Ein Algorithmus zur Pfadverfolgung großer nichtlinearer Systeme unter Ausnutzung von Sparse-Strukturen

Diplomarbeit, FU Berlin

C. Klein-Robbenhaar
Betreuer: Prof. P. Deuflhard

im April 1994

Inhalt

Einleitung 3

1 Der Pfadverfolgungs-Algorithmus 6
1.1 Problemstellung 6
1.1.1 Parameterabhängige Gleichungssysteme 6
1.1.2 Implizite Reparametrisierung über die Tangente 8
1.2 Das Lösungsverfahren 9
1.2.1 Pfadverfolgung mit dem Prediktor/Korrektor Schema . 9
1.2.2 Realisierungen des Prediktors 10
1.2.3 Das Gauß-Newton-Verfahren als Korrektor 12
1.2.4 Modifikationen des Gauß-Newton-Verfahrens 15
1.2.5 Konvergenztheorie und Schrittweitensteuerung 19

2 Singularitäten 30
2.1 Typen und Normalformen von Singularitäten 30
2.2 Ljapunov-Schmidt-Reduktion und Konstruktion der Normalform 32
2.3 Wendepunkte 35
2.4 Einfache Verzweigungen 38

3 Berechnung der Pseudoinversen 49
3.1 Definition der Moore-Penrose-Pseudoinversen 49
3.2 Herleitung des Lösungsverfahrens 51
3.3 Realisierungen des Lösungsschemas 58
3.3.1 Singulärwertzerlegung 58
3.3.2 QR-Zerlegung 59
3.3.3 LU-Zerlegung 61
3.3.4 Deflation 65
3.3.5 Block-Metalöser 69
3.4 Lösung des Mooreschen Systems 74
3.4.1 Transformation auf Block-Dreiecksgestalt 74
3.4.2 Varianten der Nachiteration 77

4 Implementierung 79
4.1 Realisierung der Skalierung 79
4.2 Weitere Details des Algorithmus 83
4.3 Kurzbeschreibung des verwendeten Sparse-Lösers 85
4.4 Objekt-orientierte Implementierung in C 91

5 Beispiele 108
5.1 Ein kleines, aber schwieriges Beispiel: Der Brusselator 108
5.2 Invariante Mannigfaltigkeiten gekoppelter Oszillatoren 118
5.3 Stabwerkkonstruktionen 126

A Die tangentiale Determinante einer $(n, n+1)$-Matrix 139

Symbolverzeichnis 143

Literaturverzeichnis 145

Einleitung

Diese Arbeit behandelt die numerische Lösung von reellen, endlichdimensionalen, nichtlinearen Gleichungssystemen, die von einem reellen Parameter abhängen. Speziell werden Techniken der linearen Algebra (sogenannte Sparse-Löser) angewandt für den Fall, daß die Linearisierung des Gleichungssystems nur in wenigen Elementen ungleich Null ist; dadurch kann der benötigte Rechenaufwand reduziert werden, auch wenn die Anzahl der Variablen groß ist (ca. 100-1000).

Die Berechnung der Lösung, die unter geeigneten Regularitätsannahmen aus einer eindimensionalen Mannigfaltigkeit besteht und dann als Lösungspfad bezeichnet wird, erfolgt nach dem Prediktor-Korrektor-Verfahren. Dabei werden von einer Anfangslösung aus nach und nach neue Punkte auf dem Lösungspfad berechnet, indem von einer aus Informationen der vorherigen Lösungen gewonnenen Näherungslösung, dem Prediktor, durch ein diese Näherung verbesserndes Verfahren, dem Korrektor, eine neue Lösung berechnet wird. Da das Verfahren nach und nach Lösungen auf dem Pfad berechnet und somit seinem Verlauf folgt, wird es auch als Pfadverfolgung bezeichnet.

Gewöhnlich wird bei der Pfadverfolgung von einer herausragenden Stellung der Parameters im Gleichungssystem ausgegangen und angenommen, daß sich der Lösungspfad als Funktion des Parameters darstellen läßt. Diese Betrachtung kann jedoch zu vermeidbaren Singularitäten in der Lösung führen. Daher wird hier als Korrektor das Gauß-Newton-Verfahren verwendet, welches den Parameter wie eine gewöhnliche Variable behandelt. Damit kann der Lösungspfad auch jenseits von Singularitäten, die sich aus obiger Betrachtung ergeben, ohne Probleme verfolgt werden, solange der Pfad glatt und lokal eindeutig ist. Ist die Eindeutigkeit lokal nicht gegeben, liegt eine Singularität vor, die nicht der Auszeichnung des Parameters geschuldet ist, und die im Rahmen der Pfadverfolgung eine besondere Behandlung erfordert.

In Kapitel 1 wird der Algorithmus der Pfadverfolgung in der Form, wie er in [9] formuliert wurde, dargestellt. Dabei wird neben der geometrischen Interpretation des Algorithmus ein breiter Raum für die Konvergenztheorie des Gauß-Newton-Verfahrens eingeräumt, da diese für die Wahl der Schrittweite, d.h. der Entfernung des Prediktors von der vorherigen Lösung, und damit für die Effizienz des Verfahrens wichtig ist.

In Kapitel 2 werden dann Singularitäten entlang eines Lösungspfades und ihre Klassifizierung der allgemeinen Theorie in [20] folgend skizziert und abhängig davon, ob der Parameter ausgezeichnet wird oder nicht, die Behandlung des jeweils einfachsten Typs von Singularität (wieder nach [9]) dargestellt.

Das Gauß-Newton-Verfahren wandelt das nichtlineare Problem in eine Folge von linearen Gleichungssystemen um, wobei die lokale Nichteindeutigkeit der Lösung sich in der Unterbestimmtheit der linearen Gleichungssysteme widerspiegelt. Diese Systeme werden mittels der Moore-Penrose-Pseudoinversen gelöst. In Kapitel 3 wird zunächst das Konzept dieser Pseudoinversen erläutert und dann, aufbauend auf einem generellen Schema zur Berechnung dieser Pseudoinversen, verschiedene Verfahren zur Lösung der Gleichungssysteme entwickelt, die den Anforderungenn der Pfadverfolgung und der Struktur der großen Gleichungssysteme angemessen sind. Schließlich wird das Schema noch auf die Lösung eines erweiterten Systems angewandt, wie es zur Berechnung von Singularitäten benötigt wird.

Kapitel 4 widmet sich der Implementierung des Algorithmus, der in der Sprache C realisiert wurde. Dabei wird neben für die Effizienz des Algorithmus wichtigen Details, insbesondere der Wahl einer geeigneten Skalierung, die Funktionsweise des Sparse-Lösers `MA28` anläßlich seiner Übertragung von Fortran in C erläutert und die in Organisation des im objekt-orientierten Stil gehaltenen Programmes `Alcon-S` erläutert.

Kapitel 5 schließt die Arbeit mit einigen illustrativen Beispielen, die die Funktionstüchtigkeit des Algorithmus unter Beweis stellen.

Zum Abschluß dieser Einleitung bleibt mir die Freude, mich beim Konrad-Zuse-Zentrum für Informationstechnik zu bedanken. Ohne Unterstützung von dieser Seite nicht zuletzt durch eine Tutorenstelle wäre diese Arbeit zumindest in dieser ausführlichen Form nicht möglich gewesen. Insbesondere möchte ich mich bei Professor Peter Deuflhard bedanken, der nicht nur mir das Diplomthema vorgeschlagen hat, sondern auch durch seine Vorlesungen mein Interesse an der Numerik überhaupt geweckt hat, in dem er gezeigt hat, daß eine algorithmisch orientierte Numerik eine spannende und vielfältige Wissenschaft ist fern dem Klischee vom Abschätzen von Rundungsfehlern. Mein ausdrücklicher Dank geht an Doktor Andreas Hohmann, der nicht nur die Seite der Programmentwicklung kontinuierlich betreut und durch eigene Entwicklungen vorangetrieben hat, sondern auch stets als Ansprechpartner

zur Verfügung stand und sich mit bewundernswerter Geduld Zeit für alle meine Fragen, auch die deutlich weniger schlauen, genommen hat.

Herrn Ulrich Nowak möchte ich für hilfreiche Diskussionen zum Thema dünnbesetzte Matrizen und Herrn Wolfgang Dalitz für die Hilfestellung beim Einfügen von `Alcon-S` in die `eLib` Dank sagen.

„Habe ich jemanden vergessen? Ich denke, schon“ (Max Goldt)

1. Der Pfadverfolgungs-Algorithmus

1.1 Problemstellung

1.1.1 Parameterabhängige Gleichungssysteme

Häufig tritt in Anwendungen der Fall auf, daß in der mathematischen Modellierung ein Gleichungssystem nicht nur von den Unbekannten abhängt, nach denen es aufgelöst werden soll, sondern auch noch von einem oder mehreren Parametern; sei es, daß sie entweder zusätzliche Freiheitsgrade im Modell darstellen oder unter der interessierenden Betrachtung über einem Bereich variieren können oder auch künstlich eingefügt werden, um eine Lösung überhaupt erst zu erreichen – meist interessiert nicht nur die Lösung zu ausgesuchten Parameterwerten, sondern auch die Abhängigkeit der Lösung bei Veränderung der Parameter.

In dieser Arbeit werden reelle, endlichdimensionale Gleichungssysteme betrachtet, die von *einem* reellen Parameter abhängen; das Problem läßt sich also formulieren als:

> Finde alle Lösungen (x, λ), wobei $x \in \mathbb{R}^n$ der Lösungsvektor und $\lambda \in \mathbb{R}$ der Parameter ist, die die Gleichung
>
> $$\begin{array}{rcl} f(x,\lambda) & = & 0 \\ f\colon \mathbb{R}^n \times \mathbb{R} & \mapsto & \mathbb{R}^n \end{array} \tag{1.1}$$
>
> erfüllen.

Interessant ist dabei vor allem die Frage, wie x von λ abhängt; d.h. ob eine glatte Funktion $x = x(\lambda)$ existiert, so daß

$$f(x(\lambda), \lambda) = 0$$

für alle $\lambda \in \mathbb{R}$ oder zumindest für $\lambda \in D$, wobei $D \subset \mathbb{R}$ ein Bereich von für das Modell relevanten Parameterwerten ist.

Beispiel 1.1: In Anwendungen der Physik, Chemie, Mechanik und Elektrotechnik führt die mathematische Modellierung zeitabhängiger Prozesse häufig zu einem System gewöhnlicher Differentialgleichungen:

$$\dot{x} = f(x, \lambda) \tag{1.2}$$

Häufig ist dabei nicht die exakte Lösung zu einem bestimmten Anfangswert gesucht, sondern strukturelle Eigenschaften des Systems, z.B. die Existenz von Gleichgewichtspunkten, Grenzzyklen oder komplizierteren Objekten wie seltsamen Attraktoren, denen sich Lösungskurven asymptotisch annähern. Insbesondere die Berechnung von Gleichgewichtspunkten ist einfach zu bewerkstelligen, da sie durch

$$x(t) \equiv x_0 = \text{konst.} \iff \dot{x} = 0$$

definiert sind, was Gleichung (1.2) sofort in die Form (1.1) der betrachteten Problemklasse bringt.

Beispiel 1.2: Beim Lösen von stark nichtlinearen Gleichungssystemen $f(x) = 0$ etwa mit einem Newton-Verfahren taucht oft das Problem auf, daß kein brauchbarer Startvektor zur Verfügung steht. Eine mögliche Abhilfe besteht darin, das Problem mit einem künstlichen Parameter zu versehen, der z.B. von 0 bis 1 variiert, so daß für das parameterabhängige Problem $h(x, \lambda)$ gilt, daß $h(x, 0)$ leicht lösbar ist (z.B. linear) während $h(x, 1) \equiv f(x)$ ist. Läßt sich nun die Lösung von $h(x(\lambda), \lambda) = 0$ von $x(0)$ bis $x(1)$ verfolgen, so ist das ursprüngliche Problem gelöst. Eine solche Lösungsmethode heißt *Einbettung* oder *Homotopie*.

Die Frage, ob sich die Lösungsmenge von (1.1) zumindest lokal als $x = x(\lambda)$ über λ parametrisieren läßt, wird durch das implizite Funktionen Theorem beantwortet, welches (auch im allgemeinen Fall von p Parametern) besagt:

Satz 1.1 *Sei $f\colon \mathbb{R}^n \times \mathbb{R}^p \mapsto \mathbb{R}^n$, $f = f(x, \lambda)$ eine stetig differenzierbare Funktion und (x_0, λ_0) ein Punkt mit $f(x_0, \lambda_0) = 0$. Falls dann $\frac{\partial f}{\partial x}\Big|_{(x_0,\lambda_0)}$ den vollen Rang n hat, dann existiert lokal eine stetig differenzierbare Funktion $x = x(\lambda)$ mit $x(\lambda_0) = x_0$ sowie*

$$f(x(\lambda), \lambda) = 0 \qquad \textit{und} \qquad \frac{\partial f}{\partial x}\frac{dx}{d\lambda} = -\frac{\partial f}{\partial \lambda}$$

Ein Beweis dieser Behauptung findet sich in fast jeden Lehrbuch über Analysis, z.B. in [23].

Unter den gegebenen Voraussetzungen setzt sich die Lösungsmenge aus glatten $p-$dimensionalen Mannigfaltigkeiten zusammen, die im Kontext parameterabhängiger Gleichungssysteme mit $p = 1$ nicht wie üblich als Kurven, sondern als *(Lösungs-)Pfade* bezeichnet werden.

1.1.2 Implizite Reparametrisierung über die Tangente

Probleme beim Lösen parameterabhängiger Gleichungen treten dann auf, wenn $\frac{\partial f}{\partial x}$ nicht regulär ist. Satz 1.1 garantiert dann nicht mehr die Parametrisierbarkeit der Lösung über λ; diese ist auch meist in solchen Punkten nicht möglich, obwohl die Lösung aus einer glatten eindimensionalen Mannigfaltigkeit besteht.

