2

Mathematische und numerische Werkzeuge

Im Folgenden wird eine Einführung in die verwendete Notation gegeben und Methoden zur numerischen Lösung von (*i*) linearen Gleichungssystemen und (*ii*) unrestringierten Optimierungsaufgaben vorgestellt. Die besprochenen Verfahren werden als ein Werkzeug und nicht als Forschungsgegenstand dieser Arbeit verstanden. Es gibt eine Vielzahl an Übersichtsarbeiten, die sich diesen Themen widmen. Für Details zur Lösung linearer Gleichungssysteme sei der geneigte Leser bspw. auf [157, 196 & 213] verwiesen. Ein tieferer Einblick in die Verfahren der numerischen Optimierung kann in [5, 50, 73 & 171] gewonnen werden.

2.1 Notation und Vorbemerkungen

Vor dem Einstieg in die technischen Aspekte werden an dieser Stelle häufig genutzte mathematische Konstrukte und Begrifflichkeiten nähergebracht. Die verwendeten Formalitäten dienen dazu, den technischen Teil dieser Arbeit mit größtmöglicher Sorgfalt und Präzision zu entwickeln. Der Anspruch an eine mathematische Rigorosität steht dabei nicht im Vordergrund. Mathematik wird als ein Werkzeug verstanden, das es erlaubt, komplizierte Sachverhalte prägnant darzustellen, nicht aber als ein Forschungsgegenstand dieser im Kern ingenieurwissenschaftlichen Arbeit.

Im Laufe der Arbeit werden Definitionen, Beweise, Anmerkungen und Beispiele aufgeführt. Diese werden durchnummeriert. Definitionen werden mit dem Symbol ♣, Beweise mit dem Symbol □, Anmerkungen mit dem Symbol ♢ und Beispiele mit dem Symbol ♠

A. Mang, *Methoden zur numerischen Simulation der Progression von Gliomen,*
Aktuelle Forschung Medizintechnik, DOI 10.1007/978-3-658-05246-1_2,
© Springer Fachmedien Wiesbaden 2014

geschlossen. Sätze und Hilfssätze werden durch Kursivschrift hervorgehoben. Algorithmen sind farblich unterlegt.

Die verwendete Notation ist wie folgt: Die Bezeichnungen für die Zahlenmengen sind konventionell. So wird bspw. mit $\mathbf{R}$ die Menge der reellen Zahlen bezeichnet. Es ist hervorzuheben, dass alle Funktionen – sofern nicht explizit vermerkt – über dem Körper der reellen Zahlen beschrieben werden[7]. Als Konvention für Koordinatenrichtungen wird der zugehörige Index rechts oben an die Variable geschrieben – bspw. werden Vektoren aus dem $\mathbf{R}^d$, wobei $d \in \mathbf{N}$ die Dimension des Raumes markiert, als Spaltenvektoren der Gestalt $x = (x^1, ..., x^d)^\mathsf{T} \in \mathbf{R}^d$ notiert. Für die Menge der strikt positiven, reellen Zahlen wird die gebräuchliche Notation $\mathbf{R}^+$ verwendet. Die Menge der positiven, reellen Zahlen wird mit $\mathbf{R}_0^+ := \mathbf{R}^+ \cup \{0\}$ bezeichnet. Weitere Einschränkungen werden durch einen Zusatz rechts unten am Symbol notiert; so ist bspw. $\mathbf{R}_{\geq s} := \{x \in \mathbf{R} : x \geq s\}$. Weitere Bezeichnungen für spezielle Zahlenmengen sind im Notationsverzeichnis auf S. 175 f. hinterlegt.

Eine Teilmenge des $\mathbf{R}^n$ wird als *Gebiet* bezeichnet, wenn sie offen und zusammenhängend ist. Mit $x := (x^1, ..., x^d)^\mathsf{T} \in \mathbf{R}^d$ wird üblicherweise die Ortsvariable bezeichnet. Der Parameter $t > 0$ markiert die Zeitvariable. Falls nicht anders vermerkt, bezeichnet Ω ein (Teil-)Gebiet des $\mathbf{R}^d$, wobei generell $d \in \{1, 2, 3\}$ gilt. Die polygonale Berandung von Ω wird mit $\partial\Omega$ oder Γ bezeichnet. Hierbei ist stets davon auszugehen, dass $\partial\Omega$ hinreichend glatt ist. Der Abschluss von Ω wird mit $\bar{\Omega} := \Omega \cup \partial\Omega$ bezeichnet. Weiter definiert das *Lebesque-Maß* $|\Omega| = \mathcal{L}^d(\Omega)$ den intuitiven Begriff des Volumens (für $d = 3$ bzw. der Fläche für $d = 2$ und der Länge für $d = 1$) des Gebietes Ω. Als Integrationssymbol für das Lebesque-Maß auf $\Omega \subseteq \mathbf{R}^d$ wird $\mathrm{d}x := \mathrm{d}x^1 \cdots \mathrm{d}x^d$ verwendet. Weiter repräsentiert $\mathrm{d}\Gamma$ ein Element der Hyperfläche $\Gamma = \partial\Omega$ und $\mathrm{d}t$ das Integrationssymbol für das Lebesque-Maß auf einem vorgegebenen Zeitintervall über dem Körper der positiven, reellen Zahlen.

Für *totale* bzw. für *partielle Ableitungen* von skalaren Funktionen $u(t)$, $u(x)$ bzw. $u(x, t)$ werden die Schreibweisen $\mathrm{d}_t u := \frac{\mathrm{d}u}{\mathrm{d}t}$, $\partial_t u := \frac{\partial u}{\partial t}$ und $\partial_{x^i} u := \frac{\partial u}{\partial x^i}$ verwendet. Der Gradient bzw. die Divergenz werden auf der Basis des *Nabla-Operators* $\nabla := (\partial_{x^1}, ..., \partial_{x^n})^\mathsf{T} \in \mathbf{R}^n$ notiert. Im Detail gilt $\nabla u = (\partial_{x^1} u, ..., \partial_{x^n} u)^\mathsf{T} \in \mathbf{R}^n$ für Funktionen $u : \mathbf{R}^n \to \mathbf{R}$ und $\nabla \cdot u = \sum_{i=1}^n \partial_{x^i} u^i$ für Funktionen $u : \mathbf{R}^n \to \mathbf{R}^n$. Die Kombination von Divergenz- und Gradientenbildung resultiert im *Laplace-Operator* – für eine skalarwertige Funktion u gilt $\nabla \cdot (\nabla u) = \sum_{i=1}^n \partial_{x^i x^i} u =: \Delta u$. Weiter wird die Ableitung einer skalaren Funktion u in Richtung einer vektoriellen Größe $v \in \mathbf{R}^n$, $\|v\|_2 = 1$, als $\partial_v u := \nabla u \cdot v$ notiert.

Der Träger einer Funktion $u : \mathbf{R}^d \to \mathbf{R}$ wird als $\mathrm{supp}(u) := \overline{\{x \in \mathbf{R}^d : u(x) \neq 0\}}$ angegeben. Funktionen werden als (hinreichend) glatt bezeichnet, wenn sie, in Abhängigkeit der jeweiligen Situation, hinreichend viele (stetige) Ableitungen besitzen. Präzisere Einschränkungen bzgl. Stetigkeit werden wie folgt vorgeschrieben: Der Raum stetiger

[7] Natürlich ist in vielen Fällen eine Verallgemeinerung auf komplexwertige Zahlen möglich. Da diese aber für die entwickelte Algorithmik irrelevant sind, wird auf deren Behandlung weitestgehend verzichtet.

Funktionen auf einem Gebiet $\Omega \subset \mathbf{R}^d$ wird als $C(\Omega) := \left\{u : \bar{\Omega} \to \mathbf{R} : u \text{ ist stetig auf } \bar{\Omega}\right\}$ notiert. Für k-mal stetig differenzierbare Funktionen ist der zugehörige Funktionenraum durch $C^k(\Omega) := \left\{u : \bar{\Omega} \to \mathbf{R} : \partial_{x^i} u \in C^{k-1}(\Omega), i = 1, ..., n\right\}$ erklärt.

So viel zu diesem einleitenden Überblick. Weitere Details werden im Verlauf der Arbeit in Bezug zum Kontext entwickelt. Für eine Übersicht sei erneut auf das Notationsverzeichnis auf S. 175 f. verwiesen.

