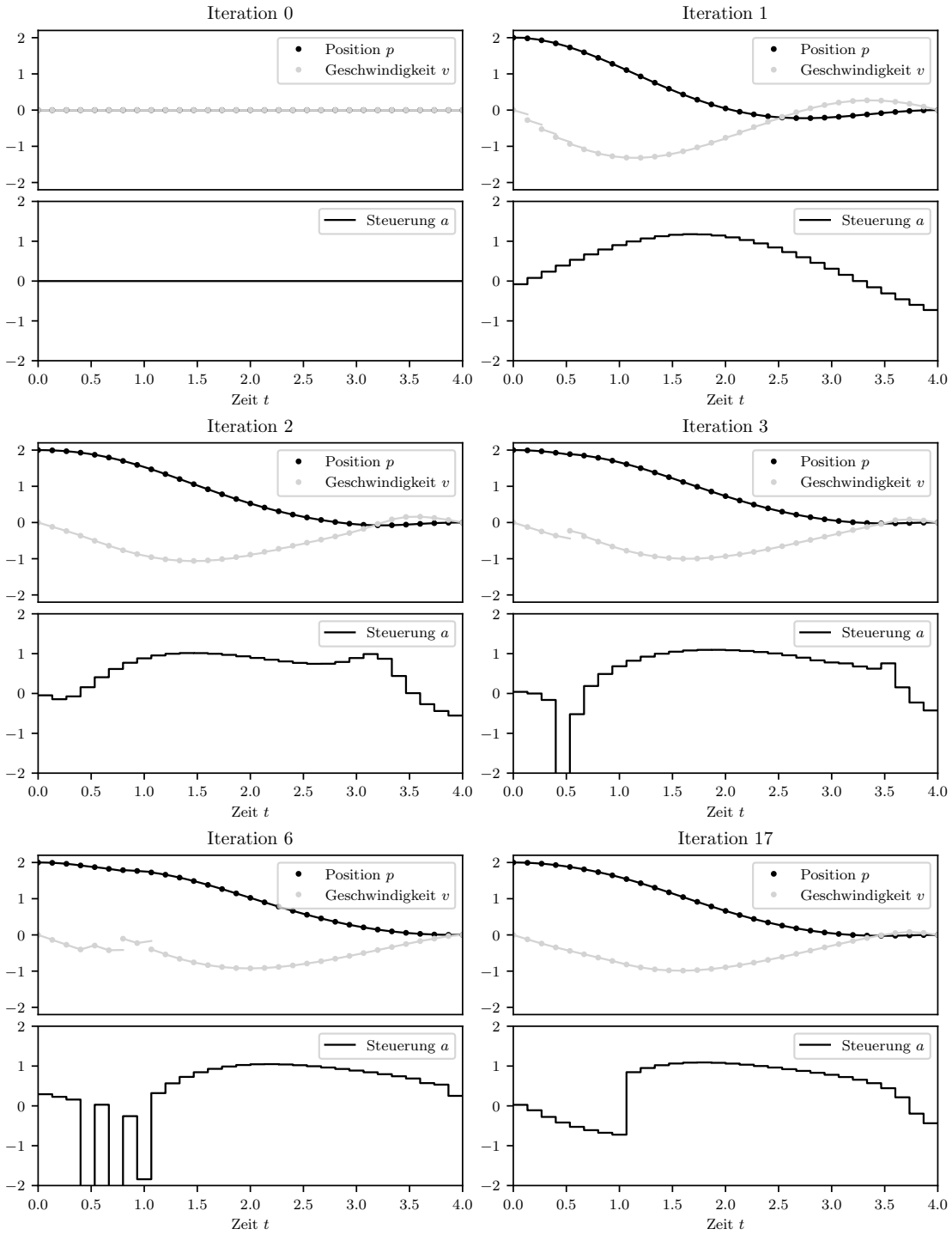

Abbildung 5: Die Iterationen des Newton-Verfahrens angewendet auf das nicht-konvexe Optimalsteuerungsproblem (23) aus Beispiel 6.6. Diesmal ist nach einer Iteration bezogen auf die Steuerungen die Lösung nur sehr grob erkennbar. Wieder beobachten wir, dass die Zustandstrajektorie während der Iterationen noch Lücken aufweisen kann, da die nichtlinearen Gleichheitsbeschränkungen noch nicht erfüllt sind. Die Steuerungstrajektorie durchläuft noch einige starke Störungen, bevor das Verfahren nach nach siebzehn Iterationen dann schließlich doch konvergiert.