Beispiel 1.3: Die Gleichung

$$f(x,\lambda) = x^2 + \lambda^2 - r^2 = 0$$

hat als Lösungsmenge einen Kreis zum Radius r, welcher plausibler weise eine eindimensionale Mannigfaltigkeit bildet; jedoch gilt in den Punkten $(x,\lambda) = (0,\pm r)$, daß $\frac{\partial f}{\partial x} = 0$ und die Lösung nicht mehr eindeutig über λ parametrisierbar ist.

Die Schwierigkeit an den Punkten $(0,\pm r)$ läßt sich in obigen Beispiel dadurch beseitigen, daß zunächst die Variable x und der Parameter λ zu einer Variable $y = (x,\lambda)$ zusammengefaßt werden und dann die Gleichung

$$f(y) = 0$$

über einem anderen Parameter dargestellt wird. In diesem konkreten Beispiel bietet sich x an, was zu der lokalen Parametrisierung

$$f(x,\lambda(x)) = 0 \quad \text{mit} \quad \lambda(x) = \pm\sqrt{r^2 - x^2} \quad \text{und} \quad \frac{\partial f}{\partial \lambda} = 2\lambda \neq 0$$

für $-r < x < r$ führt. Ein solches Verfahren wird *(lokale) Reparametrisierung* genannt und ist immer möglich, falls

$$f'(y) = \left(\frac{\partial f}{\partial x}\,\frac{\partial f}{\partial \lambda}\right)$$

vollen Rang hat.

Das Verfahren der Reparametrisierung läßt sich auf das allgemeine Problem (1.1) durch die folgende Verallgemeinerung anwenden.

Zunächst wird ein Koordinatenwechsel vorgenommen:

$$y = P\eta, \quad \eta = (\xi,\tau)$$

wobei P ein linearer, orthogonaler Operator ist und so gewählt wird, daß

$$\operatorname{rank}\left(\frac{\partial}{\partial\xi} f\circ P\right) = n \tag{1.3}$$

gilt. Damit kann die Lösung gemäß dem impliziten Funktionen Theorem 1.1 über τ parametrisiert werden.

Mit den bei dem Lösungsverfahren auftretenden linearen Gleichungssystemen im Augenwinkel wurde P orthogonal gewählt, damit die Konditionszahl in der 2-Norm

$$\operatorname{cond}_2\left(f'(y)\right) := \sigma_1/\sigma_n,$$

wobei σ_1 den größten und σ_n den kleinsten Singulärwert bezeichnen, unter der Transformation invariant bleibt, d.h. $\operatorname{cond}_2\left(f'(P\eta)\right) = \operatorname{cond}_2\left((f\circ P)'\eta\right)$. Falls $f'(y)$ vollen Rang hat, ist $\sigma_n \neq 0$ und $\operatorname{cond}_2\left(f'(y)\right) < \infty$.

Es ist jetzt eine natürliche Forderung an die Wahl von P, daß die Konditionszahl von $\frac{\partial}{\partial\xi} f\circ P$ nicht nur endlich sein soll, um (1.3) zu erfüllen, sondern so klein wie möglich. Diese Forderung ist erfüllt, wenn τ die Tangente des Lösungspfades parametrisiert, d.h. wenn $t = P\binom{0}{\tau}$ ein Tangentialvektor von $f'(y)$ ist. Denn dann ist

$$\frac{d}{d\eta} f\circ P = \begin{pmatrix} \frac{\partial}{\partial\xi} f\circ P & \frac{\partial}{\partial\tau} f\circ P \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial\xi} f\circ P & 0 \end{pmatrix}$$

da $\frac{\partial}{\partial\tau} f\circ P = f'(y)\,t = 0$; also besitzt $\frac{\partial}{\partial\xi} f\circ P$ dieselben Singulärwerte und hat daher dieselbe Konditionszahl wie $\frac{d}{d\eta} f\circ P$.

Die Idee der impliziten Reparametrisierung des Pfades über die Tangente motiviert den nun folgenden Lösungsalgorithmus.

1.2 Das Lösungsverfahren

1.2.1 Pfadverfolgung mit dem Prediktor/Korrektor Schema

Die Einsicht, daß sich die Lösung von parameterabhängigen Gleichungen lokal gut über eine lineare orthogonale Transformation reparametrisieren läßt, während eine globale Reparametrisierung im allgemeinen nur über krummlinige Koordinaten zu erreichen ist (im Beispiel 1.3 wären dies z.B. Polarkoordinaten), legt die Idee nahe, die Lösung mittels gegebener lokaler Information

schrittweise fortzusetzen; dabei werden ausgehend von einer Startlösung y_0 nach und nach Punkte $y_1, y_2 \ldots, y_\nu, \ldots$ nach folgendem Schema berechnet:

- Von einer gegebenen Lösung y_ν mit $f(y_\nu) = 0$ wird aus den Informationen der bisherigen Lösungen eine Näherung für eine weitere Lösung auf dem Pfad $\hat{y}_{\nu+1}$ vorgeschlagen, der sogenannte *Prediktor*.

- Ausgehend von diesem Prediktor wird ein iteratives Verfahren bei $y^0 := \hat{y}_{\nu+1}$ gestartet, welches gegen eine Lösung $\lim\limits_{k\to\infty} y^k = y^* =: y_{\nu+1}$ konvergiert und als *Korrektor* bezeichnet wird. Danach kann dann der Prediktiorschritt mit der neuen Lösung $y_{\nu+1}$ anstelle von y_ν durchlaufen werden. Wenn der Korrektor versagt, muß im Prediktorschritt ein verbesserter Prediktor $\hat{y}_{\nu+1,neu}$ vorgeschlagen oder, falls dies nicht möglich ist, das Verfahren abgebrochen werden werden.

Diese Methode wird als *Pfadverfolgung, Fortsetzung* oder aufgrund der Bezeichnung der beiden Teilschritte auch als *Prediktor-Korrektor-Verfahren* bezeichnet.

1.2.2 Realisierungen des Prediktors

Der *klassische Prediktor* geht von der Existenz eine Parametrisierung über λ aus.

Ausgehend von einer Lösung $y_\nu = (x_\nu, \lambda_\nu)$ wird der Parameter λ um eine Schrittweite s_ν erhöht, deren genaue Größe weiter unter diskutiert wird. Als Prediktor für $\hat{x}_{\nu+1}$ wird die alte Lösung x_ν verwendet. Also lauten die Formeln für den klassischen Prediktor:

$$\begin{aligned} \hat{x}_{\nu+1} &:= x_\nu \\ \hat{\lambda}_{\nu+1} &:= \lambda_\nu + s_\nu \end{aligned}$$

Ein solcher Prediktor führt aber in der Regel zum Scheitern des Korrektors, wenn die Rangbedingung in Satz 1.1 verletzt ist, weil für Parameterwerte jenseits der Singularität in der Ableitung keine Lösung mehr zu existieren braucht, siehe das Beispiel 1.3.

Anstatt die Lösung als $x = x(\lambda)$ darzustellen, bietet es sich nach der Diskussion von Kapitel 1.1.2 an, die Lösung über die Tangente $t_\nu = t(y_\nu)$ am

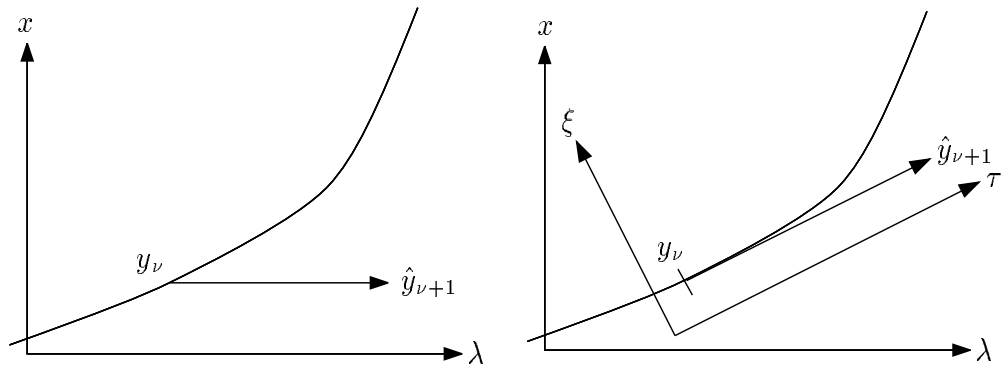


Abbildung 1.1: Klassischer und tangentialer Prediktor

Lösungspunkt y_ν zu parametrisieren, woraus die Übertragung des klassischen Prediktors auf diese Parametrisierung zum *tangentialen Prediktor* führt:

$$\hat{y}_{\nu+1} \quad := \quad y_\nu + s_\nu t_\nu \tag{1.4}$$

Solange $f'(y)$ vollen Rang hat, ist dann bezüglich dieses Prediktors das Auftauchen von Singularitäten zumindest lokal ausgeschlossen.

Andere Prediktoren ergeben sich daraus, daß sich der Lösungspfad der Gleichung $f(y) = 0$ global über seine Bogenlänge σ parametrisieren läßt, d.h. $y = y(\sigma)$. Durch Differenzieren ergibt sich dann die implizite Differentialgleichung

$$f'(y)\frac{dy}{d\sigma} = 0$$

welche mit den vorhergehenden Lösungen als Stützpunke mittels Mehrschritt-Formeln approximativ gelöst werden kann, siehe etwa [27]. Der Näherungswert wird dann als Prediktor genommen.

Im Kontext dieses Zuganges läßt sich der tangentiale Prediktor (1.4) als ein expliziter Euler-Schritt auffassen und wird daher auch als *Euler-Prediktor* bezeichnet.

Formeln höherer Ordnung ergeben bessere Abschätzungen in s von der Form

$$\mathrm{dist}(\hat{y}_{\nu+1}, y^*) < C_p s^p$$

wobei p die Ordnung des Verfahrens ist; jedoch kann ein Wachsen der Konstante C_p mit p die Verringerung von s^p wieder wettmachen, zumal bei höhe-

rer Ordnung Information aus zunehmend weiter entfernten Punkten herangezogen wird, die mit den lokalen Eigenschaften der Funktion „vor Ort“ wenig zu tun haben müssen.

Daher wird im weiteren nur der tangentiale Prediktor verwendet, zumal für ihn eine theoretisch fundierte Schrittweitensteuerung vorliegt, siehe Kapitel 1.2.5 weiter unten.

1.2.3 Das Gauß-Newton-Verfahren als Korrektor

Als Korrektor wird eine Variante des Newton-Verfahrens gewählt, welches für seine schnelle Konvergenz gegen Lösungspunkte $y^* = \lim y^k$ bei geeigneten Startwerten y^0 bekannt ist, die hier durch den Prediktor bereitgestellt werden.

Allgemein läßt sich das Newton-Verfahren beschreiben als

$$y^{k+1} = y^k + \Delta y^k \tag{1.5}$$

wobei die Newton-Korrektur Δy^k das Gleichungssystems

$$f'(y^k)\Delta y^k = -f(y^k) \tag{1.6}$$

löst.

Beim normalen Newton-Verfahren wird angenommen, daß die Lösung y^* lokal eindeutig ist; diese Annahme führt lokal auch zu eindeutig lösbaren linearen Problemen (1.6). Für die hier diskutierte Problemklasse findet sich die Unterbestimmtheit des zu Grunde liegenden Problems wieder in einer Unterbestimmtheit des linearen Gleichungssystems.

Der in der Literatur (z.B. [2, 3, 26, 39]) verbreiteteste Weg, diese Unterbestimmtheit aufzuheben, besteht darin, noch eine zusätzliche Bedingung in Form einer Funktion

$$\begin{gathered} u_s\colon \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R} \\ u_s(x, \lambda) = 0 \end{gathered}$$

zu stellen, die von der Schrittweite s abhängig ist, und dann für die erweiterte Funktion

$$G(y) = \begin{pmatrix} f(y) \\ u(y) \end{pmatrix} = 0 \tag{1.7}$$

ein gewöhnliches Newton-Verfahren zur Lösung zu benutzen. Da die Bedingung sich zumeist als Approximationen der Bedingung:

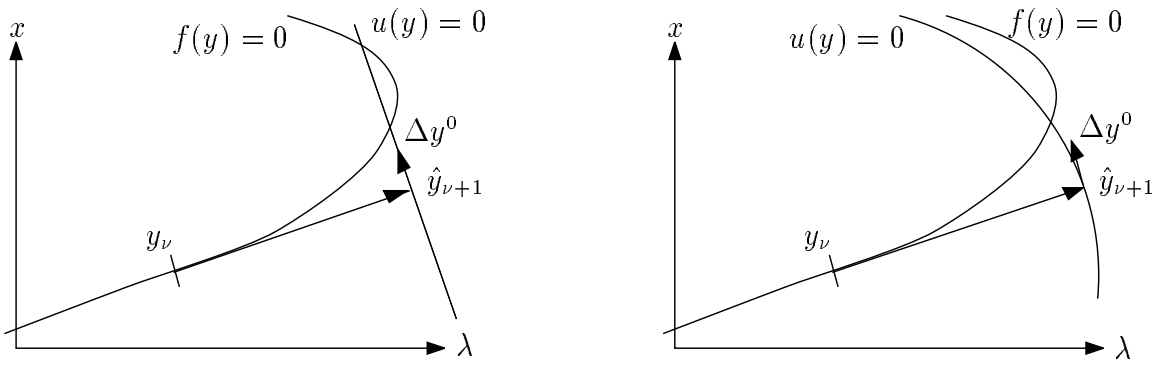


Abbildung 1.2: Lineare und Nichtlineare Zusatzbedingungen

> Finde eine Lösung auf dem Pfad, so daß die Bogenlänge σ zur vorherigen Lösung genau s ist.

auffassen läßt, wird dieser Zugang auch *Pseudo-Bogenlänge*-Verfahren genannt.