2.2 Lösung dünnbesetzter linearer Gleichungssysteme

In diesem Abschnitt wird ein Einblick in Verfahren zur Lösung *dünnbesetzter*, *linearer Gleichungssysteme* der Form

$$Ax = b, \quad A \in \mathbf{R}^{n \times n}, b \in \mathbf{R}^n, n \in \mathbf{N}, \tag{2.1}$$

gegeben[8]. Die betrachteten Verfahren sind teilweise auf bestimmte Klassen von Matrizen A beschränkt. Die Räume, die sich in diesem Zusammenhang als von Bedeutung herausstellen, sind der Raum der symmetrischen Matrizen $\mathfrak{S}^n := \left\{A \in \mathbf{R}^{n \times n} : A = A^\mathsf{T}\right\}$ und der Raum der symmetrisch positiv definiten Matrizen

$$\mathfrak{S}^{n,+} := \left\{A \in \mathfrak{S}^n : x^\mathsf{T} A x > 0 \; \forall x \in \mathbf{R}^n \backslash \{0\}\right\}.$$

Zusätzlich werden *positiv definite Matrizen* als $A \succ 0$ und *positiv semi-definite Matrizen* (d. h. $x^\mathsf{T} A x \geq 0 \; \forall x \in \mathbf{R}^n \backslash \{0\}$) als $A \succcurlyeq 0$ notiert.

Zur Lösung von (2.1) werden lediglich iterative Verfahren, die eine Sequenz an Näherungslösungen

$$x_{l+1} = x_l + \mu_l s_l \quad \forall l \in \mathbf{N} \tag{2.2}$$

mit $s_l \in \mathbf{R}^n$ und $\mu_l \geq 0$, für ein beliebiges $x_0 \in \mathbf{R}^n$ bestimmen, betrachtet. Für (2.2) soll o. B. d. A. für s_l die Forderung $\|s_l\|_2 = 1$ gelten. Wie aus (2.2) ersichtlich ist, wird der Iterationsindex $l \in \mathbf{N}$ im Folgenden rechts unten an die Variable geschrieben.

2.2.1 Grundlegendes

Die betrachteten Verfahren gehören zur Klasse der *Projektionsverfahren*. Für die allgemeine Definition eines Projektionsverfahrens werden zunächst nicht weiter spezifizierte, lineare Unterräume $\{0\} \subset \mathfrak{L}_1 \subset \mathfrak{L}_2 \subset ... \subset \mathbf{R}^n$ eingeführt. Weiter sei

$$\mathfrak{X}^\perp := \{x \in \mathbf{R}^n : \langle x, \tilde{x} \rangle_2 = 0 \text{ für alle } \tilde{x} \in \mathfrak{X}\}$$

[8] Die folgenden Ausführungen sind im Wesentlichen [157] entnommen.

für einen beliebigen Unterraum $\mathfrak{X} \subset \mathbf{R}^n$. Mit Hilfe dieser Ansatzräume ist der Begriff des Projektionsverfahrens wie folgt erklärt (vgl. [196, S. 134]):

Definition 1 Seien die l-dimensionalen, linearen Unterräume $\mathfrak{L}_l$ und $\mathfrak{D}_l$ des $\mathbf{R}^n$ gegeben. Ein *Projektionsverfahren* zur Lösung von (2.1) ist eine Methode zur Berechnung einer Näherungslösung x_l in einem affinen Unterraum $x_0 + \mathfrak{L}_l$, unter Berücksichtigung der *Orthogonalitätsbedingung*

$$r_l := (b - Ax_l) \in \mathfrak{D}_l^{\perp} \tag{2.3}$$

bzgl. des *Residuums* r_l. ♣

Anmerkung 1 Gilt $\mathfrak{L}_l = \mathfrak{D}_l$ ist der *Residuenvektor* r_l orthogonal zu $\mathfrak{L}_l$, es liegt ein *orthogonales Projektionsverfahren* vor und (2.3) heißt *Galerkin-Bedingung* [157, S. 112]. Für $\mathfrak{L}_l \neq \mathfrak{D}_l$ ergibt sich ein *schiefes Projektionsverfahren* – (2.3) heißt *Petrov-Galerkin-Bedingung* [157, S. 112]. ◇

Im Folgenden werden effiziente numerische Verfahren eingeführt, die es erlauben, eine Approximation $x_l \in x_0 + \mathfrak{L}_l$ an die Lösung $x^\star = A^{-1}b$ des Systems (2.1), ausgehend von einem beliebigen Startvektor $x_0 \in \mathbf{R}^n$, in einer Folge von affinen Unterräumen

$$\mathfrak{V}_0 = \{x_0\} \subset \mathfrak{V}_1 = x_0 + \mathfrak{L}_1 \subset \cdots \subset \mathfrak{V}_n = x_0 + \mathfrak{L}_n$$

bereitzustellen. Diese basieren auf einer, zur Lösung des Gleichungssystems (2.1) äquivalenten, Minimierungsaufgabe (vgl. [201, S. 514, Satz 11.20]):

Satz 1 *Sei $A \in \mathfrak{S}^{n,+}$ und $b \in \mathbf{R}^n$, dann gilt*

$$x^\star = \arg\min_{x \in \mathbf{R}^n} \left\{ \mathcal{F}(x) := \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} a_{jk} x_j x_k - \sum_{j=1}^{k} b_j x_j \right\} \quad \Leftrightarrow \quad Ax^\star = b. \tag{2.4}$$

Beweis Der Beweis ergibt sich unmittelbar aus den notwendigen und hinreichenden Optimalitätsbedingungen für (2.4) und ist in [201, S. 514] zu finden. □

Satz 1 liefert die Motivation, anstelle einer direkten Inversion des Systems (2.1), die Lösung $x^\star \in \mathbf{R}^n$, basierend auf einer äquivalenten, iterativen Minimierung des quadratischen Funktionals $\mathcal{F}$ zu bestimmen. Diese Idee führt zur Iterationsvorschrift (2.2).

2.2.2 Schrittweitenbestimmung

Bei vorgegebener Suchrichtung $s_l \in \mathbf{R}^n$ ist die Schrittweite $\mu_l \geq 0$, wobei $\mu_l = 0$ impliziert, dass x_l der Lösung $A^{-1}b$ von (2.1) entspricht, so zu wählen, dass μ_l das Funktional $\mathcal{F}$ ausgewertet an der Stelle $x_l + \mu_l s_l$ minimiert. Dies entspricht einer (exakten) *Liniensuche*

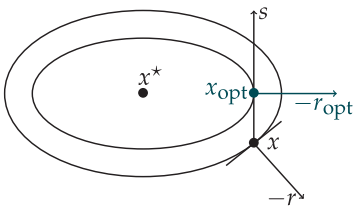


Abbildung 2.1 Geometrische Interpretation eines Iterationsschrittes (in Anlehnung an [201, S. 515, Abb. 11.7]). Die Niveaulinien $\{x \in \mathbf{R}^2 : \mathcal{F}(x) = c, c > 0\} \subset \mathbf{R}^2$ sind für ein Problem der Ordnung 2 konzentrische Ellipsen. Die Residuenvektoren stehen senkrecht zur Niveaulinie im Punkt $x \in \mathbf{R}^2$ und $x_{\text{opt}} \in \mathbf{R}^2$. Die Suchrichtung $s \in \mathbf{R}^2$ ist eine Tangente an die Niveaulinie durch x_{opt}.

(vgl. auch Abschnitt 2.3) entlang einer Geraden durch x_l, bei vorgegebener Richtung s_l. Um das zugehörige Optimalitätskriterium herzuleiten, wird für vorgegebene $x \in \mathbf{R}^n$ und $s \in \mathbf{R}^n$ die univariate Funktion $f_{x,s} : \mathbf{R} \to \mathbf{R}, f_{x,s}(\mu) := \mathcal{F}(x + \mu s)$, eingeführt. Die gesuchte, optimale Schrittweite $\mu^\star \in \mathbf{R}$ liefert folgendes Lemma (vgl. [157, S. 118, Lemma 4.51]):

Lemma 1 *Sei $A \in \mathfrak{S}^{n,+}$, $x \in \mathbf{R}^n$ und $s \in \mathbf{R}^n \setminus \{0\}$ gegeben. Dann gilt*

$$\mu^\star := \arg\min_{\mu \in \mathbf{R}} f_{x,s}(\mu) = \frac{\langle r, s\rangle_2}{\langle As, s\rangle_2} \tag{2.5}$$

mit $r := b - Ax$.