7 Nichtlineare Optimierung mit Gleichungs- und Ungleichungsbeschränkungen

Ein allgemeines NLP ist gegeben durch

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) && \text{(24a)} \\ \text{s.t.} \quad & c_{\mathrm{E}}(x) = 0, && \text{(24b)} \\ & c_{\mathrm{I}}(x) \geq 0, && \text{(24c)} \end{aligned}$$

mit Gleichheitsbeschränkungen $c_{\mathrm{E}} \colon \mathbb{R}^n \to \mathbb{R}^{m_{\mathrm{E}}}$ (E = "equalities") und Ungleichheitsbeschränkungen $c_{\mathrm{I}} \colon \mathbb{R}^n \to \mathbb{R}^{m_{\mathrm{I}}}$ (I = "inequalities").
Dieses NLP ist konvex falls f konvex ist, c_{E} affin und c_{I} konkav (jede Komponente von c_{I} konkav ist).

7.1 Nichtlineare Innere Punkte (NIP) Methode

Wir transformieren (24) in eine andere Form durch die Einführung von Schlupfvariablen $\xi \in \mathbb{R}^m_{\mathrm{I}}$, $x^+, x^- \in \mathbb{R}^n$. Dadurch können wir (24) äquivalent als

$$\begin{aligned} \min_{x^+, x^-, \xi} \quad & f(x^+ - x^-) && \text{(25a)} \\ \text{s.t.} \quad & c_{\mathrm{E}}(x^+ - x^-) = 0, && \text{(25b)} \\ & c_{\mathrm{I}}(x^+ - x^-) - \xi = 0, && \text{(25c)} \\ & x^+ \geq 0, && \text{(25d)} \\ & x^- \geq 0, && \text{(25e)} \\ & \xi \geq 0, && \text{(25f)} \end{aligned}$$

schreiben. Die Äquivalenz lässt sich durch

$$x = x^+ - x^- \quad \text{sowie} \quad \left.\begin{aligned} c_{\mathrm{I}}(x) &= \xi \\ \xi &\geq 0 \end{aligned}\right\} \Leftrightarrow c_{\mathrm{I}}(x) \geq 0, \tag{26}$$

sehen. Definieren wir nun außerdem

$$\tilde{x} := \begin{bmatrix} x^+ \\ x^- \\ \xi \end{bmatrix}, \quad \tilde{f}(\tilde{x}) = f(x^+ - x^-), \quad c(\tilde{x}) := \begin{bmatrix} c_{\mathrm{E}}(x^+ - x^-) \\ c_{\mathrm{I}}(x^+ - x^-) - \xi \end{bmatrix} \tag{27}$$

erhalten wir die Form

$$\begin{aligned} \min_{\tilde{x}} \quad & \tilde{f}(\tilde{x}) && \text{(28a)} \\ \text{s.t.} \quad & c(\tilde{x}) = 0, && \text{(28b)} \\ & \tilde{x} \geq 0, && \text{(28c)} \end{aligned}$$

die wir als Standardform für die Innere-Punkte-Methode wählen. Wie wir gesehen haben, können wir ein Problem der Form (24) äquivalent immer auch in der Form (28) schreiben. Der Vorteil letzterer Form ist, dass die Ungleichheitsbeschränkungen (28c) "weniger kompliziert" sind als (24c), was die Implementierung von Algorithmen vereinfachen kann.

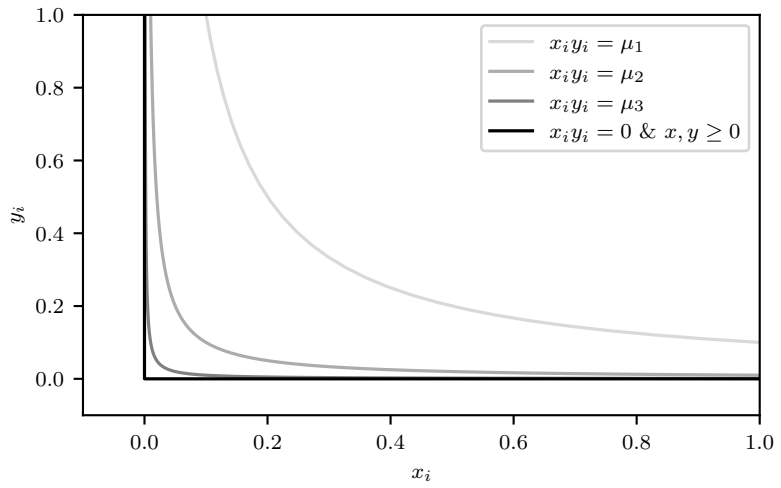

Abbildung 6: Nicht-differenzierbare Komplementaritätsbedingung $x_i y_i = 0$ & $x_i, y_i \geq 0$ sowie deren geglättete Approximation für verschiedene Werte von μ. Hierbei ist $\mu_1 > \mu_2 > \mu_3 > 0$. Konkret wurden für diesen Plot die Wert $\mu_i = 10^{-i}$ verwendet.