Eine typische lineare Zusatzbedingung ist etwa

$$u(y) := \langle t_\nu, y - \hat{y}_{\nu+1} \rangle = 0 \tag{1.8}$$

d.h. die eindeutige Lösung ist diejenige, die auf der Hyperebene orthogonal zur Tangente t_ν durch $\hat{y}_{\nu+1}$ liegt. Eine häufig benutzte nichtlineare Bedingung ist:

$$u(y) := \|y - y_\nu\|^2 - s_\nu^2 = 0 \tag{1.9}$$

d.h. die Lösung liegt auf der Hypersphäre mit dem Radius s_ν und Mittelpunkt y_ν.

Eine kleine Sammlung an möglichen Zusatzbedingungen läßt sich in [2], Kapitel 3.2 „Local Parametrizations“ finden.

Lineare Bedingungen wie (1.8) haben aber den Nachteil, daß sie an die Nichtlinearität von f nicht angepaßt sind; die Berechnung des Schnittpunktes kann schlecht konditioniert sein oder es kann für zu große Schrittweiten keine Lösung mehr existieren.

Nichtlineare Bedingungen wie (1.9) erlauben etwas größere Schrittweiten, aber dafür verletzt das Newton-Verfahren schon mit der ersten Korrektur die Zusatzbedingung, die noch vom Startwert erfüllt war, und verliert möglicherweise unnötige Iterationen, um den Schnittpunkt zu finden, obwohl eine beliebige Lösung $f(y^*) = 0$ genügen würde.

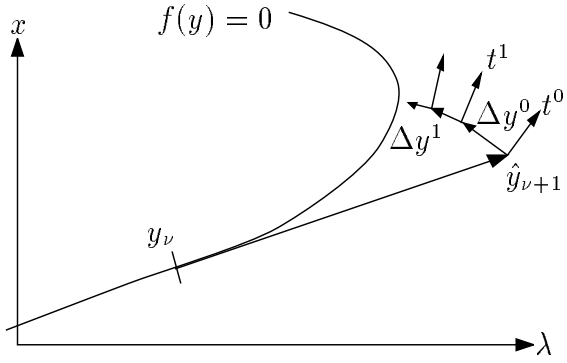


Abbildung 1.3: Gauß-Newton-Verfahren als Korrektor

Im Folgenden wird ein anderer, in [9] vorgeschlagener Weg beschritten. Anstelle durch äußere Zusatzbedingungen, die auf das lokale Problem keine Rücksicht nehmen, eine Eindeutigkeit des nichtlinearen Problems (1.1) zu erzwingen, wird eine innere Zusatzbedingung an das lineare Problem innerhalb der Newton-Iteration (1.6) gestellt, welche eng mit dem lokalen Problem zusammenhaängt. Verwendet wird die Bedingung, daß die Lösung Δy^k senkrecht auf der Tangente von $f'(y^k)$ steht. Wie in Abschnitt 3.1 gezeigt wird, ist dies gleichbedeutend mit der Bedingung

$$\|\Delta y^k\|_2 = \min \tag{1.10}$$

Die so eindeutig charakterisierte Lösung läßt sich formal als

$$\Delta y^k = -f'(y^k)^+ f(y^k)$$

schreiben, wobei die Notation A^+ die Moore-Penrose-Pseudoinverse eines linearen Operators A bezeichnet und in Kapitel 3 genauer beschrieben wird. Die Verwendung der Newton-Iteration mit der Pseudo-Inversen als Löser wird als *Gauß-Newton-Verfahren* bezeichnet.

Die Bedingung (1.10) läßt sich auf zwei verschiedene Weisen motivieren. Zum einen ist der Fehler durch die Linearisierung des Problems

$$0 = f(y^*) = f(y^k) + f'(y^k)\Delta y^k + \mathcal{O}(\|\Delta y^k\|^2)$$

von der Ordnung $\|\Delta y^k\|^2$; dadurch, daß diese Größe minimiert wird, läßt sich eine möglichst schnelle Konvergenz des Gauß-Newton Verfahrens erhoffen. Diese Hoffnung wird mit Satz 1.2 in Kapitel 1.2.5 theoretisch untermauert.

Zum anderen läßt sich mit der Terminologie von Kapitel 1.1.2 die Lösung von (1.6) und (1.10) als Lösung auf der Hyperebene $P\binom{\xi^k}{0}$ auffassen, auf der die lineare Abbildung $f'(y^k)$ optimal konditioniert ist, was dem Verfahren sicherlich zugute kommt.

Wird die Minimierungsbedingung (1.10) als von y^k abhängige Zusatzfunktion hingeschrieben,

$$u_{y^k}(y) = \langle t(y^k), \Delta y^k \rangle$$

wobei $t(y^k)$ ein Tangentialvektor von $f'(y^k)$ ist, ergibt sich für das zu lösende Gleichungssystem

$$\begin{pmatrix} f'(y^k) \\ t(y^k)^* \end{pmatrix} \Delta y^k = \begin{pmatrix} -f(y^k) \\ 0 \end{pmatrix} \tag{1.11}$$

daß die hinzugefügte Zeile orthogonal auf den Zeilen von $f'(y^k)$ steht. Von allen möglichen Zusatzbedingungen ist dies diejenige, welche im Sinne der Konditionszahl der 2-Norm die günstigste ist.

Bemerkung 1.1: Es sei an dieser Stelle ausdrücklich darauf hingewiesen, daß das Verfahren neben lokaler Information in der Nähe des Lösungspunktes auch von dem verwendeten inneren Produkt abhängt. Im Algorithmus wird daher ein problemangepaßtes skaliertes Produkt verwendet.

1.2.4 Modifikationen des Gauß-Newton-Verfahrens

Die Auswertung und Zerlegung der Jacobi-Matrix insbesondere für Probleme hoher Dimension in Verbindung mit direkten Lösern stellt oft ein im Verhältnis zu einer einfachen Funktionsauswertung sehr viel höheren Aufwand dar. Zur Konvergenz des Gauß-Newton-Verfahrens ist es aber auch nicht erforderlich ist, an jedem Iterationspunkt die Ableitung exakt zu berechnen. Daher sind Verfahren entwickelt worden, bei denen die $f'(y^k)$ durch geeignete Approximationen M_k ersetzt werden. Die Iteration ändert sich dann formal folgendermaßen:

$$\begin{array}{lrcl} \text{löse} & M_k s^k & = & -f(y^k) \\ & y^{k+1} & = & y^k + s^k \\ \text{ersetze} & M_k & \Rightarrow & M_{k+1} \end{array}$$

Durch die Approximationen wird im allgemeinen der Konvergenzradius und die Konvergenzgeschwindigkeit beeinträchtigt; z.B. gilt für den Approximationsfehler nur noch

$$f(y^*) - f(y^k) - M_k s^k = (f'(y_k) - M_k)s^k + \mathcal{O}(\|y^* - y^k\|^2) \approx \mathcal{O}(\|y^* - y^k\|)$$

Die einfachste solcher Modifikationen ergibt sich, wenn die Ableitung des Startpunktes während der Iteration beibehalten wird, d.h.

$$M_k = f'(y^0).$$

Das so erhaltene Verfahren heißt dementsprechend *vereinfachtes Gauß-Newton-Verfahren* und ist nur noch linear konvergent ([8]). Die vereinfachten Gauß-Newton-Korrekturen werden im weiteren mit

$$s_k = \overline{\Delta y}^k$$

bezeichnet.

Ein weiteres Verfahren wurde von Broyden in [5] vorgeschlagen und besteht in der Idee, die Ableitung durch eine Sekante zu approximieren; dazu wird das Verfahren in y^0 mit $B_0 = M_0 = f'(y^0)$ gestartet; nach der Berechnung der Korrektur, die hier mit $s^k = \widehat{\Delta y}^k$ bezeichnet wird, wird in jeder Iteration verlangt, daß die neue Approximation B_{k+1} die Sekantenbedingung

$$B_{k+1}\widehat{\Delta y}^k = f(y^{k+1}) - f(y^k)$$

erfüllen soll; da B_k schon die Gleichung $B_k\,\widehat{\Delta y}^k = -f(y^k)$ erfüllt, muß also für die Korrektur der Ableitung die Gleichung

$$(B_{k+1} - B_k)\,\widehat{\Delta y}^k = f(y^{k+1}) \tag{1.12}$$

gelten. Diese Bedingung bestimmt die Korrektur nur auf dem eindimensionalen Raum $\mathcal{R}(\widehat{\Delta y}^k)$; Broyden schlug vor, die Korrektur dadurch eindeutig zu bestimmen, daß sie sich auf $\mathcal{R}(\widehat{\Delta y}^k)^\perp$ sich nicht ändert; dadurch ergibt sich

$$B_{k+1} - B_k = \frac{f(y^{k+1})\widehat{\Delta y}^{k^*}}{\widehat{\Delta y}^{k^*}\widehat{\Delta y}^k} \tag{1.13}$$

Dieses Korrekturverfahren läßt sich auch als Lösung der Miminierungsproblems

$$\|B_{k+1} - B_k\|_F = \min\{\, \|X - B_k\|_F \mid X - B_k \text{ erfüllt Gleichung (1.12)} \,\}$$

auffassen, siehe [8].

Das Verfahren wird als *Broyden-Rank 1-Updates* bezeichnet; es ist möglich, superlineare Konvergenz des Verfahrens zu beweisen, siehe wieder z.B. [8].

Bemerkung 1.2: Beide der vorgestellten Modifikationen lassen den Tangentialraum invariant:

$$M_0\, t_0 = 0 \;\Rightarrow\; M_k\, t_0 = 0$$

Dies ist für das vereinfachte Verfahren offensichtlich; für die Broyden-Updates folgt es aus der Verwendung der Pseudoinversen als Löser; dadurch sind die Korrektur $\widehat{\Delta y}^k$ und der Tangentialvektor t_k orthogonal zueinander; da aber ein Broyden-Update $B_{k+1} - B_k$ senkrecht zu $\widehat{\Delta y}^k$ nicht ändert, gilt $t_{k+1} = t_k = \ldots = t_0$. Dadurch können beide Verfahren als gewöhnliche Newton-Verfahren auf der Hyperebene $H = \mathcal{R}(t_0)^\perp$ aufgefaßt werden, was zeigt, daß bei diesen Verfahren die Vorteile der impliziten Reparametrisierung in jedem Schritt wieder verloren gehen.

Bemerkung 1.3: Gelegentlich wird in der Literatur gegen die Broyden-Updates ins Feld geführt, daß diese bei Verwendung von direkten Lösungsverfahren für die Jacobimatrix am Startpunkt $f'(y^0)$ auch zu Updates der Faktorisierung von $f'(y^0)$ führen; dies ist jedoch nicht erforderlich, wie bereits Broyden in dem zitierten Artikel ausführt.

Statt dessen läßt sich aus der Einsicht, daß es sich um ein gewöhnliches Newton-Verfahren auf H handelt, und aus einem Spezialfall der Sherman-Morrison-Woodbury-Formel $(Id - uv^*)^{-1} = Id + \frac{1}{1-u^*v} uv^*$ (siehe z.B. [19]) die Pseudoinverse von

$$B_{k+1} = B_k + \frac{f(y^{k+1})\widehat{\Delta y}^{k^*}}{\widehat{\Delta y}^{k^*}\widehat{\Delta y}^k} = B_k \left(Id - \frac{\overline{\Delta y}^{k+1}\widehat{\Delta y}^{k^*}}{\widehat{\Delta y}^{k^*}\widehat{\Delta y}^k} \right)$$

wobei hier $\overline{\Delta y}^{k+1} = -f'(y^k)^+ f(y^{k+1})$ die vereinfachte Newton-Korrektur bzgl. y^k bezeichnet, durch

$$B_{k+1}{}^+ = \left(Id + \frac{\overline{\Delta y}^{k+1}\widehat{\Delta y}^{k^*}}{\widehat{\Delta y}^{k^*}\widehat{\Delta y}^k(1-\alpha)} \right) B_k{}^+ \tag{1.14}$$

aus $B_k{}^+$ berechnen, wobei

$$\alpha := \frac{\overline{\Delta y}^{k+1*}\widehat{\Delta y}^k}{\widehat{\Delta y}^{k*}\widehat{\Delta y}^k} \neq 1 \tag{1.15}$$

sicher gilt, falls $\|\overline{\Delta y}^{k+1}\| < \|\widehat{\Delta y}^k\|$, was wegen der weiter unten motiverten Abruchbedingung $\|\overline{\Delta y}^{k+1}\| \leq \overline{\Theta}\|\widehat{\Delta y}^k\|$ mit $\overline{\Theta} = 1/2$ während der Iteration erfüllt bleibt.

Besonders einfach ist die Berechnung von

$$\begin{aligned}\widehat{\Delta y}^{k+1} = -B_{k+1}{}^+ f(y^{k+1}) &= \left(Id + \frac{\overline{\Delta y}^{k+1}\widehat{\Delta y}^{k*}}{\widehat{\Delta y}^{k*}\widehat{\Delta y}^k(1-\alpha)}\right)\overline{\Delta y}^{k+1} \\ &= \left(1 + \frac{\alpha}{1-\alpha}\right)\overline{\Delta y}^{k+1} = \frac{1}{1-\alpha}\overline{\Delta y}^{k+1} \end{aligned} \tag{1.16}$$

wodurch sich 1.14 zu

$$B_{k+1}{}^+ = \left(Id + \frac{\widehat{\Delta y}^{k+1}\widehat{\Delta y}^{k*}}{\widehat{\Delta y}^{k*}\widehat{\Delta y}^k}\right) B_k{}^+$$

vereinfacht.