Beweis Die Beweisführung ist in [157, S. 118] nachzulesen. □

Eine geometrische Interpretation eines Iterationsschrittes liefert folgender Satz (vgl. [201, S. 515, Satz 11.21]):

Satz 2 *Im Minimalpunkt x_{opt} bzgl. des Optimalitätskriteriums* (2.5) *gilt $s \perp r_{\text{opt}}$ für $r_{\text{opt}} = b - Ax_{\text{opt}}$.*

Beweis Der Beweis ist in [201, S. 515] nachzulesen. □

Satz 2 ist in Abb. 2.1 illustriert.

Durch Lemma 1 ist ein Basislöser für eine vorgegebene Folge an Suchrichtungen $\{s_l\}_{l \in \mathbf{N}_0}$ aus $\mathbf{R}^n \setminus \{0\}$ bereitgestellt. Die Verfahrensvorschrift lautet in Analogie zu (2.2)

$$x_{l+1} = x_l + \mu_l s_l = x_l + \frac{\langle r_l, s_l\rangle_2}{\langle As_l, s_l\rangle_2} s_l, \quad l \in \mathbf{N}_0, \tag{2.6}$$

mit $r_l = b - Ax_l$ (vgl. [157, S. 118]).

2.2.3 Berechnung der Suchrichtung

Bis jetzt ist lediglich die Schrittlänge $\mu_l \geq 0$ festgelegt. Eine natürliche Wahl für die Suchrichtung $s_l \in \mathbf{R}^n \setminus \{0\}$ ist der negative Gradient des Zielfunktionals $\mathcal{F}$. Dieser ist gleich dem Residuum r_l (siehe bspw. [201, S. 514]). Dieser Ansatz führt auf die *Methode des steilsten Abstiegs* (MSA (engl. *steepest descent method*)). Einsetzen der Suchrichtung $s_l = -\nabla\mathcal{F}(x_l) = b - Ax_l = r_l$ in die Verfahrensvorschrift (2.6) liefert (vgl. [157, S. 119])

$$x_{l+1} = \begin{cases} x_l + \frac{\|r_l\|_2^2}{\langle Ar_l, r_l\rangle_2} r_l & \text{für } r_l \neq 0, \\ x_l & \text{für } r_l = 0. \end{cases}$$

Aus $\mu_l = \|r_l\|_2^2 / \langle Ar_l, r_l\rangle_2$ folgt die Orthogonalität zweier konsekutiver Residuenvektoren [157, S. 123]. Nach Def. 1 entspricht die MSA damit in jedem Iterationsschritt einer orthogonalen Projektionsmethode mit $\mathfrak{L}_l = \operatorname{span}\{r_{l-1}\}$ [157, S. 124].

Wegen Satz 2 hängt die Konvergenzgeschwindigkeit der MSA entscheidend von der Geometrie der Niveaumenge $\{x \in \mathbf{R}^n : \mathcal{F}(x) = c, c > 0\} \subset \mathbf{R}^n$ und dem gewählten Startvektor $x_0 \in \mathbf{R}^n$ ab. Daher erweist sich die MSA in praktischen Anwendungen als ungeeignet. Wird allerdings eine an A angepasste Basis $\{s_i\}_{i=0}^{l}$, $l \in \{1, ..., n-1\} \subset \mathbf{N}$, des $\mathbf{R}^n$ verwendet, lässt sich der erforderliche Rechenaufwand entscheidend reduzieren. Dieser Zugang führt auf das Konzept A-orthogonaler Vektoren (vgl. [157, S. 125, Def. 4.60]):

Definition 2 Die Vektoren $s_i \in \mathbf{R}^n \setminus \{0\}$, $i = 0, ..., l$, heißen *paarweise konjugiert* oder *A-orthogonal*, falls für $A \in \mathbf{R}^{n\times n}$, $A \succ 0$,

$$\langle s_j, s_k\rangle_A = \langle As_j, s_k\rangle_2 \overset{A=A^\mathsf{T}}{=} \langle s_j, As_k\rangle_2 = 0 \quad \forall j, k \in \{0, ..., l\} \subset \mathbf{N}_0 \wedge j \neq k \tag{2.7}$$

gilt. ♣

Anmerkung 2 Für $A = \operatorname{diag}(1, ..., 1) \in \mathbf{R}^{n\times n}$ folgt $s_j \perp s_k$ $\forall j, k \in \{0, ..., l\} \subset \mathbf{N}_0$, $j \neq k$, und damit, dass A-Orthogonalität eine natürliche Generalisierung des Begriffes der Orthogonalität von Vektoren ist. ◇

Basierend auf $A \succ 0$ kann für ein A-orthogonales Vektorsystem $\{s_i\}_{i=0}^{l}$ die lineare Unabhängigkeit der Vektoren s_i gezeigt werden (siehe [157, S. 125]). Damit spannen sie eine bzgl. des *Energieskalarprodukts* $\langle \cdot, \cdot \rangle_A$ orthogonale Basis im $\mathbf{R}^n$ auf. Es lässt sich zeigen, dass ein Verfahren der Gestalt (2.6) mit A-orthogonalen (linear unabhängigen) Suchrichtungen s_l, für jedes $x_0 \in \mathbf{R}^n$ nach n Iterationen gegen die Lösung $x^\star = A^{-1}b$ des linearen Systems (2.1) konvergiert (vgl. bspw. [171, S. 103 ff.]). Insgesamt führt dies zum Begriff des *Verfahrens der konjugierten Richtungen*.

Das *Verfahren der konjugierten Gradienten* (CG-Verfahren (engl. *conjugate gradient method*)) [104] nutzt dieses Konzept aus[9]. Für die Konstruktion A-orthogonaler Suchrichtungen wird eine Modifikation des Gram-Schmidtschen Orthogonalisierungsverfahrens (siehe [24, S. 320]) verwendet. Als erste Näherung an eine geeignete Suchrichtung dient, analog zur MSA, der Residuenvektor $r_l = b - Ax_l$. Mit $s_0 := r_0$ berechnet sich die l-te Suchrichtung über (vgl. [157, S. 126])

$$s_l = r_l + \sum_{j=0}^{l-1} \beta_j s_j \quad \text{für } l = 1, ..., n-1. \tag{2.8}$$

Der Zerlegungskoeffizient β_j in (2.8) wird nun, im Sinne einer Gram-Schmidtschen Orthogonalisierung, dazu genutzt, die Konjugiertheit der Suchrichtung s_l zu erzwingen. Aus der geforderten A-Orthogonalität ergibt sich (vgl. [157, S. 126 f.])

$$0 = \langle s_l, s_i \rangle_A \overset{(2.8)}{=} \langle r_l, s_i \rangle_A + \sum_{j=0}^{l-1} \beta_j \langle s_j, s_i \rangle_A \overset{(2.7)}{=} \langle r_l, s_i \rangle_A + \beta_i \langle s_i, s_i \rangle_A \quad \Leftrightarrow \quad \beta_i = -\frac{\langle r_l, s_i \rangle_A}{\langle s_i, s_i \rangle_A}$$

und hiermit für (2.8)

$$s_l = r_l - \sum_{j=0}^{l-1} \frac{\langle r_l, s_j \rangle_A}{\langle s_j, s_j \rangle_A} s_j \quad \text{für } l = 1, ..., n-1. \tag{2.9}$$

Die Berechnungsvorschrift in (2.9) ist ineffizient, da für die Berechnung der l-ten Suchrichtung, alle Vektoren $\{s_i\}_{i=0}^{l-1}$ benötigt werden. Im ungünstigsten Fall wird der Speicherplatz für eine vollbesetzte Matrix der Dimension $n \times n$ benötigt. Für großmaßstäbliche Probleme ist dies eine impraktikable Forderung. Die Berechnungsvorschrift in (2.9) lässt sich allerdings entscheidend verbessern. Es stellt sich heraus (vgl. [157, S. 128 ff.]), dass sich (2.9) zu

$$s_l = r_l - \frac{\langle r_l, s_{l-1} \rangle_A}{\langle s_{l-1}, s_{l-1} \rangle_A} s_{l-1} = r_l + \frac{\langle r_l, r_l \rangle_2}{\langle r_{l-1}, r_{l-1} \rangle_2} s_{l-1} \quad \text{für } l = 1, ..., n-1 \tag{2.10}$$

vereinfachen lässt. Mit (2.10) ist eine effiziente Berechnungsvorschrift für das CG-Verfahren bereitgestellt. Der resultierende Algorithmus ist in Alg. 2.1 skizziert.