Um die Notation wieder etwas übersichtlicher zu machen, betrachten wir ab jetzt – wie demonstriert ohne Verlust von Allgemeinheit – Probleme der Form

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) && \text{(29a)} \\ \text{s.t.} \quad & c(x) = 0, && \text{(29b)} \\ & x \geq 0. && \text{(29c)} \end{aligned}$$

Die zugehörige Lagrangefunktion ist

$$\mathcal{L}(x, \lambda, \mu) = f(x) - \lambda^\top c(x) - y^\top x, \tag{30}$$

mit Lagrangemultiplikatoren $\lambda \in \mathbb{R}^m$ und $y \in \mathbb{R}^n$. Die KKT-Bedingungen sind

$$\begin{aligned} \nabla_x \mathcal{L}(x, \lambda, y) &= 0, && \text{(31a)} \\ c(x) &= 0, && \text{(31b)} \\ x_i \geq 0,\ y_i \geq 0,\ x_i y_i &= 0, \text{ für } i = 1, \ldots, n, \quad \text{(“Komplementaritätsbedingungen”).} && \text{(31c)} \end{aligned}$$

Problem: Die Komplementaritätsbedingungen beschreiben in (x_i, y_i) eine nicht-differenzierbare “L-förmige” Menge, siehe Abb. 6.
Idee 1: Glätten der Ecke durch den Parameter $\mu > 0$ und Ersetzen der Bedingung $x_i y_i = 0$ durch $x_i y_i = \mu$. Dies ergibt die “geglätteten KKT-Bedingungen”,

$$\begin{aligned} \nabla_x \mathcal{L}(x, \lambda, y) &= 0, && \text{(32a)} \\ c(x) &= 0, && \text{(32b)} \\ x_i y_i - \mu &= 0, \quad i = 1, \ldots, n. && \text{(32c)} \end{aligned}$$

Durch Definition von

$$z := \begin{bmatrix} x \\ \lambda \\ y \end{bmatrix} \in \mathbb{R}^{n+m+n}, \quad F(z,\mu) := \begin{bmatrix} \nabla_x \mathcal{L}(x,\lambda,y) \\ c(x) \\ x_1 y_1 - \mu \\ \vdots \\ x_n y_n - \mu \end{bmatrix} \in \mathbb{R}^{n+m+n}, \tag{33}$$

können wir das Ganze wieder als Nullstellenproblem in z auffassen, welches durch μ parametrisiert ist:

$$F(z,\mu) = 0. \tag{34}$$

Dieses können wir wie in Abschnitt 6.3 durch das Newton-Verfahren lösen.
Idee 2: Starte mit großem μ, verringere μ schrittweise.
Das Resultat ist der grundlegende NIP-Algorithmus:

1. Wähle $z_{[0]}$, $\mu_1 \gg 0$, setze $k \leftarrow 0$.
2. Teste, ob $\|F(z_{[k]}, 0)\| \leq \varepsilon$. Stopp, wenn ja.
3. Löse $\boxed{F(z, \mu_{k+1}) = 0}$ mit Newton-Verfahren, gestartet in $z_{[k]}$.
4. Setze $k \leftarrow k + 1$.
5. Verringere $\mu_{k+1} \leftarrow \alpha \mu_k$, mit $\alpha \in (0, 1)$.
6. Gehe zu 2.