Damit läßt sich nur durch Abspeichern von $f'(y^0)$, $\widehat{\Delta y}^i$und$\|\widehat{\Delta y}^i\|^2$ für $i = 0 \ldots k$ die Berechnung von $\overline{\Delta y}^{k+1}$, $\widehat{\Delta y}^{k+1}$ aufgrund von (1.14) und (1.16) effizient programmieren:

Schritt I) $v_0 = -f'(y^0)^+ f(y^{k+1})$

Schritt II) für $i = 1, \ldots, k$, berechne

i) $\beta = (v_{i-1}{}^* \widehat{\Delta y}^{i-1})/\|\widehat{\Delta y}^{i-1}\|^2$

ii) $v_i = v_{i-1} + \beta\widehat{\Delta y}^i$

Schritt III)

i) $\overline{\Delta y}^{k+1} = v_k$

ii) $\alpha = (v_k{}^* \widehat{\Delta y}^k)/\|\widehat{\Delta y}^k\|^2$

iii) $\Theta_k = \|v_k\|/\|\widehat{\Delta y}^k$

Falls $\Theta_k \leq 1/2$ berechne $\widehat{\Delta y}^{k+1} = \frac{1}{1-\alpha}\overline{\Delta y}^{k+1}$

1.2.5 Konvergenztheorie und Schrittweitensteuerung

Zurück zum Prediktor, bei dem noch eine Frage offen geblieben war: die Wahl der Schrittweite. Falls die Schrittweite zu groß gewählt wird, konvergiert sicherlich der Korrektor nicht mehr und das Verfahren bricht zusammen; wird die Schrittweite hingegen zu klein gewählt, wird in vielen kleinen Schritten viel überflüssige Arbeit geleistet; denn unabhängig von der Realisierung des Korrektors ist zumindest eine Auswertung der Ableitung in jedem Schritt nötig.

Die Idee der verwendeten Schrittweitensteuerung besteht darin, gerade die Schrittweite zu nehmen, bei der aufgrund der Konvergenzsätze noch Konvergenz des Korrektors beweisbar ist. Die Aussagen werden aus zwei Konvergenzsätzen gewonnen, in denen etwas unhandliche aussehende Lipschitz-Bedingungen auftauchen, die aber zum Zwecke des Beweises optimiert sind und deren Lipschitz-Konstanten, durch numerisch leicht zugängliche Schätzer approximierbar, in den verwendeten Algorithmus eingehen. Besonders sei darauf hingewiesen, daß die angegebenen Bedingungen keinen Gebrauch von Normen des Bildraumes machen und dadurch die Invarianz sowohl der Problemstellung als auch des Verfahrens widerspiegeln.

Der erste Satz macht Aussagen darüber, „wie weit“ sich der Prediktor $\hat{y}_{n+1}$ vom Lösungspfad entfernen darf, ohne die Konvergenz des Korrektors zu zerstören; dabei ist die „Entfernung“ nicht so sehr im Abstand $\|\hat{y}_{n+1} - y_n\|$ als vielmehr in der Größe der auftretenden Lipschitz-Konstanten zu verstehen, die als Maß für die Nichtlinearität von f dienen.

Der zweite Satz macht hingegen Aussagen darüber, wie weit sich der (tangentiale) Prediktor tatsächlich bei einer bestimmten Schrittweite vom Lösungspfad entfernt hat.

Zusammen ergibt sich als Korollar die maximal mögliche Schrittweite.

Der erste Satz stammt in seiner ursprünglichen Form von Mysovskii aus dem Jahre 1949, wobei allerdings eine etwas andere Lipschitz-Bedingung angegeben ist. Die Beweistechnik geht auf Kantorowitch zurück und besteht darin, zu zeigen, daß die Iterationsfunktion der Newton-Iteration $\Phi(y) = y - f'(y)^+ f(y)$ für geeignete Startwerte kontraktiv ist und die Normen der Iterationskorrekturen $\|\Delta y^k\|$ eine quadratisch konvergente Majorante haben; die Glieder der Majorante heißen Kantorowitch-Größen. Die Voraussetzungen sind etwas stärker als in Kantorowitch's Satz, was die Beweisführung erheblich vereinfacht.

Die hier benutzte Version ist aus [8] bzw. [11] entnommen, in denen der Beweis für $n = m$ geführt wird; außer daß dabei noch eine lokale Eindeutigkeit der Lösung herausspringt (wovon hier keine Rede sein kann), unterscheidet sich der Beweis nicht von dem hier angegebenen.

Satz 1.2 *(Newton-Mysovskii) Sei $f\colon \mathbb{R}^n \mapsto \mathbb{R}^m$ mit $n \geq m$ eine stetig differenzierbare Abbildung mit* $\operatorname{rang}(f'(y)) = m$ *für alle $y \in \mathbb{R}^n$.*

Es gelte die Bedingung (mit der Lipschitz-Konstante ω_H)

$$\|f'(y)^+(f'(u+sv) - f'(u))v\| \leq s\omega_H \|v\|^2 \tag{1.17}$$

für alle $s \in [0,1]$ und für alle $u \in \mathbb{R}^n$, $v \in \mathcal{R}(f'(u)^+)$, $y \in u + \mathcal{R}(v)$.

Falls dann der Startpunkt für die Gauß-Newton-Iteration so gewählt ist, daß

$$\|\Delta y^0\| = \|f'(y^0)^+ f(y^0)\| \leq \alpha \tag{1.18}$$

und

$$h_0 := \frac{1}{2}\omega_H \alpha < 1, \tag{1.19}$$

dann gelten die folgenden Aussagen:

a) $\|y^{k+1} - y^k\| \leq \frac{\omega_H}{2}\|y^k - y^{k-1}\|^2$ *für alle $k > 0$*

b) $\frac{\omega_H}{2}\|y^{k+1} - y^k\| \leq h_k$, *wobei die Kantorowitch-Größen h_k rekursiv durch $h_{k+1} := {h_k}^2$ definiert sind*

c) Die Iterierten y^k streben gegen einen Grenzwert $\lim y^k = y^*$ *mit $f(y^*) = 0$; dabei entfernen sich alle Iterierten nicht mehr als*

$$\|y^k - y^0\| < \frac{\alpha}{1 - h_0} \quad \text{für} \quad k = 1, 2, \ldots$$

vom Startpunkt y^0.

Beweis: Der eigentliche Teil der Arbeit besteht im Beweis der Abschätzung a):

Da die Korrektur $\Delta y^k = y^{k+1} - y^k$ sich von einer Differenz in ein Integral umformen läßt:

$$\begin{aligned}
y^k - y^{k+1} &= f'(y^k)^+ f(y^k) \\
&= f'(y^k)^+ f(y^k) - f'(y^k)^+ \underbrace{\left(f(y^{k-1}) + f'(y^{k-1})\Delta y^{k-1}\right)}_{0} \\
&= f'(y^k)^+ \left(f(y^{k-1} + \Delta y^{k-1}) - f(y^{k-1}) - f'(y^{k-1})\Delta y^{k-1}\right) \\
&= \int_0^1 \left[f'(y^k)^+ \left(f'(y^{k-1} + s\Delta y^{k-1}) - f'(y^{k-1})\right)\right] \Delta y^{k-1} ds
\end{aligned}$$

kann nun die Lipschitz-Bedingung (1.17) nach Anwendung der Dreiecksungleichung $\|\int f\| \leq \int \|f\|$ zuschlagen:

$$\begin{aligned}
\|y^{k+1} - y^k\| &\leq \int_0^1 \| \left[f'(y^k)^+ \left(f'(y^{k-1} + s\Delta y^{k-1}) - f'(y^{k-1})\right)\right] \Delta y^{k-1} \| ds \\
&\leq \int_0^1 s\omega_H \|\Delta y^{k-1}\|^2 ds = \frac{\omega_H}{2} \|y^k - y^{k-1}\|^2 \qquad (1.20)
\end{aligned}$$

Aussage b) hingegen ist eine kurze Induktion: für $k = 0$ ist sie Bestandteil der Voraussetzungen; für $k > 0$ folgt aus (1.20) induktiv:

$$\frac{\omega_H}{2} \|\Delta y^k\| \leq \left(\frac{\omega_H}{2}\right)^2 \|\Delta y^{k-1}\|^2 = h_{k-1}^2 =: h_k$$

Damit bildet die Reihe h_k eine Majorante in der Norm für die Reihe der Newton-Iterierten $y^k - y^0 = \sum_{l=0}^{k-1} \Delta y^k$, die wegen der Voraussetzung $h_0 < 1$ konvergiert mit der Abschätzung

$$\sum_{k=0}^{\infty} h_k = \sum_{k=0}^{\infty} h_0^{2^k} < \frac{h_0}{1 - h_0}.$$

Daraus folgt neben der Konvergenz nach dem Majorantenkriterium auch die Abschätzung

$$\|y^k - y^0\| \leq \sum_{l=0}^{k-1} \|\Delta y^l\| \leq \frac{2}{\omega_H} \sum_{l=0}^{k-1} h_l < \frac{(2/\omega_H)\, h_0}{1 - h_0} = \frac{\alpha}{1 - h_0}$$

für alle $k \in \mathbb{N}$ und aus Stetigkeitsgründen auch für den Grenzwert.
Schließlich gilt wegen $f'(y^k)^+ f(y^k) = -\Delta y^k \overset{k\to\infty}{\longrightarrow} 0$ aus Stetigkeitsgründen

$$f'(y^*)^+ f(y^*) = 0 \implies f(y^*) = 0$$

□

Bemerkung 1.4: Bei genauem Durchsehen läßt sich feststellen, daß der Beweis bei der Pseudo-Inversen nur von der Eigenschaft $A^+AA^+ = A^+$ Gebrauch macht. Tatsächlich gilt der Satz also für alle Pseudo-Inversen, die nur diese Gleichung erfüllen, etwa auch für die, die im Kontext der Pseudo-Bogenlängen-Verfahren durch Lösung der Gleichungssysteme (1.7) definiert sind. Damit läßt im Prinzip sich die in diesem Kapitel zu entwickelnde Schrittweitensteuerung auch auf diese Methoden anwenden.

Allerdings führt die Form der Abschätzung geradewegs zur Moore-Penrose-Pseudo-Inversen, weil diese zur Lösung von $Ax = b$ die Norm $\|x\|$ minimiert. Damit werden aber gerade die linken Seiten von (1.17) und (1.18), und damit α und ω_H und weiter h_0 und alle h_k minimiert. Folglich sorgt diese Pseudoinverse für die schnellste Konvergenz und in Verbindung mit dem tangentialen Prediktor für maximale Schrittweiten.

Aus dem Satz folgt, daß das Gauß-Newton-Verfahren konvergiert, wenn die Bedingung $h_0 < 1$ und weiter $h_k < 1$ für alle k erfüllt ist. Um die Konvergenz des Verfahrens zu testen, werden diese h_k geschätzt. Dazu werden die Ungleichungen a) und b) aus Satz 1.2 kombiniert, um den Schätzer

$$[h_k] := \frac{\|\Delta y^{k+1}\|}{\|\Delta y^k\|} \leq \frac{h_{k+1}}{h_k} = h_k$$

zu erhalten.

Dieser Schätzer benötigt allerdings die Auswertung von $f'(y^{k+1})$, um Δy^{k+1} zu berechnen; insbesondere im ersten Schritt der Iteration ($k = 0$), in dem die Divergenz meist auftritt, verdoppelt sich der Aufwand faktisch, da die Auswertung und Zerlegung von f' den Löwenanteil der Arbeit ausmacht. Statt dessen wird Δy^1 durch die vereinfachte Newton-Korrektur $\overline{\Delta y}^1$ ersetzt, für die auch die Abschätzung

$$\|\overline{\Delta y}^1\| \leq \frac{\omega_H}{2}\|\Delta y^0\|^2 \tag{1.21}$$

gilt, die sich genauso beweist wie (1.20).

Damit läßt sich der *restringierte Monotonietest*

$$\Theta_0 := \frac{\|\overline{\Delta y}^1\|}{\|\Delta y^0\|} \leq \overline{\Theta} < 1 \tag{1.22}$$

durchführen, wobei sich der Faktor $\overline{\Theta} = 1/2$ durch genauere Analyse des Konvergenzverhaltens und der Bedingung, daß die Iterierten sich auch zur

Lösung bewegen, ergibt (siehe [8]). Er kann aber auch als Sicherheitsfaktor interpretiert werden, so daß $\Theta_k < \overline{\Theta} h_k$. Falls anstelle des exakten Gauß-Newton-Verfahrens die Vereinfachung der Broyden-Updates gewählt wird, ergibt sich aus (1.16), daß sich die Implikation

$$\Theta_0 = \frac{\|\overline{\Delta y}^1\|}{\|\widehat{\Delta y^0}\|} < \overline{\Theta} \quad \Longrightarrow \quad [h_0] = \frac{\|\widehat{\Delta y}^1\|}{\|\widehat{\Delta y^0}\|} < 1$$

für $\overline{\Theta} = 1/2$ gerade noch beweisen läßt.

Äquivalent werden die weiteren h_k durch die entsprechenden $\Theta_k = \|\overline{\Delta y}^{k+1}\|/\|\Delta y^k\|$ geschätzt und für den Fall, daß $\Theta_k > \overline{\Theta}$, die Newton-Iteration abgebrochen. Dabei ist hier $\overline{\Delta y}^{k+1}$ die vereinfachte Newton-Korrektur bezüglich der k-ten Ableitung.

Der zweite Satz, der dem Artikel [9] entnommen ist, macht eine Aussage darüber, wie groß $\alpha = \|\Delta y^0\|$ in Abhängigkeit von der Schrittweite s ist; die Beweistechnik ist dabei dieselbe wie im Satz 1.2, Teil a).

Satz 1.3 *Sei f wie in Satz 1.2, $u \in \mathbb{R}^n$ mit $f(u) = 0$ und t_u ein Tangentialvektor von $f'(u)$. Des weiteren sei*

$$y^0 = \hat{y}(s) = u + st_u$$

der tangentiale Prediktor zur Schrittweite s.