Ausgehend von dem Ansatzraum $\mathfrak{L}_l = \text{span}\{s_0, ..., s_{l-1}\} \subset \mathbf{R}^n$, $l = 1, ..., n-1$ ist das CG-Verfahren wegen $r_l \in \mathfrak{L}_l^\perp$ nach Def. 1 eine orthogonale Projektionsmethode. Es kann gezeigt werden (siehe bspw. [171, S. 109 ff.]), dass für eine Iterierte $x_l \in x_0 + \mathfrak{K}_l$ gilt, wobei

$$\mathfrak{K}_l := \text{span}\{r_0, Ar_0, ..., A^{l-1} r_0\} \subset \mathbf{R}^n$$

[9] In der Originalarbeit [104] wurde das CG-Verfahren als ein direktes Verfahren eingeführt. Erst in [184] wurde es als iteratives Verfahren zur Lösung dünnbesetzter, linearer Gleichungssysteme etabliert.

Algorithmus 2.1 CG-Verfahren (nach [171, S. 112, Alg. 5.2]).

choose $x_0 \in \mathbf{R}^n$, $\epsilon > 0$
initialise $r_0 \leftarrow b - Ax_0$, $s_0 \leftarrow r_0$, $l \leftarrow 0$
while $\|r_l\| > \epsilon$ **do**
  $\mu_l \leftarrow \frac{\langle r_l, r_l \rangle_2}{\langle s_l, s_l \rangle_A}$
  $x_{l+1} \leftarrow x_l + \mu_l s_l$
  $r_{l+1} \leftarrow r_l + \mu_l A s_l$
  $s_{l+1} \leftarrow r_l + \frac{\langle r_{l+1}, r_{l+1} \rangle_2}{\langle r_l, r_l \rangle_2} s_l$
  $l \leftarrow l + 1$
end while

ein sog. *Krylow-Unterraum* ist. Damit zählt das CG-Verfahren zur Klasse der orthogonalen *Krylow-Unterraum-Verfahren*. Weitere Verfahren dieser Klasse, die in der vorliegenden Arbeit nicht weiter betrachtet werden, sich aber generell für die Lösung der auftretenden Systeme eignen, sind das BiCG-Verfahren (bikonjugiertes CG-Verfahren), das stabilisierte BiCG-Verfahren (BiCGSTAB) oder das GMRES-Verfahren (engl. *generalized minimal residual*).

Eine Abschätzung für die Konvergenzgeschwindigkeit des CG-Verfahrens zeigt, dass die Konvergenz direkt von der sog. Konditionszahl der Matrix A abhängt (siehe bspw. [196, S. 214] oder [157, S. 132, Satz 4.65]). Der Begriff der Konditionszahl ist wie folgt erklärt (vgl. [157, S. 25, Def. 2.37]):

Definition 3 Für $A \in \mathbf{R}^{n \times n}$ regulär heißt $\mathrm{cond}_\alpha(A) := \|A\|_\alpha \|A^{-1}\|_\alpha$ *Konditionszahl* der Matrix A bzgl. der induzierten Matrixnorm $\|\cdot\|_\alpha$. ♣

Über ein äquivalente Umformulierung von (2.1) kann versucht werden, die Konditionszahl herabzusetzen. Diese Überlegung führt zum Konzept der *Vorkonditionierung*. Für nahezu alle großmaßstäblichen Anwendungen ist, bei Verwendung des CG-Verfahrens, eine Vorkonditionierung obligatorisch [205].

2.2.4 Vorkonditionierung

Wie im vorherigen Abschnitt gesehen, hängt die Konvergenzgeschwindigkeit numerischer Verfahren zur Lösung eines Systems der Gestalt (2.1) von der Konditionszahl der Koeffizientenmatrix A ab. Eine wichtige Abschätzung für den Wertebereich der Konditionszahl liefert das folgende Lemma (vgl. [157, S. 25, Lemma 2.38]):

Lemma 2 *Für $A \in \mathbf{R}^{n \times n}$ regulär ist* $\mathrm{cond}_\alpha(A)$ *nach unten beschränkt und es gilt* $\mathrm{cond}_\alpha(A) \geq 1 = \mathrm{cond}_\alpha(E)$.

Beweis Ein Beweis findet sich in [157, S. 25]. □

Daneben spielen die Spektraleigenschaften der Matrix A eine entscheidende Rolle. Die notwendigen Begrifflichkeiten in diesem Zusammenhang liefert die folgende Definition (vgl. [157, S. 20, Def 2.29]):

Definition 4 Es bezeichnet $\lambda \in \mathbf{R}$ den *Eigenwert* einer Matrix $A \in \mathbf{R}^{n\times n}$, $n \in \mathbf{N}$, falls ein Vektor $v \in \mathbf{R}^n \setminus \{0\}$ existiert, so dass $Av = \lambda v$. Hierbei ist v der einem Eigenwert λ zugehörige *Eigenvektor*. Das *Spektrum* von A ist durch

$$\sigma(A) := \{\lambda \in \mathbf{R} : \det(A - \lambda E) = 0, E = \operatorname{diag}(1, ..., 1) \in \mathbf{R}^{n\times n}\}$$

erklärt. Weiter heißt $\rho(A) := \max_{1\leq i\leq n}|\lambda_i|, \lambda_i \in \sigma(A)$, *Spektralradius* der Matrix A. ♣

Anmerkung 3 Werden die Eigenwerte $\lambda_i \in \sigma(A)$, $A \in \mathbf{R}^{n\times n}$, $i = 1, ..., n$, in der *Spektralmatrix* $\Lambda := \operatorname{diag}(\lambda_1, ..., \lambda_n) \in \mathbf{R}^{n\times n}$ und die Eigenvektoren $v_i \in \mathbf{R}^n \setminus \{0\}$, $i = 1, ..., n$, spaltenweise in der *Modalmatrix* $Y := (v_1, ..., v_n) \in \mathbf{R}^{n\times n}$ zusammengefasst, lässt sich das Eigenwertproblem als Suche nach der zugehörigen Modal- und Spektralmatrix und damit in Form einer *Eigenwertzerlegung* $A = Y\Lambda Y^{-1}$ darstellen. ◇

Basierend auf der *Spektralnorm* (siehe [201, S. 51 f.])

$$\|A\|_2 := \max_{\|x\|_2=1} \|Ax\|_2 = \max_{\|x\|_2=1} \{(Ax)^\mathsf{T}(Ax)\}^{1/2} = \sqrt{\rho(AA^\mathsf{T})}$$

lässt sich ein Zusammenhang zwischen den Eigenwerten und der Konditionszahl (*Spektralkondition*) einer Matrix herstellen (vgl. [157, S. 28 f., Korollar 2.34]):

Lemma 3 *Sei $A \in \mathfrak{S}^{n,+}$, dann gilt mit $\lambda_i \in \sigma(A)$, $i = 1, ..., n$, die Identität*

$$\operatorname{cond}_2(A) = \frac{\max_{1\leq i\leq n} |\lambda_i|}{\min_{1\leq i\leq n} |\lambda_i|}.$$

Beweis Der Beweis der Behauptung folgt unmittelbar aus den in [201, S. 52] skizzierten Zusammenhängen. □

Ziel dieses Abschnittes ist es, äquivalente Umformulierungen für (2.1) bereitzustellen, die potenziell eine Verringerung der Konditionszahl des Systems zur Folge haben. Derartige Techniken werden als *Vorkonditionierung* bezeichnet. Folgende Definition präzisiert diesen Begriff (vgl. [157, S. 200, Def. 5.1]):

Definition 5 Sei das System aus (2.1), $y \in \mathbf{R}^n$ und die regulären Matrizen $P_L \in \mathbf{R}^{n\times n}$ und $P_R \in \mathbf{R}^{n\times n}$ gegeben. Das zu (2.1) zugehörige, vorkonditionierte System ist durch $P_L^{-1}AP_R^{-1}y = P_L^{-1}b$, $x = P_R^{-1}y$ erklärt. ♣

Die Matrizen P_L und P_R heißen *Links-* bzw. *Rechtspräkonditionierer* [157, S. 200]. Gilt $P_L \neq E_n \wedge P_R \neq E_n$ mit $E_n := \mathrm{diag}(1, ..., 1) \in \mathbf{R}^{n \times n}$, dann heißt das System *beidseitig vorkonditioniert*, wobei sich P_L und P_R in der Regel aus der Faktorisierung $P = P_L P_R$ errechnen [157, S. 200]. Entspricht jeweils nur einer der beiden Matrixoperatoren der Einheit, heißt das System *links-* bzw. *rechtspräkonditioniert* [157, S. 200].