Theorem 7.1. *Das Nullstellenproblem* (32) *bzw.* (34) *ist äquivalent zum "Barriere-Optimierungsproblem", gegeben durch*

$$\min_{x \in \mathbb{R}^n} \quad f(x) - \mu \sum_{i=1}^{n} \log x_i \tag{35a}$$

$$\text{s.t.} \quad c(x) = 0. \tag{35b}$$

Beweis. Das Nullstellenproblem (32) ist gegeben durch

$$\nabla f(x) - \nabla c(x)\lambda - y = 0, \tag{36a}$$

$$c(x) = 0, \tag{36b}$$

$$x_i y_i - \mu = 0, \quad i = 1, \ldots, n. \tag{36c}$$

Die letzte Zeile (36c) ist äquivalent zu $y_i = \mu / x_i$, $i = 1, \ldots, n$, sodass wir $y = (y_1, \ldots, y_n)$ in (36a) eliminieren können:

$$\nabla f(x) - \mu \begin{bmatrix} \frac{1}{x_1} \\ \vdots \\ \frac{1}{x_n} \end{bmatrix} - \nabla c(x)\lambda = 0, \tag{37a}$$

$$c(x) = 0. \tag{37b}$$

Hierbei handelt es sich genau um die KKT-Bedingungen des Barriereproblems (35). □

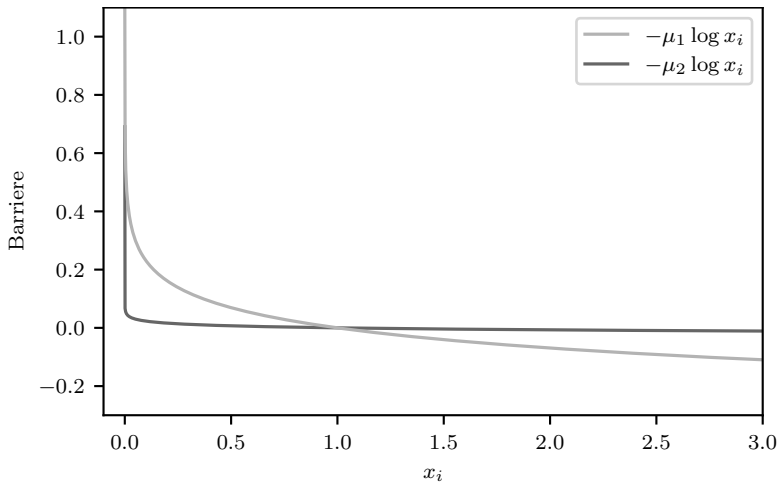

Abbildung 7: Interpretation der geglätteten KKT-Bedingungen als logarithmische Barriere, mit $\mu_1 > \mu_2$. Konkret wurden für diesen Plot die Wert $\mu_i = 10^{-i}$ verwendet.

Interpretation: Die Ungleichungen $x_i \geq 0$ werden durch "Barrieren" $-\mu \log x_i$ erzwungen. Je mehr sich x_i dem Wert 0, also dem Rand der zuverlässigen Menge, nähert, desto mehr wird der Wert der Zielfunktion erhöht. Im Limit $x_i \to 0$ geht dieser Wert gegen $+\infty$, siehe Abb. 7. Deshalb sind alle Iterierten $z_{[k]}$ "innere Punkte" (IP) der zulässigen Menge (bezogen auf die Ungleichungen).

Der NLP-Löser IPOPT [4] ist eine sehr robuste Implementierung einer IP-Methode. IP-Methoden können auch sehr zuverlässig zur Lösung von Linearen Programmen (LP) und Quadratischen Programmen (QP) angewendet werden.

7.2 *Sequentielle Quadratische Programmierung (SQP)

Wir betrachten wieder ein allgemeines NLP der Form (24), also

$$\min_{x \in \mathbb{R}^n} \quad f(x) \tag{38a}$$

$$\text{s.t.} \quad c_{\mathrm{E}}(x) = 0, \tag{38b}$$

$$c_{\mathrm{I}}(x) \geq 0. \tag{38c}$$

Die zugehörige Lagrangefunktion ist

$$\mathcal{L}(x, \lambda_{\mathrm{E}}, \lambda_{\mathrm{I}}) = f(x) - \lambda_{\mathrm{E}}^\top c_{\mathrm{E}}(x) - \lambda_{\mathrm{I}}^\top c_{\mathrm{I}}(x). \tag{39}$$

Falls wir einen QP-Löser zur Verfügung haben (z.B. eine IP-Methode), dann können wir iterativ QP lösen, die durch Ableitungen der NLP-Funktionen an der Stelle $x_{[k]}$ gewonnen werden:

$$\mathrm{QP}(x_{[k]}, Q_k): \min_{x \in \mathbb{R}^n} \quad \tfrac{1}{2}(x - x_{[k]})^\top Q_k (x - x_{[k]}) + \nabla f(x_{[k]})^\top (x - x_{[k]}) \tag{40a}$$

$$\text{s.t.} \quad c_{\mathrm{E}}(x_{[k]}) + \nabla c_{\mathrm{E}}(x_{[k]})^\top (x - x_{[k]}) = 0, \tag{40b}$$

$$c_{\mathrm{I}}(x_{[k]}) + \nabla c_{\mathrm{I}}(x_{[k]})^\top (x - x_{[k]}) \geq 0. \tag{40c}$$