Falls nun f die Lipschitz-Bedingung zur Konstante ω_t erfüllt:

$$\|f'(y^0)^+(f'(u + \sigma t_u) - f'(u))t_u\| \leq \sigma\omega_t\|t_u\|^2 \tag{1.23}$$

für alle $\sigma \in [0, s]$, dann ist die erste Korrektur Δy^0 des in y^0 gestarteten Gauß-Newton-Verfahrens in der Norm beschränkt durch

$$\|\Delta y^0\| = \alpha \leq \frac{1}{2}\omega_t s^2\|t_u\|^2 \tag{1.24}$$

Beweis: (aus [9])

Da sich die Newton-Korrektur wieder in ein Integral überführen läßt:

$$\begin{aligned} -\Delta y^0 &= f'(y^0)^+ f(y^0) &&= f'(y^0)^+(f(y^0) - \underbrace{f(u)}_{0}) \\ &&&= \int_0^s f'(y^0)^+ f'(u + \sigma t_u)\, t_u\, d\sigma \end{aligned}$$

läßt sich die Bedingung (1.23) ausnutzen für die Abschätzung:

$$\begin{aligned} \alpha = \|\Delta y^0\| &\leq \int_0^s \|f'(y^0)^+ f'(u+\sigma t_u)\, t_u\|\, d\sigma \\ &\leq \int_0^s \sigma\omega_t \|t_u\|^2 d\sigma = \frac{1}{2}\omega_t s^2 \|t_u\|^2 \end{aligned}$$

welche gerade zu zeigen war. □

Korollar 1.4 *Falls $f\colon \mathbb{R}^{n+1} \mapsto \mathbb{R}^n$ stetig differenzierbar ist, und die Ableitung überall vollen Rang hat sowie die Bedingungen (1.17) und (1.23) erfüllt, dann konvergiert das Gauß-Newton Verfahren als Korrektor zum tangentialen Prediktor, wenn die Schrittweite s des Prediktors kleiner ist als*

$$s_{max} = \frac{2}{\sqrt{\omega_t\, \omega_H}}$$

Denn dann ist nach Satz 1.3

$$h_0 := \frac{1}{2}\omega_H \alpha \leq \frac{1}{4}\omega_H \omega_t s^2 < \frac{1}{4}\omega_H \omega_t s_{max}^2 = 1$$

womit (1.19) und somit alle Bedingungen von Satz 1.2 erfüllt wären.

Um das erhaltene Ergebnis in einen Algorithmus umsetzen zu können, ist es notwendig, die Lipschitz-Konstanten ω_t und ω_H durch berechenbare Größen $[\omega_t]$ bzw. $[\omega_H]$ zu schätzen, um daraus den Schrittweitenvorschlag

$$[s_{max}] := \frac{2}{\sqrt{[\omega_t]\,[\omega_H]}}$$

zu erhalten.

Dabei sind zwei verschiedene Schätzer zu unterscheiden:

- ein Schätzer, der bei erfolgreich verlaufener Berechnung $\hat{y}_\nu \Rightarrow y_\nu$ zur Schrittweite s_ν aus den Informationen des vorhergehenden Schrittes eine neue Schrittweite $s_{\nu+1}$ schätzt, d.h. ein *a-priori-Schätzer*

- ein Schätzer, der bei fehlender Konvergenz zur Schrittweite $s_{\nu,fail}$ aus der Information der Divergenz des aktuellen Schrittes eine neue Schrittweite s_ν vorschlägt, d.h. ein *a-posterori-Schätzer*

Dabei wird der Schätzwert der theoretisch größt möglichen Schrittweite mit einem Sicherheitsfaktor ρ versehen, z.B. $\rho = 1/\sqrt{2}$ oder $= 1/2$:

$$s_\nu \text{ bzw. } s_{\nu+1} = \rho \frac{2}{\sqrt{[\omega_t]\,[\omega_H]}}$$

Zunächst wird der a-posterori-Schätzer vorgestellt:

Dem Schätzer liegt die Idee zugrunde, daß das Gauß-Newton Verfahren abgebrochen wird, wenn der restringierte Monotonietest $\Theta_k > \overline{\Theta}$ verletzt ist; sei angenommen, daß dies in ersten Schritt geschehen ist.

Als Schätzer für ω_H ergibt sich dann unter Berücksichtigung der Annahme $\Theta_0 \leq \overline{\Theta} h_0$ aus Gleichung (1.19)

$$[\omega_H] =: 2\frac{\Theta_0/\overline{\Theta}}{\|\Delta y^0\|} \leq \omega_H \tag{1.25}$$

Für ω_t ergibt sich ähnlich aus (1.24) der Schatzer

$$[\omega_t] := 2\frac{\|\Delta y^0\|}{s_\nu^2} \leq \omega_t \tag{1.26}$$

so daß sich insgesamt die Schrittweite

$$[s_{max}] := \rho\sqrt{\frac{\overline{\Theta}}{\Theta_0}} \tag{1.27}$$

ergibt, die für den Fall, daß der Monotonietest (1.22) erst im $k-$ten Schritt verletzt wird, zu

$$[s_{max}] := \rho\sqrt{\frac{\overline{\Theta}}{\Theta_k}} s_\nu$$

verallgemeinert wird; in jedem Fall ist vernünftigerweise

$$[s_{max}] < s_\nu.$$

Für den a-priori-Schätzer läßt sich der zum einem auch der aposterori-Schätzer (1.27) verwenden ([12],[40]), wobei i.allg. wegen $\Theta_0 \leq \overline{\Theta}$ auch

$[s_{max}] \geq s_\nu$ möglich ist; es ist aber auch möglich, die lokale Krümmung in den a-priori-Schätzer einzubeziehen; dadurch ergibt sich der Schätzer

$$[\omega_t] := \frac{2c_0\|y_\nu - \hat{y}_\nu\|}{s_\nu^2} \tag{1.28}$$

wobei $c_0 = \hat{t}_\nu^{\,*} t_\nu$ der Cosinus des Winkels zwischen den Tangenten des Prediktors $\hat{y}_\nu$ und des Startpunktes $y_{\nu-1}$ und $\|y_\nu - \hat{y}_\nu\|$ der Abstand des Prediktors zum Konvergenzpunkt des Korrektors ist. Einen Beweis diese Abschätzung findet sich in [8] oder [9].

Wenn der alte Schätzer $[\omega_H]$ aus (1.25) weiterverwendet wird, ergibt sich aus dieser Schätzung als a-priori-Schrittweite:

$$s_{\nu+1} = [s_{max}] := \rho \left(\frac{\|\Delta y^0\|}{\|y_\nu - \hat{y}_\nu\|} \frac{\overline{\Theta}}{\Theta_0} \frac{1}{c_0} \right)^{1/2} s_\nu \tag{1.29}$$

Bemerkung 1.5: Damit sich aufgrund von möglichen Fehlschätzungen die Schrittweite nicht zu abrupt ändert, werden für den a-priori-Schätzer (1.29) die Randbedingungen

$$\text{pred}_{min} \leq \frac{s_{\nu+1}}{s_\nu} \leq \text{pred}_{max}$$

und für den a-posterori-Schätzer

$$\text{corr}_{min} \leq \frac{s_\nu}{s_{\nu,fail}} \leq \text{corr}_{max}$$

verlangt, wobei bei den Beispielrechnungen $\text{pred}_{min} = \text{corr}_{min} = 1/10$, $\text{pred}_{max} = 2$ und $\text{corr}_{max} = 7/10$ benutzt wurde.

Die Schrittweitensteuerung wurde unter der ständigen Voraussetzung $\text{rang}\,(f'(y)) = n$ entwickelt, was die lokale Eindeutigkeit des Lösungspfades garantiert.

Hat hingegen $f'(y)$ für ein y einen Rangdefekt, so bedeutet dies den Verlust der lokalen Eindeutigkeit. Damit das Verfahren nicht vom aktuellen Lösungsast abspringt, sind zusätzliche Schrittweiteneinschränkungen notwendig.

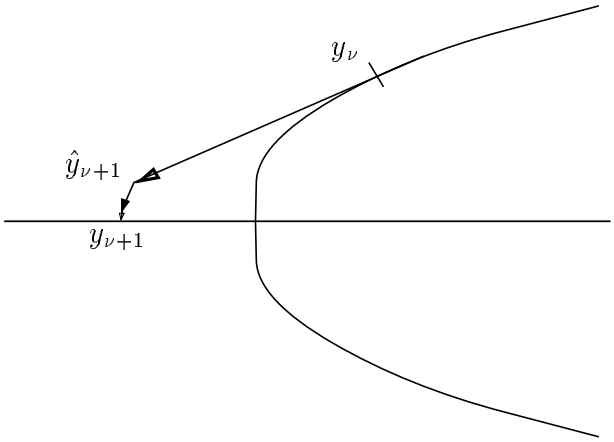


Abbildung 1.4: Abspringen von einem Lösungsast nahe einer Singularität

Eine einfache, aber sehr effektive Einschränkung, die aus [17] stammt, besteht darin, die Kondition der Ableitung entlang eines Pfades zu schätzen. Da an Singularitäten asymptotisch die Relation

$$\mathrm{cond} f'(y) \sim \frac{1}{s} \tag{1.30}$$

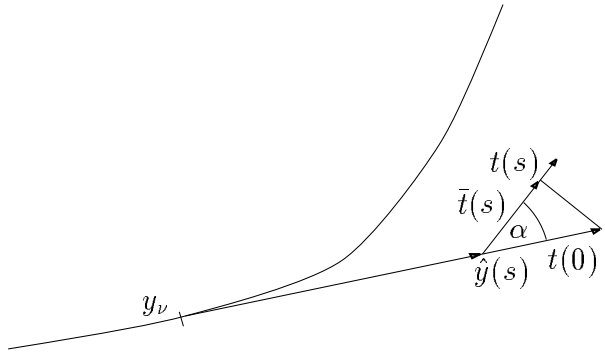
gilt, wobei $s = \|y - y^\star\|$ die Entfernung zur Singularität $y^\star$ bezeichnet, ist es eine plausible Schrittweiteneinschränkung, daß der Prediktor $\hat{y}_{\nu+1}$ nicht weiter von der Lösung y_ν entfernt sein soll als die Singularität. Falls die Kondition im Vergleich zum vorherigen Schritt angestiegen ist, also

$$\mathrm{cond}_2\left(f'(y_{\nu-1})\right) / \mathrm{cond}_2\left(f'(y_\nu)\right) =: \alpha < 1,$$

dann läßt sich aus der Annahme (1.30) die Singularität zu $y^\star = y_\nu + \frac{\alpha}{1-\alpha}(y_\nu - y_{\nu-1})$ extrapolieren. Aus der Bedingung, daß keine Komponente des Prediktors von y_ν weiter als die entsprechende Komponente von $y^\star$ entfernt sein soll, ergibt sich die Schrittweiteneinschränkung

$$(s_\nu t_\nu)_k < \frac{\alpha}{1-\alpha} \left|y_\nu - y_{\nu-1}\right|_k \qquad \text{für alle } k = 1 \ldots n+1$$

Eine weitere Schrittweiteneinschränkung nahe Singularitäten ergibt sich aus der Tatsache, daß die Tangente beim Abspringen von einem Pfad sich



unstetig ändert. Diese Unstetigkeit kann schon an der Tangente entlang des Prediktors beobachtet werden, indem die Tangente des Prediktors einen ungewöhnlich großen Winkel mit der Tangente der letzten Lösung einschließt.

Der folgende Satz führt zu einem Schätzer für dem Winkel zwischen den Tangenten; dieser Schätzer kann dann mit dem tatsächlich angetroffenen Winkel verglichen werden.

Sei dazu $\hat{y}(s) = y_\nu + st_\nu$ der tangentiale Prediktor zur Schrittweite s und $t(s)$ die Tangente von $f'(\hat{y}(s)) = f'(s)$. Dann läßt sich der Sinus des Winkels, der zwischen $t_\nu = t(0)$ und $t(s)$ liegt, als die Norm der Differenz zwischen der orthogonalen Projektion $\bar{t}(s)$ von $t(0)$ auf $\mathcal{R}(t(s))$ und $t(0)$ berechnen. Da aber aufgrund der Projektionseigenschaften der Pseudoinversen (siehe Abschnitt 3.1) $\bar{t}(s) = \mathcal{P}_{t(s)}t(0) = (Id - f'(s)^+ f'(s))t(0)$ gilt, folgt der Satz:

Satz 1.5 *Falls für f die Bedingungen von Satz 1.3, insbesondere die Ungleichung 1.23 erfüllt sind, dann gilt für den Winkel α zwischen $t(0)$ und $\bar{t}(s)$ die Abschätzung*

$$\sin\alpha = \|t(0) - \bar{t}(s)\| \leq s\omega_t \tag{1.31}$$

Beweis: Da wegen $f'(0)t(0) = 0$:

$$\begin{aligned} t(0) - \bar{t}(s) &= f'(s)^+ f'(s)t(0) \\ &= f'(s)^+(f'(s) - f'(0))t(0) \end{aligned}$$

läßt sich nach Vorraussetzung 1.23 mit $y^0 = y(s)$, $u = y_\nu$ die Abschätzung

$$\|t(0) - \bar{t}(s)\| = \|f'(s)^+(f'(s) - f'(0))t(0)\| \leq s\omega_t$$

zeigen, die zu beweisen war. □

Der Test besteht nun darin, den Sinus des Winkels α aus entweder dem a-priori-Schätzer (1.28) oder dem a-posterori-Schätzer (1.26) zu schätzen und mittels dem Kosinus des tatsächlichen Winkels $\cos\alpha = t(s)^* t(0)$ die Abweichung zu ermitteln. Ist die Abweichung größer als ein bestimmter Winkel $\Delta\alpha_{max}$, wird die Schrittweite reduziert, indem der Schätzer $[\omega_t]_{tan} = \sqrt{1-\cos^2\alpha}/s$ berechnet wird. Ist $[\omega_t]_{tan} > [\omega_t]$, dann wird die neue Schrittweite zu

$$[s_{max}] = \sqrt{\frac{[\omega_t]}{[\omega_t]_{tan}}} s_\nu \tag{1.32}$$

berechnet. Sonst wird die Schrittweite einfach um den Faktor corr_{max} zusammengezogen. Als $\Delta\alpha_{max}$ wurde $0.2 \approx \frac{\pi}{15}$ verwendet.