Die Äquivalenz der Linksvorkonditionierung zu (2.1) folgt aus der Regularität der Matrix P_L. Wegen $\ker(P_L^{-1}) = \{0\}$ gilt $P_L^{-1}(Ax - b) = 0 \Leftrightarrow Ax - b = 0$. Für die Rechtsvorkonditionierung ist aus $(AP_R^{-1})y = AP_R^{-1}P_R x = Ax$ die Äquivalenz zu (2.1) ersichtlich.

Ziel einer Vorkonditionierung ist es, wie eingangs erwähnt, Matrizen P_L und P_R in der Gestalt zu bestimmen, dass $\mathrm{cond}_\alpha(P_L^{-1}AP_R^{-1}) \ll \mathrm{cond}_\alpha(A)$ bzw. die Eigenwerte der Matrix eine bessere Verteilung aufweisen, als dies für die orginäre Koeffizientenmatrix A der Fall ist. Die Beschränktheit der Konditionszahl $\mathrm{cond}_\alpha(A)$ liefert eine simple Bedingung für eine optimale Wahl der Präkonditionierung: Aus Lemma 2 ergibt sich unmittelbar die Forderung $P_L^{-1}AP_R^{-1} \approx E_n$ und damit, dass bei einer einseitigen Präkonditionierung die rechte bzw. linke Präkonditioniermatrix eine möglichst gute Approximation an A darstellen sollte. Aus Effizienzgründen ist es zweckmäßig, dass die Präkonditionierer einfach zu berechnen sind und einen geringen Speicherplatzbedarf erfordern.

Es existieren unterschiedliche Varianten der Vorkonditionierung, die bspw. auf einer Äquilibrierung (Skalierung), einer unvollständigen Zerlegung, auf Mehrgitterverfahren oder auf *Splitting*-Verfahren basieren (siehe bspw. [157, S. 200 ff.]). Die vorliegende Arbeit beschränkt sich auf letztere.

2.2.4.1 Splitting-assoziierte Vorkonditionierung

Splitting-Verfahren (siehe [157, S. 65 ff. bzw. 196, S. 95 ff.]) sind, zu den vorgestellten Krylow-Unterraum-Verfahren alternative, iterative Verfahren zur Lösung eines Gleichungssystems der Gestalt (2.1). Sie basieren auf einer Zerlegung der Gestalt

$$A = B + (A - B), \quad A, B \in \mathbf{R}^{n \times n},$$

so dass sich das zu (2.1) äquivalente System $Bx = (B - A)x + b$, $x \in \mathbf{R}^n$, $b \in \mathbf{R}^n$, ergibt. Für eine reguläre Matrix B resultiert die *Fixpunktform*

$$x = \underbrace{B^{-1}(B - A)}_{=: M} x + B^{-1}b.$$

Um eine schnelle Konvergenz zu garantieren, sollte B einfach zu invertieren sein und eine möglichst gute Approximation der Matrix A darstellen. Dies folgt unmittelbar aus $M = B^{-1}(B - A) = E - B^{-1}A$. Diese Beobachtung legt es nahe, für die Konstruktion eines Präkonditionierers, die in *Splitting*-Verfahren auftretende Matrix B, zu verwenden [157, S. 207, Def. 5.9]:

Definition 6 Sei die reguläre Matrix $B \in \mathbf{R}^{n\times n}$, $M \in \mathbf{R}^{n\times n}$ und $x_l \in \mathbf{R}^n$, $l = 1, 2, \ldots$ gegeben und $x_l = Mx_{l-1} + B^{-1}b$ das zugehörige *Splitting*-Verfahren, dann bezeichnet $P = B$ den, mit dem *Splitting*-Verfahren assoziierten, Vorkonditionierer. ♣

Analog zu Def. 6 ergibt sich, mit der Standardzerlegung $A = D - L - R$, bspw. der zum Jacobi-Verfahren analoge Vorkonditionierer (*Jacobi-Vorkonditionierer*) zu $P_J = D$, der zum Gauss-Seidel-Verfahren analoge Vorkonditionierer (*Gauss-Seidel-Vorkonditionierer*) zu $P_{GS} = D + L$ und der zum Überrelaxations-Verfahren analoge Vorkonditionierer (*SOR-Vorkonditionierer*) zu $P_{SOR} = \omega(D + \omega L)$, $\omega \in (0, 2)$. Hierbei sind, bei vorliegender, regulärer Koeffizientenmatrix $A = (a_{ij})_{i,j=1}^{n,n}$, $a_{ij} \in \mathbf{R}$, $a_{ii} \neq 0$, die Matrizen $D \coloneqq \operatorname{diag}(a_{11}, \ldots, a_{nn}) \in \mathbf{R}^{n\times n}$,

$$L = (l_{ij})_{i,j=1}^{n,n} \quad \text{mit} \quad l_{ij} \coloneqq \begin{cases} a_{ij}, & \text{falls } i > j, \\ 0, & \text{sonst,} \end{cases}$$

und

$$R = (r_{ij})_{i,j=1}^{n,n} \quad \text{mit} \quad r_{ij} \coloneqq \begin{cases} a_{ij}, & \text{falls } i < j, \\ 0, & \text{sonst.} \end{cases}$$

2.2.4.2 Präkonditioniertes CG-Verfahren

In diesem Abschnitt wird die Vorkonditionierung des in Abschnitt 2.2.3 vorgestellten CG-Verfahrens besprochen. Dies führt zum Begriff des *präkonditinionierten CG-Verfahrens* (PCG-Verfahren; engl. *preconditioned conjugate gradient method*) (siehe bspw. [157, S. 216 ff.]).

Eine Grundvorraussetzung, die an das System (2.1), im Rahmen der Herleitung des CG-Verfahrens, gestellt wurde, ist $A \in \mathfrak{S}^{n,+}$. Dieser Forderung gilt es, bei der Vorkonditionierung Rechnung zu tragen. Durch eine *Kongruenztransformation* $C^{-1}AC^{-\mathsf{T}} =: \tilde{A}$, mit $C \in \mathbf{R}^{n\times n}$ regulär, wird sichergestellt, dass die Eigenschaften der Koeffizientenmatrix A erhalten bleiben. Hieraus ergibt sich unmittelbar die zu (2.1) äquivalente Umformung $C^{-1}AC^{-\mathsf{T}}C^{\mathsf{T}}x = C^{-1}b$, die einer beidseitigen Vorkonditionierung (vgl. Def. 5), mit dem zweckmäßigen Ziel $\operatorname{cond}_2(C^{-1}AC^{-\mathsf{T}}) \ll \operatorname{cond}_2(A)$, entspricht. Die Ähnlichkeitstransformation[10] $C^{-\mathsf{T}}\tilde{A}C^{\mathsf{T}} =: \tilde{C}$ liefert eine geeignete Matrix C. Es gilt [201, S. 525]

$$C^{-\mathsf{T}}\tilde{A}C^{\mathsf{T}} = C^{-\mathsf{T}}C^{-1}AC^{-\mathsf{T}}C^{\mathsf{T}} = C^{-\mathsf{T}}C^{-1}A = (CC^{\mathsf{T}})^{-1}A \overset{P\coloneqq CC^{\mathsf{T}}}{=} P^{-1}A$$

mit $P \in \mathfrak{S}^{+,n}$. Aus der Ähnlichkeit von $\tilde{C}$ und $\tilde{A}$ folgt, mit Lemma 3, die Identität $\operatorname{cond}_2(\tilde{A}) = \operatorname{cond}_2(P^{-1}A)$ und damit, nach Lemma 2, erneut die Forderung, dass P eine möglichst gute Approximation an A darstellen sollte.