Die Wahl der "Hesseapproximation" Q_k sollte (a) positive semidefinit sein, $Q_k \succeq 0$, (b) ähnlich zur exakten Hessematrix der Lagrangian sein, $Q_k \approx \nabla_x^2 \mathcal{L}(x, \lambda_{\mathrm{E}}, \lambda_{\mathrm{I}})$. Wird die Hessematrix exakt

gewählt, $Q_k = \nabla_x^2 \mathcal{L}(x, \lambda_\mathrm{E}, \lambda_\mathrm{I})$, konvergiert das Verfahren mit quadratischer Rate zur Lösung; andernfalls langsamer (z.B. superlinear oder linear, wenn überhaupt).

7.2.1 *Beschränktes Gauss-Newton (GN) Verfahren

Wir betrachten nun einen Spezialfall, in dem die Zielfunktion von (38) in der Form

$$f(x) = \tfrac{1}{2}\|r(x)\|_2^2, \tag{41}$$

mit "innerer Nichtlinearität" $r : \mathbb{R}^n \to \mathbb{R}^q$, gegeben ist (vgl. "nonlinear least-squares"). In diesem Fall ist eine gute Wahl der Hesseapproximation $Q_k = J(x_{[k]})^\top J(x_{[k]})$, wobei $J(x) = \frac{\partial r}{\partial x}(x)$ die Jakobimatrix von r ist. Das QP (40) ist dann äquivalent zu

$$\begin{aligned}
\mathrm{QP}(x_{[k]}, Q_k): \min_{x \in \mathbb{R}^n} \quad & \tfrac{1}{2}\|r(x_{[k]}) + J(x_{[k]})(x - x_{[k]})\|_2^2 && \text{(42a)}\\
\text{s.t.} \quad & c_\mathrm{E}(x_{[k]}) + \nabla c_\mathrm{E}(x_{[k]})^\top (x - x_{[k]}) = 0, && \text{(42b)}\\
& c_\mathrm{I}(x_{[k]}) + \nabla c_\mathrm{I}(x_{[k]})^\top (x - x_{[k]}) \geq 0. && \text{(42c)}
\end{aligned}$$

Beweis. Durch Expandieren der Zielfunktion erhalten wir

$$\begin{aligned}
&\tfrac{1}{2}\|r(x_{[k]}) + J(x_{[k]})(x - x_{[k]})\|_2^2\\
&= \tfrac{1}{2}\left(r(x_{[k]}) + J(x_{[k]})(x - x_{[k]})\right)^\top \left(r(x_{[k]}) + J(x_{[k]})(x - x_{[k]})\right)\\
&= \tfrac{1}{2}(x - x_{[k]})^\top \underbrace{J(x_{[k]})^\top J(x_{[k]})}_{Q_k}(x - x_{[k]}) + \underbrace{r(x_{[k]})^\top J(x_{[k]})}_{=\nabla f(x_{[k]})^\top}(x - x_{[k]}) + \tfrac{1}{2} r(x_{[k]})^\top r(x_{[k]}).
\end{aligned}$$

Bei letztem Term in obiger Summe handelt es sich um einen konstanten Term (bezogen auf die Optimierungsvariable x), sodass er die Lösung nicht beeinflusst und weggelassen werden kann. □

Ein Vorteil von GN-SQP ist, dass konvexe Strukturen des NLP im QP erhalten bleiben, was zuverlässige und robuste Konvergenz befördert. Hierbei handelt es sich um einen Spezialfall von Sequential Convex Programming (SCP), das auf dem Prinzip der sequentiellen Linearisierung der nicht-konvexen Problemkomponenten beruht. Ein Überblick ist in [5] gegeben.

Literatur

[1] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition, 2006.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. University Press, Cambridge, 2004.

[3] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[4] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[5] Florian Messerer, Katrin Baumgärtner, and Moritz Diehl. Survey of sequential convex programming and generalized Gauss-Newton methods. *ESAIM: Proceedings and Surveys*, 71:64–88, 2021.