Als Schätzer für $\sin\alpha$ erweist es sich als günstiger, den a-posterori-Schätzer zu verwenden, da der a-priori-Schätzer zu wenig Information über der Prediktor enthält; der a-priori-Schätzer ist zwar theoretisch zuverlässiger als der a-posterori-Schätzer, dessen Voraussetzungen bei Anschlagen des Tangententests eigentlich schon nicht mehr erfüllt sind, aber diese Robustheit erwies sich nicht als notwendig; statt dessen führte der auf dem a-priori-Schätzer basierende Tangententest zu oft zu unnötigen Schrittweiteneinschränkungen. Daher wird, auch wegen Bemerkung 4.1, nur die Verwendung der a-posterori-Schätzers enpfohlen.

Bemerkung 1.6: Aus dem Beweis des Satzes 1.5 ergibt sich eine gute Approximation für die erste Newton-Korrektur Δy^0 aus:

$$\begin{aligned}\Delta y^0 &= -\int_0^s f'(s)^+ f'(\sigma)\, t(0)\, d\sigma \\ &\approx -f'(s)^+ \frac{s}{2}\,(f'(s) - f'(0))\, t(0)\, d\sigma = \frac{s}{2}(\bar{t}(s) - t(0)),\end{aligned}$$

wobei das Integral mit Hilfe der Trapezregel näherungsweise gelöst wurde. Diese Approximation kann zum Beispiel als Startwert für die iterative Berechnung der Pseudoinversen dienen. Der relative Fehler lag dabei abhängig vom Sicherheitsfaktor ρ bei numerischen Experimenten im Mittel bei 20% für $\rho = 0.5$ und bei 6% für $\rho = 0.1$, weitgehend unabhängig vom berechneten Problem. Dabei bestand der Fehler zumeist in der Länge der Approximation, die durch Verwendung einer anders gewichteten Quadraturformel beliebig verändert werden könnte, während der Fehler in der Richtung meist weniger als 8° betrug.

2. Singularitäten

2.1 Typen und Normalformen von Singularitäten

Von besonderem Interesse bei der Pfadverfolgung sind Lösungspunkte, bei denen die Bedingung des vollen Rangs im impliziten Funktionen Theorem 1.1 verletzt ist. Denn zum einen kann das Verfahren an solchen Punkten entweder zusammenbrechen oder unvollständige Lösungen liefern, wenn diese nicht berücksichtigt werden. Zum anderen haben solche Punkte für die Interpretation der Ergebnisse in den zugrundeliegenden Modellen meist im Vergleich zu regulären Lösungspunkten eine herausragende Stellung, so daß ihre Berechnung von besonderem Interesse ist.

Dabei lassen sich zwei Perspektiven unterscheiden, unter denen ein Punkt als singulär angesehen werden muß, je nach dem, ob der Parameter λ eine besondere Rolle spielt oder nicht.

Insbesondere bei Modellen von Typ (1.2) stellt der Parameter λ oft eine von außen kontrollierbare Größe dar, während sich die Lösungen mit diesem Parameter verändern. Dies geschieht auf stetige Weise, wenn eine Parametrisierung der Lösung als $x = x(\lambda)$ existiert. Anderenfalls kann sich die Lösung auf unstetige Weise ändern, was für die physikalischen Systeme drastische Auswirkungen haben kann; nicht zufällig heißt der Zweig der theoretischen Physik, der sich mit solchen Phänomenen befaßt, „Katastrophentheorie“.

Lösungspunkte, in denen Satz 1.1 die Existenz einer Lösung nicht garantiert, weil

$$\operatorname{rank}\left(\frac{\partial}{\partial x} f\right) < n$$

werden als *Singularitäten aus der λ-Perspektive* bezeichnet.

Dazu im Gegensatz werden solche Punkte, bei denen überhaupt die Existenz einer lokalen Parametrisierung $y = y(t)$ durch das implizite Funktionen Theorem nicht gesichert ist, da

$$\operatorname{rank}(f') < n$$

als *Singularitäten aus der y-Perspektive* bezeichnet.

Für den Algorithmus der tangentialen Fortsetzung haben dabei nur Singularitäten aus der y-Perspektive Bedeutung, während Singularitäten aus der λ-Perspektive nur bei Bedarf behandelt werden müssen.

In dieser Arbeit werden nur Singularitäten mit einfachen Rangdefekt, d.h. $\operatorname{rank}\left(\frac{\partial}{\partial x} f\right) = n - 1$ bzw. $\operatorname{rank}(f') = n - 1$ behandelt.

Die theoretischen Behandlung von Singularitäten, die hier dem Buch von Golubitsky und Schaeffer [20] folgt, besteht aus drei Schritten.

- Zunächst wird mittels der Technik der Ljapunov-Schmidt-Reduktion das Problem auf ein lokal äquivalentes Problem niedrigerer Dimension projiziert.

- Danach wird für dieses System ein äquivalentes System mit der topologisch gleichen Lösungsstruktur gesucht, welches aber eine möglichst einfache Form hat, etwa durch ein Polynom niedriger Ordnung dargestellt wird. Dieses System wird als *Normalform* der Singularität bezeichnet.

- Durch Untersuchung der Normalform lassen sich dann Rückschlüsse ziehen auf die Struktur der Lösung der ursprünglichen Singularität.

Die ersten beiden Schritte werden im folgenden Abschnitt skizziert; im Rest des Kapitels wird dann ohne Umschweife von der Normalform ausgehend ein Verfahren zur numerischen Behandlung der Singularität entwickelt.

Dabei gehören zur numerischen Behandlung zwei Schritte:

1. Ein möglichst einfaches Verfahren, um herauszufinden, ob zwischen zwei berechneten Punkten y_ν und $y_{\nu+1}$ auf dem Lösungspfad eine Singularität liegt. (Die Wahrscheinlichkeit, daß das Verfahren direkt eine Singularität als Lösungspunkt findet, ist sehr gering.)

2. Ein Verfahren, das die Singularität findet und gegebenenfalls die Pfadverfolgung von der Singularität aus neu startet.

Beide Schritte werden für Wendepunkte (Singularitäten aus der λ-Perspektive) und einfache Verzweigungen (Singularitäten aus der y-Perspektive) ausgeführt.

2.2 Ljapunov-Schmidt-Reduktion und Konstruktion der Normalform

Der erste Schritt, ein System von Gleichungen um einen singulären Punkt herum zu vereinfachen, besteht in der Ljapunov-Schmidt-Reduktion; dabei wird mit einem Trick auf dem regulären Teil der Ableitung das implizite Funktionen Theorem angewandt, was zu einer Reduktion in der Dimension um den Rang der Ableitung führt.

Das Verfahren wird zunächst aus der y-Perspektive skizziert. Sei im Punkt $y^\star$ die Ableitung $f'(y^\star)$ rangdefekt mit $\operatorname{rank}(f'(y^\star)) = n-1$; sei des weiteren f partitioniert als

$$\begin{aligned} f\colon \mathcal{N}(f'(y^\star))^\perp \times \mathcal{N}(f'(y^\star)) &\to \mathcal{R}(f'(y^\star)) \times \mathcal{R}(f'(y^\star))^\perp \\ f(\xi,\tau) &= \begin{cases} f_1(\xi,\tau) \in \mathcal{R}(f'(y^\star)) \\ f_2(\xi,\tau) \in \mathcal{R}(f'(y^\star))^\perp \end{cases} \end{aligned}$$

wobei hier $\xi \in \mathcal{N}(f'(y^\star))^\perp \cong \mathbb{R}^{n-1}$, $\tau \in \mathcal{N}(f'(y^\star)) \cong \mathbb{R}^2$. Damit läßt sich nun das implizite Funktionen Theorem auf f_1 anwenden, da nach Konstruktion $\operatorname{rank}\left(\frac{\partial}{\partial \xi} f_1(\xi^\star,\tau^\star)\right) = n-1$, also existiert

$$\xi = \xi(\tau) \quad \text{mit} \quad f_1(\xi(\tau),\tau) = 0 \quad \text{und} \quad \xi(\tau^\star) = \xi^\star$$

Einsetzen dieser Funktion $\xi(\tau)$ in f_2 definiert dann die reduzierte Funktion

$$f_{red}\colon \mathcal{N}(f'(y^\star)) \cong \mathbb{R}^2 \to \mathcal{R}(f'(y^\star))^\perp \cong \mathbb{R}$$

durch

$$f_{red}(\tau) = f_2(\xi(\tau),\tau)$$

Dann ist

$$f(\xi(\tau),\tau) = 0 \iff f_{red}(\tau) = 0 \qquad \text{und} \qquad f'_{red}(\tau^\star) = 0$$

Aus der λ-Perspektive gestaltet sich die Reduktion für Punkte $y^\star = (x^\star,\lambda^\star)$, bei denen $\operatorname{rank}\left(\frac{\partial}{\partial x} f(y^\star))\right) = n-1$ ist, äquivalent. Nach Einführung der Aufspaltung

$$\begin{aligned} f\colon \mathcal{N}(\tfrac{\partial}{\partial x} f(y^\star))^\perp \times \mathcal{N}(\tfrac{\partial}{\partial x} f(y^\star)) \times \mathbb{R} &\to \mathcal{R}(\tfrac{\partial}{\partial x} f(y^\star)) \times \mathcal{R}(\tfrac{\partial}{\partial x} f(y^\star))^\perp \\ f(\xi,\tau,\lambda) &= \begin{cases} f_1(\xi,\tau,\lambda) \in \mathcal{R}(\frac{\partial}{\partial x} f(y^\star)) \\ f_2(\xi,\tau,\lambda) \in \mathcal{R}(\frac{\partial}{\partial x} f(y^\star))^\perp \end{cases} \end{aligned}$$

läßt sich wieder wegen rank $\left(\frac{\partial}{\partial \xi} f_1\right) = n-1$ die implizite Funktion $\xi = \xi(\tau, \lambda)$ in f_2 einsetzen, welche sich zu $f_{red}\colon \mathcal{N}(\frac{\partial}{\partial x} f(y^\star)) \times \mathbb{R} \cong \mathbb{R}^2 \to \mathcal{R}(\frac{\partial}{\partial x} f(y^\star))^\perp \cong \mathbb{R}$ mit

$$f_{red}(\tau, \lambda) = f_2(\xi(\tau, \lambda), \tau, \lambda), \quad \frac{\partial}{\partial \tau} f_{red} = 0$$

reduziert.

Nachden so die Dimension des Systems reduziert ist, wird die reduzierte Funktion f_{red} mittels einer nichtlinearer Koordinatentransformation auf eine äquivalente Normalform φ, in der Regel ein Polynom niedrigen Grades, gebracht, wobei diese Äquivalenz weiter unten definiert wird.

Die zugelassenen Koordinatentransformationen unterscheiden sich dabei in der λ- und der y-Perspektive:

Aus der y-Perspektive sind zwei Funktionen g und h in obigem Sinne äquivalent, wenn es einen lokalen Diffeomorphismus $\Psi(y)$ und eine Skalierungsfunktion S mit $S(y) \neq 0$ für alle y gibt, so daß

$$h(y) = S(y) g(\Psi(y)),$$

während es aus der λ-Perspektive notwendig ist, für den Diffeomorphismus Ψ die Darstellbarkeit $\Psi(x, \lambda) = (\Xi(x, \lambda), \Lambda(\lambda))$ zu fordern, so daß sich die Äquivalenz als

$$h(x, \lambda) = S(x, \lambda) g(\Xi(x, \lambda), \Lambda(\lambda))$$

schreiben läßt. Interpretieren läßt sich diese Bedingung folgendermaßen: Wenn bei einer Funktion die Variable x variiert und der Parameter $\lambda = \lambda_0$ festgehalten wird, dann variiert bei allen äquivalenten Funktionen auch nur die Variable $\Xi(x, \lambda_0)$, während der Parameter $\Lambda(\lambda_0)$ fest bleibt.

Bemerkung 2.1: Falls das Gleichungssystem ein dynamisches System wie in Beispiel 1.1 modelliert, ist es nicht nur interessant, Gleichgewichtslösungen zu berechnen, sondern ihre Stabilität unter kleinen Störungen zu untersuchen. Als einfaches Kriterium für die Stabilität einer Lösung gilt, daß alle Eigenwerte von $\frac{\partial}{\partial x} f$ negativen Realteil haben. Soll diese Eigenschaft unter äquivalenten Funktionen erhalten bleiben, muß zusätzlich noch $\det(\Psi'(y)) > 0$ und $S(y) > 0$ gefordert werden, um die Struktur der Eigenwerte zu erhalten. Anderenfalls sind z.B. $f(y)$ und $-f(y)$ zueinander äquivalent, was einer Zeitumkehr entspricht; daher kann aber y nicht gleichzeitig ein stabiler Fixpunkt von f und $-f$ sein. Um die bei Stabilitätsbetrachtungen auftretenden

Komplikationen zu vermeiden, wird diese Zusatzannahme im Weiteren jedoch nicht gemacht.

Bemerkung 2.2: Nichts desto trotz kann mittels der Normalformen auf die Veränderung der Stabilität an einer Singularität Rückschlüse gezogen werden. Wechselt etwa bei der Normalform entlang eines Pfades ein Eigenwert sein Vorzeichen, dann wird bei einem Pfad des ursprünglichen Problems ebenfalls ein Eigenwert das Vorzeichen wechseln; jedoch ist nicht immer klar, in welche Richtung er wechselt.

Ist der Pfad vor der Singularität stabil, dann wird er nach der Singularität nicht mehr stabil sein, da einer der Eigenwerte, die vorher alle negativen Realteil hatten, jetzt einen positiven Realteil hat. Andersherum muß bei einem instabilen Pfad an der Singularität nicht notwenigerweise eine Wendung zur Stabilität geschehen; es kann auch einer der bisher in der linken Hälfte der komplexen Zahlenebene verbliebener Eigenwert die Seite wechseln und die instabile Mannigfaltigkeit um eine weitere Dimension vergrößern.