[10] Siehe [24, S. 321].

Mittels elementarer Umformungen kann gezeigt werden, dass für das PCG-Verfahren lediglich eine Vorkonditionierung über die Matrix P erforderlich ist (siehe [201, S. 526 ff.]). Es muss keine Zerlegung $P = CC^\mathsf{T}$ bestimmt werden. Infolgedessen können die Vorkonditionierer aus obigem Abschnitt direkt verwendet werden.

Für P_J folgt $C = \operatorname{diag}(\sqrt{a_{11}}, ..., \sqrt{a_{nn}})$ und damit $\tilde{A} = C^{-1}AC^{-\mathsf{T}} = C^{-1}AC^{-1} = L + E + L^\mathsf{T}$ (vgl. [201, S. 528]). Letzteres Verfahren wird exklusiv verwendet.

2.3 Numerische Optimierung

Der folgende Abschnitt wird einen Einblick in Begrifflichkeiten und Verfahren der numerischen Optimierung liefern. Die Ausführungen basieren im Wesentlichen auf [171].

2.3.1 Grundlegendes

Im Rahmen dieser Arbeit werden lediglich ableitungsbasierte iterative Verfahren für unrestringierte Optimierungsaufgaben besprochen. Die betrachteten Probleme sind von der Gestalt

$$\min_{x \in \mathbf{R}^n} f(x), \tag{2.11}$$

wobei $f : \mathbf{R}^n \to \mathbf{R}$ eine hinreichend glatte Funktion ist. Generell ist, bei der Lösung der Optimierungsaufgabe (2.11), wenig über den globalen Verlauf von f bekannt [171, S. 11]. Meist liegt der Wert der Funktion bzw. ihrer Ableitung lediglich an einigen wenigen Punkten $x_l \in \mathbf{R}^n$, $l = 0, 1, 2, ...$, vor. Die Auswertung von f (und der Ableitungen von f) an einer Stelle x_l ist typischerweise rechen- und damit zeitintensiv [171, S. 11]. Deshalb sollte sie so selten wie möglich erfolgen.

Den betrachteten Verfahren ist gemein, dass sie eine Iterationsfolge[11] $x_l \in \mathbf{R}^n$ mit vorgegebenem Startwert $x_0 \in \mathbf{R}^n$ erzeugen. Diese hat zum Ziel, dass die Werte von f monoton fallen, d. h. $f(x_{l+1}) \leq f(x_l)\ \forall l \in \mathbf{N}$. Die Aktualisierungsvorschrift ist von der Gestalt

$$x_{l+1} = x_l + \mu_l s_l, \quad \mu_l \in \mathbf{R}_0^+, \quad l = 0, 1, \ldots. \tag{2.12}$$

In einem ersten Schritt wird die Suchrichtung $s_l \in \mathbf{R}^n$ bestimmt. Dann wird entlang dieser vorgegebenen Richtung nach einer geeigneten Iterierten x_{l+1} (ausgehend von der aktuellen Iterierten x_l) gesucht. Optimierungsalgorithmen dieser Form werden als *Liniensuchverfahren* (engl. *line search method*) bezeichnet (vgl. bspw. [171, S. 19 ff.] bzw. [171, S. 30 ff.]).

[11] Der Iterationsindex $l \in \mathbf{N}_0$ wird, wie schon in Abschnitt 2.2, unten rechts an die Variable geschrieben.

Vor dem Einstieg in die Algorithmik werden einige Begrifflichkeiten der numerischen Optimierung nähergebracht: Unter dem Begriff *globaler Minimierer* wird diejenige Lösung $x^\star$ verstanden, die $f(x^\star) \leq f(x) \ \forall x \in \mathrm{dom}(f)$ erfüllt [171, S. 12]. Gilt dies nur für eine Umgebung $\mathfrak{U}_\epsilon(x) \coloneqq \{\tilde{x} \in \mathbf{R}^n : \|x - \tilde{x}\|_2 \leq \varepsilon\}$, $\varepsilon > 0$, des Urbildes der Funktion f, d. h. $f(x^\star) \leq f(x) \ \forall x \in \mathfrak{U}_\varepsilon(x^\star) \subset \mathrm{dom}(f)$, handelt es sich um einen *lokalen Minimierer* [171, S. 12]. Für die hier betrachteten Verfahren kann lediglich sichergestellt werden, dass das berechnete Optimum $x^\star$ ein lokaler Minimierer der Funktion f ist. Ist dieser anhand noch festzulegender Abbruchkriterien (siehe Abschnitt 2.3.4) identifiziert, terminiert die Optimierung.

Im weiteren Verlauf wird davon ausgegangen, dass die berechnete Suchrichtung $s_l \in \mathbf{R}^n$ eine *Abstiegsrichtung* ist, so dass für $f \in C^1(\mathbf{R}^n)$, $d_l \coloneqq \nabla f(x_l)$, $d_l \in \mathbf{R}^n$,

$$s_l^\mathsf{T} d_l < 0 \tag{2.13}$$

gilt [171, S. 30]. Der schematische Ablauf eines Liniensuchverfahrens ist in Alg. 2.2 dargestellt. Wie sich zeigen wird, liegen die Unterschiede der Verfahren in den Berechnungsvorschriften für die Schrittweite $\mu_l > 0$ (die sog. *Liniensuche*) und die Suchrichtung $s_l \in \mathbf{R}^n$.

Algorithmus 2.2 Schematischer Ablauf eines Liniensuchverfahrens (modifiziert aus [5, S. 69, Verfahren 4.2.1]).

initialise $x_0 \in \mathbf{R}^n$, $l \leftarrow 0$
while convergence criteria not satisfied **do**
  compute $f(x_l)$, $\nabla f(x_l)$ (possibly $\nabla^2 f(x_l)$)
  compute $s_l \in \mathbf{R}^n$ s. t. (2.13)
  compute $\mu_l > 0$ s. t. $f(x_l + \mu_l s_l) < f(x_l)$
  $x_{l+1} \leftarrow x_l + \mu_l s_l$
  $l \leftarrow l + 1$
end while

2.3.2 Schrittweitenbestimmung

Die Forderung nach einer optimalen Schrittweite $\mu^\star > 0$, für eine vorgegebene Suchrichtung $s_l \in \mathbf{R}^n$, führt auf die Idee, zu jeder Iteration, das in μ_l univariate Teilproblem [171, S. 31]

$$\mu^\star = \underset{\mu \in \mathbf{R}^+}{\arg\min} \{\psi(\mu) \coloneqq f(x_l + \mu s_l)\} \tag{2.14}$$

mit $\psi : (0, \infty) \to \mathbf{R}$ zu lösen[12]. Einer effizienten Implementierung wegen wird (2.14) allerdings nicht exakt, sondern nur näherungsweise gelöst [171, S. 31] – d. h.

$$\mu_l \approx \arg\min_{\mu \in \mathbf{R}^+} \{\psi(\mu) := f(x_l + \mu s_l)\}.$$

Es werden zusätzliche Bedingungen an die Iterierte x_{l+1} gestellt, um eine approximative Lösung zu ermöglichen. Die zugehörige Schrittweite μ_l hat diese Bedingungen zu erfüllen.

Eine Möglichkeit ist es, für $f \in C^1(\mathbf{R}^n)$ einen hinreichenden Abstieg des Funktionswertes, im Sinne der *Armijo-Bedingung* [171, S. 33]

$$f(x_l + \mu_l s_l) \leq f(x_l) + \beta \mu_l d_l^\mathsf{T} s_l, \tag{2.15}$$

zu fordern. Hierbei wird $\beta \in (0, 1)$ gewählt. Ist (2.15) erfüllt, dann impliziert (2.13) [99, S. 138]

$$f(x_l) + \beta \mu_l d_l^\mathsf{T} s_l < f(x_l) \quad \overset{(2.15)}{\Rightarrow} \quad \underbrace{f(x_l + \mu_l s_l)}_{= f(x_{l+1})} < f(x_l),$$

so dass (2.15) die geforderte Sequenz monoton fallender Funktionswerte erzeugt. Die Abstiegsbedingung (2.15) ist allerdings kein Garant für eine schnelle Konvergenz, da (2.15) für hinreichend kleine β stets erfüllt ist [165]. Um eine Mindestschrittweite zu garantieren, muss eine zusätzliche Bedingung an die Abstiegsrichtung gestellt werden. Dies führt auf die sog. *Wolfe-Bedingungen* [171, S. 34]

$$\text{(W1)} \quad f(x_l + \mu_l s_l) \leq f(x_l) + \beta \mu_l d_l^\mathsf{T} s_l, \tag{2.16.a}$$

$$\text{(W2)} \quad s_l^\mathsf{T} \nabla f(x_l + \mu_l s_l) \geq \lambda s_l^\mathsf{T} d_l^\mathsf{T}, \tag{2.16.b}$$

mit $0 < \beta < \lambda < 1$. Die Bedingung (W1) in (2.16.a) entspricht der Armijo-Bedingung aus (2.15). Die Bedingung (W2) in (2.16.b) heißt *Krümmungsbedingung*. Typische Werte für die Konstanten in (2.16) sind $\beta = 1 \times 10^{-4}$ und $\lambda \in [1 \times 10^{-1}, 9 \times 10^{-1}]$ [171, S. 33 ff.].