Des weiteren ist diese Form der Stabilitätsanalyse dadurch problematisch, daß auch an regulären Punkten die Stabilität sich ändert, falls eine *Hopf-Verzweigung* auftritt, bei der zwei zueinander konjugierte Eigenwerte über die imaginäre Achse wechseln. In diesem Punkt zweigt ein Pfad von periodischen Lösungen ab, die der stabilen Mannigfaltigkeit der stationären Lösung einen zweidimensionalen Unterraum stehlen oder hinzufügen; für die Pfadverfolgung bleibt diese Form von Verzweigung aber unsichtbar.

Bemerkung 2.3: Die Definition, daß in der λ-Perspektive $\Lambda(\lambda)$ nicht von x abhängt, führt dazu, daß die Anzahl $n_f(\lambda)$ verschiedener Lösungen x zu demselben Parameterwert λ für eine Funktion f unter den Äquivalenztransformationen invariant bleibt, d.h. falls g und h äquivalent sind, dann ist

$$n_h(\lambda) = n_g(\Lambda(\lambda)).$$

Der Nachweis, daß eine Funktion zu einer gegebenen Normalform äquivalent ist, wird mittels Taylor-Entwicklung geführt, wobei die höheren Ableitungen hinter einer von Null verschiedenen Ableitung lokal wegtransformiert werden können. Eine mathematisch präzise Rechtfertigung der Verwendung von Normalformen läßt sich in dem schon erwähnten Buch [20] finden.

2.3 Wendepunkte

Zunächst werden die einfachsten Singularitäten aus der $\lambda-$Perspektive untersucht. Sie werden aus Gründen, die beim Anblick der zugehörigen Normalform schnell verständlich werden, *Wendepunkte*, genauer *quadratische Wendepunkte* (engl. turning points) genannt.

Dabei sind Wendepunkte definiert als Lösungspunkte $y^\star = (x^\star, \lambda^\star)$, an denen gilt:

$$\operatorname{rank}\left(\frac{\partial}{\partial x} f(x^\star, \lambda^\star)\right) = n - 1 \quad \text{und} \quad \frac{\partial}{\partial \lambda} f(x^\star, \lambda^\star) \notin \mathcal{R}(\frac{\partial}{\partial x} f(x^\star, \lambda^\star)) \tag{2.1}$$

sowie eine Nichtdegeneriertheitsbedingung an die zweite Ableitung, nämlich, daß

$$\frac{\partial^2}{\partial x^2} f(x^\star, \lambda^\star) \neq 0 \quad \text{auf} \quad \mathcal{N}(\frac{\partial}{\partial x} f(x^\star, \lambda^\star)).$$

Damit ergibt sich nach der Ljapunov-Schmidt-Reduktion für die Taylor-Entwicklung der reduzierten Funktion, wenn o.B.d.A. $(x^\star, \lambda^\star) = (0, 0)$ angenommen wird:

$$f_{red}(x, \lambda) \;=\; \underbrace{f_{red}(0)}_{0} + \underbrace{\frac{\partial}{\partial x} f_{red}(0)}_{0}\, x + \frac{\partial^2}{\partial x^2} f_{red}(0) x^2 + \frac{\partial}{\partial \lambda} f_{red}(0) \lambda + \ldots$$

und als Normalform

$$\varphi_{turn}(x, \lambda) = x^2 - \lambda = 0 \tag{2.2}$$

Das negative Vorzeichen vor λ ist dabei konventionshalber gewählt; ein positives Vorzeichen ergibt eine alternative Normalform mit derselben Lösungsstruktur. Daraus ergeben sich als Lösungsmenge zwei Lösungspfade

$$x(\lambda) = \pm\sqrt{\lambda} \qquad \text{mit} \qquad \frac{\partial}{\partial x} \varphi_{turn} = 2x$$

die sich im singulären Punkt $(0, 0)$ treffen.

Dabei ist im oberen Lösungpfad $\frac{\partial}{\partial x}\varphi_{turn}$ positiv, im unteren negativ. Im allgemeinen unterscheiden sich die Lösungspfade in ihrem Eigenwertspektrum darin, daß einer der Zweige einen Eigenwert mit positivem Realteil mehr als der andere hat. Besteht z.B. einer der Zweige aus stabilen Lösungen, dann ist der andere als Folgerung von Bemerkung 2.2 instabil.

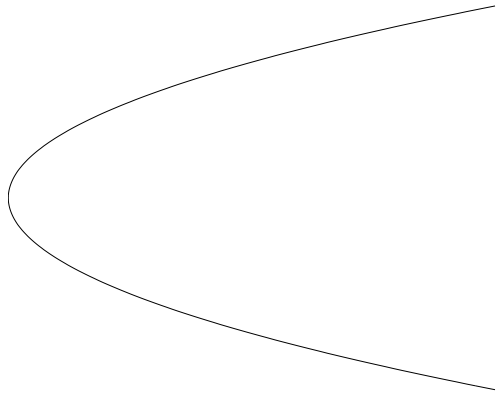


Abbildung 2.1: Normalform eines quadratischen Wendepunktes

Bemerkung 2.4: Die Bedingung $\frac{\partial}{\partial\lambda}f(x^\star,\lambda^\star) \notin \mathcal{R}(\frac{\partial}{\partial x}f(x^\star,\lambda^\star))$ bewirkt, daß $f'(y^\star)$ vollen Rang hat. Daher ist ein Wendepunkt kein singulärer Punkt aus der y-Perspektive. Dies ist auch an der Normalform ersichtlich, bei der sich die Lösungsmenge als $\lambda = \lambda(x) = x^2$ reparametrisieren läßt. Die Normalform in der y-Perspektive ist also, wie bei jedem regulären Punkt, eine Gerade $x = 0$.

Für die numerische Berechnung stellt sich nun die Frage, wie sich eine Singularität verhält, wenn das Problem leicht gestört wird. Für den Wendepunkt stellt sich dabei heraus, daß er als einzige von allen Singularitäten unter kleinen Störungen erhalten bleibt. Quadratische Wendepunkte sind eine *generische* Erscheinung.

Am einfachsten läßt sich das stabile Verhalten unter kleinen Störungen erkennen, wenn der Normalform eine Störung α aufaddiert wird, die wegen der lokalen Natur der Betrachtung konstant gewählt werden kann; durch die Störung

$$\tilde{\varphi}(x,\lambda) = x^2 - \lambda + \alpha$$

verschiebt sich die Lösung nur nach $x(\lambda) = \pm\sqrt{\lambda - \alpha}$ und der Wendepunkt nach $(x^\star,\lambda^\star) = (0,\alpha)$.

Bemerkung 2.5: Singularitäten, die die Normalform

$$x^k - \lambda = 0 \qquad \text{mit } k > 2$$

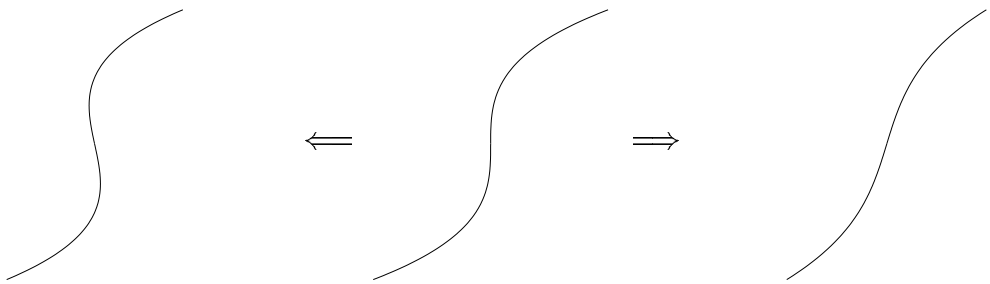


Abbildung 2.2: Kubischer Wendepunkt mit Störungen

besitzen, werden als Wendepunkte höherer Ordnung bezeichnet (etwa für $k = 3$ als *kubische* Wendepunkte). Diese Wendepunkte, die auch durch die Bedingung

$$f_{red} = \frac{\partial}{\partial x} f_{red} = \frac{\partial^2}{\partial x^2} f_{red} = \cdots = \frac{\partial^{k-1}}{\partial x^{k-1}} f_{red} = 0 \qquad \frac{\partial^k}{\partial x^k} f_{red} \neq 0$$

charakterisiert werden können, sind aber nicht generisch, da die Bedingungen, daß auch höhere Ableitungen gleich Null sind, auch bei kleinen Störungen nicht mehr erfüllt werden können. Etwa zerfällt ein kubischer Wendepunkt je nach Störung entweder in ein Paar von quadratischen Wendepunkten oder verschwindet völlig; solche Wendepunkte sind für die numerische Behandlung unangenehm, da abhängig von den Details des Algorithmus entweder zwei Wendepunkte oder keine gefunden werden.

Berechnung

Von den zwei Schritten zur Berechnung eines Wendepunktes ist Schritt 1 sehr einfach. Nämlich liegt zwischen y_ν und $y_{\nu+1}$ sicher mindestens ein Wendepunkt, wenn

$$t(y_\nu)_\lambda * t(y_{\nu+1})_\lambda \leq 0 \tag{2.3}$$

d.h. die letzte Komponente der Tangente des Lösungsvektors das Vorzeichen ändert.

Ausgehend von diesen beiden Punkten läßt sich nach [9] in einer Art Intervallschachtelungsalgorithmus der Wendepunkt berechnen:

1. Setze $y^0 := y_\nu, t^0 := t(y_\nu) \quad y^1 := y_{\nu+1}, t^1 := t(y_{\nu+1})$
2. Berechne den Wendepunkt $\hat{y}$ des kubischen Hermite-Polynoms h, welches durch y^0, t^0 und y^1, t^1 eindeutig bestimmt wird. Dazu muß nach Aufstellen des Polynoms nur die quadratische Gleichung $h'_\lambda = 0$ gelöst werden, was wegen $h'_\lambda(y^0) * h'_\lambda(y^1) = {t^0}_\lambda * {t^1}_\lambda \leq 0$ sicher immer möglich ist.
3. Ausgehend von diesem Punkt $\hat{y}$ berechne mit dem Gauss-Newton-Verfahren einen Lösungspunkt $f(\bar{y}) = 0$.
4. Falls nun für die Tangente $\bar{t} = t(\bar{y})$ die Bedingung $|\bar{t}_\lambda| < \varepsilon$ mit ε als vorgeschriebener Toleranz erfüllt ist, akzeptiere $\bar{y}$ als Wendepunkt; sonst setze $y_0 := \bar{y}$, $t^0 := \bar{t}$, falls $\bar{t}_\lambda * {t^1}_\lambda \leq 0$, oder $y^1 := \bar{y}$, $t^1 := \bar{t}$ anderenfalls, und wiederhole ab Schritt 2

Das Verfahren wird abgebrochen, wenn in Schritt 2 der Wendepunkt der Hermite-Interpolierenden nicht zwischen den Schachtelungspunkten y^0, y^1 auf der Interpolierenden liegt, das Gauß-Newton-Verfahren in Schritt 3 nicht konvergiert, oder nach einer bestimmten Anzahl von Schachtelungen die gewünschte Genauigkeit noch nicht erreicht ist. In diesem Fall wird der Lösungspunkt $y_{\nu+1}$ verworfen und der Pfadverfolgungsalgorithmus mit einer verkleinerten Schrittweite in y_ν neu gestartet.

2.4 Einfache Verzweigungen

Die einfachsten Singularitäten aus der y-Perspektive lassen sich aus den Bedingungen

$$\operatorname{rank}(f'(y)) = n - 1 \qquad \text{und} \qquad f(y)'' \quad \text{auf} \quad \mathcal{N}(f'(y)) \quad \text{regulär} \tag{2.4}$$

erhalten; für die reduzierte Funktion gilt dann

$$f_{red}(y) = \underbrace{f_{red}(0) + f'_{red}(0)}_{0} y + \frac{1}{2} f''_{red}(0)(y, y) + \cdots$$

Dabei ist $f''_{red}(0)$ als quadratische Form diagonalisierbar, also lauten die möglichen Normalform mit $y = (x, \lambda)$:

$$\begin{pmatrix} x & \lambda \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \pm 1 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = x^2 \pm \lambda^2 = 0$$

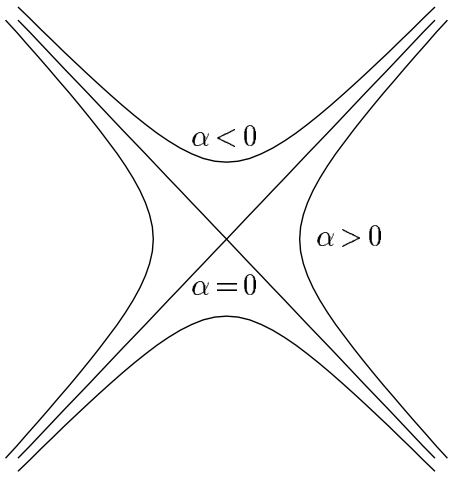


Abbildung 2.3: Einfache Verzweigung mit unvollständigen Lösungen

Dabei besitzt $x^2 + \lambda^2 = 0$ nur eine isolierte Lösung $(0,0)$, kommt also als Normalform für eine Singularität entlang eines Pfades nicht in Frage; folglich kommt zu den Bedingungen 2.4 noch die Bedingung

$$\mathcal{P}_{\mathcal{R}}{}^{\perp} f(y^\star)'' \quad \text{auf } \mathcal{N}(f') \times \mathcal{N}(f') \text{ indefinit} \tag{2.5}$$

für eine Singularität. Dabei rühren sowohl der orthogonale Projektor $\mathcal{P}_{\mathcal{R}}{}^{\perp}$ auf $\mathcal{R}(f'(y^\star))^{\perp} = \mathcal{R}^{\perp}$ ebenso wie die Einschränkung auf $\mathcal{N}(f'(y^\star))$ von der Ljapunov-Schmidt-Reduktion her.

Singularitäten, die die Bedingungen 2.4 und 2.5 erfüllen, werden als *einfache Verzweigungen* (engl. simple bifurcations) bezeichnet; ihre Normalform ist

$$\varphi(x, \lambda) = x^2 - \lambda^2 = 0,$$

so daß sich im Ursprung $(0,0)$ die Lösungspfade $x = \lambda$ und $x = -\lambda$ kreuzen.