Basiert die Liniensuche auf den Bedingungen (2.16) wird das Verfahren als *Wolfe-Powell-Schrittweitenverfahren* bezeichnet (vgl. bspw. [5, S. 95 ff.]). *Strenge Wolfe-Bedingungen* liegen vor, wenn anstelle von (2.16.b) die Krümmungsbedingung

$$|s_l^\mathsf{T} \nabla f(x_l + \mu_l s_l)| \geq \lambda |s_l^\mathsf{T} d_l|$$

verwendet wird [171, S. 34].

[12] Die Einschränkung auf strikt positive μ folgt aus (2.13).

Algorithmus 2.3 Armijo-Liniensuche als *Backtracking*-Verfahren (nach [171, S. 37]).

choose $\mu_I > 0, \beta \in (0,1), \rho \in (0,1)$
$\mu_{LS} \leftarrow \mu_I$
while $f(x_l + \mu_{LS} s_l) \geq f(x_l) + \beta \mu_{LS} d_l^\mathsf{T} s_l$ **do**
  $\mu_{LS} \leftarrow \rho \mu_{LS}$
end while
$\mu_l \leftarrow \mu_{LS}$

Die Krümmungsbedingung (W2) erzeugt einen zusätzlichen Rechenaufwand. Sowohl der Wert der Funktion f als auch der Gradient an der Stelle x_l liegen vor. Damit muss für die Beachtung der Bedingung (W1) die Funktion f in einem Durchlauf der Liniensuche lediglich an der Stelle $x_l + \mu_{LS} s_l$, $\mu_{LS} > 0$, ausgewertet werden. Wird zusätzlich die Abstiegsbedingung (W2) in (2.16.b) beachtet, ist darüber hinaus der Gradient an der Stelle $x_l + \mu_{LS} s_l$ zu bestimmen. Diese zusätzlichen Kosten können mit Hilfe eines *Backtracking-Verfahrens* vermieden werden (vgl. [171, S. 37]). Die Idee ist es, anstelle der Bedingung (W2) in (2.16.b), eine Annäherung an eine maximal zulässige Schrittweite, im Sinne der Armijo-Bedingung (2.16.a), ausgehend von einer sukzessiven Kontraktion (gesteuert über den *Kontraktionsfaktor* $\rho \in (0,1)$), zu bestimmen. Dieses Verfahren wird in der Literatur als *Armijo-Schrittweitenverfahren* (vgl. bspw. [171, S. 37 ff.]) bezeichnet. Alg. 2.3 stellt es schematisch vor. Die Wahl der initialen Schrittweite für die Liniensuche ($\mu_I > 0$) hängt von dem verwendeten Ansatz zur Berechnung der Suchrichtung s_l ab. Für Newton- und Quasi-Newton-Verfahren (siehe Abschnitt 2.3.3.2 bzw. Abschnitt 2.3.3.3) gilt $\mu_I = 1$. Für die praktische Anwendung ist die Kontraktion der Schrittweite auf eine geringe Anzahl an Durchläufen zu beschränken, da bei kleinen Schrittweiten der Unterschied zwischen den Iterierten zu einem frühen Zeitpunkt zu gering wird.

2.3.3 Berechnung der Suchrichtung

Die folgenden Ausführungen beantworten die Frage nach der Wahl einer geeigneten Suchrichtung für die Vorschrift (2.12). Die betrachteten Verfahren sind von der Gestalt [171, S. 31]

$$s_l = -B_l^{-1} d_l \quad \text{u. d. Nb.} \quad B_l \succ 0 \tag{2.17}$$

und unterscheiden sich damit lediglich in der Wahl der Matrix $B_l \in \mathbf{R}^{n \times n}$. Einleitend ist festzuhalten, dass für $B_l \succ 0$ die Bedingung (2.13) erfüllt ist, denn [171, S. 31]

$$s_l^\mathsf{T} d_l = -\underbrace{d_l^\mathsf{T} B_l^{-1} d_l}_{>0} < 0.$$

2.3.3.1 Methode des steilsten Abstiegs

Die Suchrichtung

$$s_l = -d_l = -\nabla f\left(x_l\right), \quad l = 1, 2, \ldots, \tag{2.18}$$

führt auf die sog. *Methode des steilsten Abstiegs* (MSA) (d. h. für (2.17) gilt nach (2.18) $B_l = \operatorname{diag}(1, \ldots, 1) \in \mathbf{R}^{n \times n}$) [171, S. 20]. Die MSA geht aus der *Taylor-Entwicklung* (Linearisierung) $f\left(x_l + \mu x_l\right) = x_l + \mu s_l^{\mathsf{T}} d_l + O(\mu^2)$ hervor [171, S. 21]. Die Implementierung ist unkompliziert, die Konvergenzrate allerdings in aller Regel gering (lineare Konvergenz).

Neben den in Abschnitt 2.3.2 vorgestellten Verfahren zur Schrittweitenbestimmung, ist, im Zusammenhang mit der Abstiegsrichtung (2.18), die *MSA mit regulärer Schrittweite* [112, S. 426, ff.] zu erwähnen. Zur Stabilisierung wird hierbei die Schrittweite $\mu_l > 0$ halbiert, sobald ein Richtungswechsel in s_l zu verzeichnen ist.

2.3.3.2 Newton-Verfahren

Eine Verbesserung in Bezug auf die Konvergenz liefert das sog. Newton-*Verfahren* (siehe bspw. [50, S. 86 ff.]). Es basiert auf der Idee, das Optimierungsproblem (2.11) durch eine sukzessive Minimierung einer quadratischen Näherung $q_s : \mathbf{R}^n \to \mathbf{R}$ zu lösen. Präziser: Der Wert für eine hinreichend glatte Funktion f kann an der Stelle $x_l + s_l$ durch die *Taylor-Entwicklung*

$$f\left(x_l + s_l\right) \approx f\left(x_l\right) + s_l^{\mathsf{T}} d_l + \frac{1}{2} s_l^{\mathsf{T}} H_l s_l =: q_s\left(x_l\right)$$

mit $H_l := \nabla^2 f\left(x_l\right)$, $H_l \in \mathbf{R}^{n \times n}$, angenähert werden [171, S. 22]. Aus der Annahme, dass in einer Umgebung eines Minimierers $H_l \succcurlyeq 0$ gilt, folgt, dass die aus der Minimierung von q_s gewonnene Suchrichtung adäquat ist. Nullsetzen der Ableitung $\partial_{s_l} q_s$ liefert die *Newton-Richtung*

$$s_l = -H_l^{-1} d_l \tag{2.19}$$

und damit die Verfahrensvorschrift $x_{l+1} = x_l - H_l^{-1} d_l$ [171, S. 12]. Aus (2.17) folgt unmittelbar die Forderung $H_l \succ 0$ um sicherzustellen, dass (2.19) eine Abstiegsrichtung ist. Ist die *Hesse-Matrix* fast singulär, kann eine Korrekturmatrix[13] $K_l \in \mathbf{R}^{n \times n}$ eingeführt werden, so dass $\left(H_l + K_l\right) \succ 0$ gilt. Hieraus ergibt sich die *modifizierte Newton-Richtung* $s_l = -\left(H_l + K_l\right)^{-1} d_l$.

Für das Newton-Verfahren stellt sich eine lokal quadratische Konvergenz ein [171, S. 44 ff.]. Um den Konvergenzbereich zu erweitern, kann (2.19) durch eine Schrittweitensteuerung $\mu_l \in (0, 1)$ ergänzt werden.

[13] Typischerweise wird $K_l := \epsilon \operatorname{diag}(1, \ldots, 1) \in \mathbf{R}^{n \times n}$, $\epsilon > 0$, gewählt.

Die Berechnung der Hesse-Matrix ist allerdings aufwändig [171, S. 23]. Eine effiziente Alternative ist die Verwendung von Quasi-Newton-Verfahren.

2.3.3.3 Quasi-Newton-Verfahren

Quasi-Newton-Verfahren (*QN-Verfahren*) [171, S. 135] sind eine attraktive Alternative zum Newton-Verfahren. Aus Effizienzgründen wird auf eine explizite Berechnung der (Inversen der) Hesse-Matrix $H_l \in \mathbf{R}^{n\times n}$ (bzw. auf das Lösen des Gleichungssystems $H_l s_l = -d_l$) verzichtet. Stattdessen wird H_l durch eine Approximation $B_l \in \mathbf{R}^{n\times n}$ ersetzt. Diese wird in jedem Iterationsschritt $l \in \mathbf{N}_0$ aktualisiert und ist so gewählt, dass B_l die *Sekantengleichung* (*QN-Bedingung*)

$$B_{l+1}(x_{l+1} - x_l) = d_{l+1} - d_l \tag{2.20}$$

erfüllt [171, S. 24]. Durch (2.20) ist B_l nicht eindeutig bestimmt. Weitere Bedingungen sind anzugeben. Diese führen auf unterschiedliche QN-Verfahren. Ein prominenter Vertreter ist das *Broyden-Fletcher-Goldfarb-Shanno-Verfahren* (*BFGS-Verfahren*) (siehe [171, S. 136 ff.]). Zusätzlich zu (2.20) wird $B_l \in \mathfrak{S}^{n,+}$ gefordert. Mit $y_l := d_{l+1} - d_l$, $y_l \in \mathbf{R}^n$, und $q_l := x_{l+1} - x_l = \mu_l s_l$, $q_l \in \mathbf{R}^n$, lautet die Aktualisierungsvorschrift des BFGS-Verfahrens für die Matrix B_l [171, S. 140]

$$B_{l+1} = B_l + \frac{y_l y_l^\mathsf{T}}{y_l^\mathsf{T} q_l} - \frac{B_l q_l q_l^\mathsf{T} B_l}{q_l^\mathsf{T} B_l q_l} \tag{2.21}$$

mit einer vorgegebenen Matrix $B_0 \in \mathfrak{S}^{n,+}$. Für diese kann u. a. $B_0 = \operatorname{diag}(1, \ldots, 1) \in \mathbf{R}^{n\times n}$ verwendet werden. Die erste Iteration ist nach (2.17) äquivalent zur MSA. Für QN-Verfahren stellt sich trotz des approximativen Charakters eine superlineare Konvergenz ein.

Aus (2.17) ist ersichtlich, dass anstelle von B_l die Inverse B_l^{-1} zu bestimmen ist. Die Lösung eines linearen Systems kann durch die Aktualisierungsvorschrift [171, S. 140]

$$B_{l+1}^{-1} = \left(E - \frac{q_l y_l^\mathsf{T}}{y_l^\mathsf{T} q_l}\right) B_l^{-1} \left(E - \frac{y_l q_l^\mathsf{T}}{y_l^\mathsf{T} q_l}\right) + \frac{q_l q_l^\mathsf{T}}{y_l^\mathsf{T} q_l} \tag{2.22}$$

mit $E = \operatorname{diag}(1, \ldots, 1) \in \mathbf{R}^{n\times n}$ vermieden werden.

Eine Verbesserung des BFGS-Algorithmus, im Sinne einer Speicherplatzoptimierung, ist das sog. *limited-memory BFGS-Verfahren* (*L-BFGS*) [170]. Im Gegensatz zum BFGS-Verfahren wird B_l nicht explizit gespeichert, sondern die Aktualisierungsvorschriften (2.21) bzw. (2.22) in Form weniger Vektoren implizit repräsentiert. Die Grundidee lässt sich direkt aus (2.21) bzw. (2.22) ablesen. Anstelle der Matrix B_l werden die Vektoren y_l und q_l gespeichert. Damit sind nach l Iterationen anstatt $l \times n \times n$ Werten, nur noch $2l \times n$ Werte im Speicher zu halten. Für große l und großmaßstäbliche Probleme stellt dies

allerdings nicht wirklich eine Verbesserung dar. Beim L-BFGS-Verfahren wird deshalb lediglich eine Untermenge der Vektoren y_k und q_k, $k = 1, ..., l$, zur Berechnung der Matrix B_l in (2.21) (bzw. der Inversen in (2.22)) verwendet. Präziser werden für ein vorgegebenes $L \in \mathbf{N}_{<l}$ nach l Iterationen die Vektoren $y_l, y_{l-1}, ..., y_{l-L}$ und $q_l, q_{l-1}, ..., q_{l-L}$, mit $l > L$ genutzt. Hierbei gilt $B_{l-L} := B_0$. Damit reduziert sich der Speicherplatzbedarf von $2l \times n$ auf $2L \times n$ Werte. Für großmaßstäbliche Probleme stellt sich bereits für kleine L ($L \in \{3, 5, 7\}$) eine hinreichende Performanz ein [99 & 145].

2.3.4 Abbruchkriterien

Als Abbruchkriterien werden Standardbedingungen aus der Literatur verwendet [73, S. 305 ff.]. Diese sind durch

$$\begin{aligned}
&(\text{K1}) \quad \|f(x_{l-1}) - f(x_l)\| < \tau (1 + \|f(x_l)\|) \\
&(\text{K2}) \quad \|x_{l-1} - x_l\| < \sqrt{\tau} (1 + \|x_l\|) \\
&(\text{K3}) \quad \|d_l\| < \sqrt[3]{\tau} (1 + \|f(x_l)\|) \\
&(\text{K4}) \quad \|d_l\| < \varepsilon \\
&(\text{K5}) \quad l > l_{\max}
\end{aligned}$$

erklärt. Hierbei bezeichnet $\tau > 0$ eine vorgegebene Konstante, $l_{\max} \in \mathbf{N}$ eine maximal zugelassene Iterationszahl und $\varepsilon > 0$ die Maschinengenauigkeit. Die Optimierung wird angehalten, falls $((\text{K1}) \wedge (\text{K2}) \wedge (\text{K3})) \vee (\text{K4}) \vee (\text{K5})$ erfüllt ist. Die Kriterien (K1), (K2) und (K3) deuten auf eine Konvergenz des Verfahrens hin. Die Abbruchbedingungen (K4) und (K5) können als eine Form von „Notfallkriterien“ aufgefasst werden. Es wird garantiert, dass das Verfahren, nach einer bestimmten Anzahl an Schritten, oder wenn die Berechnungen auf Zahlen in der Größenordnung der Maschinengenauigkeit führen, terminiert.

2.4 Fazit

In den vorangegangenen Abschnitten wurden Verfahren zur Lösung dünnbesetzter, linearer Gleichungssysteme (siehe Abschnitt 2.2) und zur numerischen Optimierung unrestringierter Optimierungsaufgaben (siehe Abschnitt 2.3) vorgestellt. Einer prägnanten Darstellung wegen wurde nur das Wesentliche besprochen. Weder die Lösung linearer Gleichungssysteme noch die numerische Optimierung werden als Forschungsgegenstand dieser Arbeit verstanden. Der geneigte Leser sei auf ergänzende Literatur zu Verfahren der Lösung linearer Gleichungssysteme [157, 196 & 213] oder zur numerischen Optimierung [5, 50, 73 & 171] verwiesen. In der konkreten Implementierung sind die Verfahren unterschiedlichen Bibliotheken entliehen. Details sind in Abschnitt zusammengetragen.

http://www.springer.com/978-3-658-05245-4

Methoden zur numerischen Simulation der Progression von Gliomen
Modellentwicklung, Numerik und Parameteridentifikation

Mang, A.

2014, XV, 206 S. 75 Abb., 10 Abb. in Farbe., Softcover
ISBN: 978-3-658-05245-4