Bemerkung 2.6: Im Gegensatz zu Wendepunkten bleiben Verzweigungen unter kleinen Störungen nicht erhalten, wie schon an der Normalform erkannt werden kann. Sei nämlich α eine solche kleine Störung, dann besteht das Verzweigungsdiagramm von

$$\tilde{\varphi}(x, \lambda) = x^2 - \lambda^2 + \alpha = 0$$

aus zwei Hyperbeln, die je nach Vorzeichen von α über x oder λ parametrisiert werden können. Der Punkt $(0,0)$ bleibt aber weiterhin durch die Eigenschaft $\tilde{\varphi}' = 0$ ausgezeichnet. Für Störungen des ursprünglichen System ist es wichtig, daß nur die Komponente der Störung, die in $\mathcal{R}(f'(y^\star))^\perp$ liegt, die Verzweigung zerstört; Komponenten in $\mathcal{R}(f'(y^\star))$ werden während der Ljapunov-Schmidt-Reduktion wegprojiziert. Dies läßt sich auch an der Normalform erkennen, wenn sie um eine weitere Variable y und eine weitere Gleichung $y - \beta = 0$ aufgeblasen wird. Die Ableitung lautet dann

$$\varphi'(x,y,\lambda) = \begin{pmatrix} 2x & 0 & -2\lambda \\ 0 & 1 & 0 \end{pmatrix}$$

so daß von dem Störungsvektor $\binom{\alpha}{\beta}$ nur $\alpha \in \mathcal{R}(\varphi'(0))^\perp$ die Verzweigung zerstört, während $\beta \in \mathcal{R}(\varphi'(0))$ die Verzweigung nur nach $(0,\beta,0)$ verschiebt.

Da bei der numerischen Auswertung von Funktionen kleine Störungen durch Rundungsfehler unvermeidlich sind, muß eine Methode gefunden werden, die auch solche gestörten oder *unvollständigen* Verzweigungen (engl. unfolded, imperfect bifurcations) behandeln kann.

Bemerkung 2.7: In der Literatur findet sich oft eine anderere Definition, die einfache Verzweigungen aus der λ-Perspektive charakterisiert. Dabei wird eine Singularität einfache Verzweigung genannt, wenn

$$\operatorname{rank}\left(\frac{\partial}{\partial x} f(y)\right) = n-1 \qquad \text{und} \qquad \frac{\partial}{\partial \lambda} f(y) \in \mathcal{R}(\frac{\partial}{\partial x} f(y)) \tag{2.6}$$

sowie Nichtdegeneriertheitsanforderungen an höhere Ableitungen erfüllt sind. Diese Definition führt zu den beiden Normalformen

$$f_{red}(x,\lambda) = \frac{1}{2}\frac{\partial^2}{\partial x^2} f_{red}(0,0)x^2 - \frac{1}{2}\frac{\partial^2}{\partial \lambda^2} f_{red}(0,0)\lambda^2 + \cdots \Rightarrow \phi(x,\lambda) = x^2 - \lambda^2$$

welche als *transkritische Verzweigung* bezeichnet wird, und

$$f_{red}(x,\lambda) = \frac{1}{6}\frac{\partial^3}{\partial x^3} f_{red}(0,0)x^3 - \frac{\partial}{\partial x}\frac{\partial}{\partial \lambda} f_{red}(0,0)x\lambda + \cdots \Rightarrow \phi(x,\lambda) = x^3 - \lambda x \tag{2.7}$$

was eine *Pitchfork-(bzw. Heugabel-)Verzweigung* charakterisiert. Beide Normalformen sind nicht äquivalent, wie aus Bemerkung 2.3 folgt. Aus ihren

Verzweigungsdiagrammen ist zu ersehen, daß sie jeweils verschieden viele Lösungen $x = x(\lambda)$ besitzen, die transkritische je zwei für $\lambda < 0$ und $\lambda > 0$, die Pitchfork eine für $\lambda < 0$ und drei für $\lambda > 0$.

Andere Normalformen, die aus anderen Taylorentwicklungen stammen, sind zu einer der beiden äquivalent.

Diese Charakterisierung von einfachen Verzweigungen ist aber schwächer als die Charakterisierung über (2.4) und (2.5), da der Rangdefekt in $f'(y)$ auch anders erreicht werden kann als in (2.6), nämlich durch

$$\operatorname{rank}\left(\frac{\partial}{\partial x}f(y)\right) = n-2 \qquad \text{und} \qquad \frac{\partial}{\partial \lambda}f(y) \notin \mathcal{R}(\frac{\partial}{\partial x}f(y)) \tag{2.8}$$

Dies führt nach einer Ljapunov-Schmidt-Reduktion aus der λ-Perspektive wie in Kapitel 2.2 zu einer reduzierten Abbildung, die jetzt dem höheren Rangdefekt entspechend von zwei Variablen zusätzlich zum Parameter abhängt:

$$f_{red}(x,y,\lambda) = 0 \qquad \text{mit} \qquad \frac{\partial}{\partial(x,y)}f_{red} = 0 \quad \text{und} \quad \frac{\partial}{\partial\lambda}f_{red} \neq 0$$

Eine mögliche Normalform dieser *nichtklassischen Verzweigung* läßt sich zu

$$\varphi(x,y,\lambda) = \begin{pmatrix} \varphi_1(x,y,\lambda) \\ \varphi_2(x,y,\lambda) \end{pmatrix} = \begin{pmatrix} (x^2-\lambda)+(y^2-\lambda) \\ xy \end{pmatrix} = 0$$

angeben. Das Verzweigungsdiagramm besteht aus zwei Parabeln, einer in der (x,λ)-Ebene und einer in der (y,λ)-Ebene, die sich im Ursprung kreuzen.

Da $\varphi'(0,0,0) = \left(\begin{smallmatrix} 0 & 0 & -2 \\ 0 & 0 & 0 \end{smallmatrix}\right)$, läßt sich in der y-Perspektive eine weitere Reduktion durchführen; nämlich ist $\lambda = \lambda(x,y) = \frac{1}{2}(x^2+y^2)$, die Normalform reduziert sich dadurch zur Normalform $xy = 0$, welche zu $\bar{x}^2 - \bar{y}^2$ äquivalent ist. (z.B. $x = \bar{x} - \bar{y}$, $y = \bar{x} + \bar{y}$).

Zur Stabilitätsanalyse läßt sich entlang der Linien von Bemerkung 2.2 sagen, daß an transkritischen Verzweigungen ein Stabilitätsaustausch stattfindet; ist für $\lambda < 0$ die eine Lösung stabil, dann ist für $\lambda > 0$ die andere stabil (vorausgesetzt, sie hatte nur eine eindimensionale instabile Mannigfaltigkeit). Bei einer Heugabel verliert ein vom „Gabelgriff“ her kommender stabiler Pfad seine Stabilität an beide äußeren „Gabelzinken“; ist er hingegen instabil, so sind die beiden äußeren Zinken dann ebenfalls instabil, während

die mittlere Zinke möglicherweise eine stabile Lösung darstellt. Bei der nichtklassischen Verzweigung läßt sich wenig mehr sagen, als daß entlang jedem Pfad zwei Eigenwerte das Vorzeichen wechseln.

Alle aus der λ-Perspektive verschiedenen Typen von einfachen Verzweigungen sind aus der y-Perspektive gleich und können auf dieselbe Weise berechnet werden.

Berechnung

Zunächst stellt sich wieder die Frage, wie entdeckt werden soll, ob zwischen zwei berechneten Lösungspunkten eine Verzweigung liegt. Die Antwort ergibt sich aus einer Verallgemeinerung der Tatsache, daß entlang eines Pfades, der sich über λ parametrisieren läßt, die Determinante von $\frac{\partial}{\partial x} f$ ihr Vorzeichen an einer einfachen Verzweigung wechselt. Um den Problemen der λ-Perspektive zu entrinnen, wird zunächst die Determinante der $(n, n+1)$-Matrix $f'(y)$ definiert.

Zur Formulierung dieser Definition bezeichne A_i die i-te Spalte einer Matrix A. Mit A^i wird hingegen die Matrix bezeichnet, die durch Streichen der i-ten Spalte entsteht:

$$A^i = (A_1, A_2, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n)$$

Dann folgt nun in Definition + Satz:

Satz 2.1 *Sei für eine beliebige $(n, n+1)$ Matrix A mit (normiertem) Tangentialvektor t die tangentiale Determinante als*

$$det_t\,(A) := (-1)^i \frac{det\,(A^i)}{t_i} \qquad \text{für } i \text{ bel. mit } t_i \neq 0 \tag{2.9}$$

definiert. Dann gilt

a) Die Definition ist von i unabhängig.

b) $rank(A) = n \iff det_t\,(A) \neq 0$.

c) Für jede (n, n) Matrix B gilt

$$det_t\,(BA) = det\,(B)\, det_t\,(A)$$

d) Für jede $(n+1, n+1)$-Matrix B gilt

$$\|B\big|_{B^{-1}(\mathcal{N}(A))}\|\, det_t\,(AB) = det_t\,(A)\, det\,(B)$$

Der **Beweis** ist eher technisch und in den Anhang verlegt. Wichtiger ist das Korollar:

Korollar 2.2 *Die tangentiale Determinante von $f'(y)$ ist entlang eines glatten Pfades $f(y) = 0$ eine stetige Funktion. Insbesondere wechselt sie nicht das Vorzeichen, wenn $f'(y)$ vollen Rang hat, d.h. entlang eines Pfades regulärer Punkte.*

Denn Vorraussetzung für die Existenz eines Pfades ist die stetige Differenzierbarkeit der Funktion f. Damit ist aber $f'(y)$ stetig; da der Pfad glatt sein soll, ist auch der Tangentialvektor stetig. Sowohl die Determinante einer Teilmatrix als auch die Koordinate eines Tangentialvektors hängen stetig von der Matrix bzw. dem Vektor ab. Da zumindest lokal stets ein Index i gefunden werden kann mit $t_i \neq 0$ in einer ganzen Umgebung des Punktes auf dem Pfad, ist der Quotient $\det_t(A) = \det(A)_i / t_i$ stetig.

Bemerkung 2.8: Eine alternative Definition, die in Anbetracht von Gleichung (1.11) naheliegt, besteht darin, die Determinante als

$$\det_t(A) := \det \begin{pmatrix} A \\ t^* \end{pmatrix} \tag{2.10}$$

zu definieren. Wie im Anhang gezeigt wird (siehe Gleichung A.5), ist diese Definition nur um einen Faktor $(-1)^{n-1}$ verschieden von der obigen Definition (2.9). Folgend der Beobachtung, daß für $A = (A_x, A_\lambda)$

$$\det(A_x) = \det \begin{pmatrix} A_x & A_\lambda \\ 0 & 1 \end{pmatrix} = \det \begin{pmatrix} A \\ e_n \end{pmatrix}$$

läßt sich diese Definition als Determinante von A eingeschränkt auf das orthogonale Komplement zum Tangentialraum interpretieren. Dies motiviert die Bezeichnung „tangentiale Determinante“. Die Definition aus (2.10) läßt sich auch einfach auf andere rechteckige Matrizen übertragen. Für die Berechnung mit Hilfe von Eliminationsverfahren ist dieser Zugang jedoch weniger gut geeignet als die Definition (2.9) in Satz 2.1.

Bemerkung 2.9: Für eine allgemeine Matrix A sind beide Determinanten im Fall von vollem Rang nur bis auf das Vorzeichen definiert, das von der Orientierung der Tangente abhängt. In Pfadverfolgungskontext bereitet

diese Mehrdeutigkeit aber keine Probleme, da für die Tangente entlang eines Pfades immer eine eindeutige Orientierung vorhanden ist. Daher wird im weiteren der Index t fallengelassen und einfach $\det(A)$ für $\det_t(A)$ geschrieben.

Mit Hilfe der Reduktionstechniken aus Abschnitt 2.2 läßt sich nun leicht beweisen:

Satz 2.3 *Die tangentiale Determinante von $f'(y)$ wechselt entlang eines Pfades an einer einfachen Verzweigung das Vorzeichen.*

Beweis: Zunächst zeigt die Ljapunov-Schmidt-Reduktion wegen

$$\frac{d}{d\tau}f_{red} = \frac{\partial}{\partial\xi}f_2\frac{d\xi}{d\tau} + \frac{\partial}{\partial\tau}f_2 \quad \text{und} \quad \frac{d\xi}{d\tau} = -\frac{\partial}{\partial\xi}f_1^{-1}\frac{\partial}{\partial\tau}f_1$$

sowie

$$\det(f') = \det\begin{pmatrix} \frac{\partial}{\partial\xi}f_1 & \frac{\partial}{\partial\tau}f_1 \\ \frac{\partial}{\partial\xi}f_2 & \frac{\partial}{\partial\tau}f_2 \end{pmatrix} = \det\left(\frac{\partial}{\partial\xi}f_1\right)\det\left(\frac{d}{d\tau}f_{red}\right)$$

daß die Determinante von f' nur zusammen mit der von f'_{red} das Vorzeichen wechselt, da $\frac{\partial}{\partial\xi}f_1$ nach Konstruktion am Verzweigungspunkt und aus Stetigkeitsgründen auch in einer ganzen Umgebung regulär ist.

Sodann folgt aus der Äquivalenz zur Normalform

$$\varphi(y) = y^*\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}y = S(y)f_{red}(\Psi(y)),$$

daß

$$\det(\varphi'(y)) = \det\left(\frac{d}{dy}(S(y)f_{red}(\Psi(y)))\right) = S(y)\det(f'_{red}(\Psi(y)))\det(\Psi'(y))/c$$

und wegen der Bedingungen $S(y) \neq 0$ und $\det(\Psi'(y)) \neq 0$ sowie $c := \|\Psi'^{-1}|_{\Psi'(\mathcal{N}(f'_{red}))}\| \neq 0$ ändert die Determinante von f'_{red} genau dann ihr Vorzeichen, wenn es auch die der Normalform φ' tut.

Für die Normalform $\varphi(x,\lambda) = x^2 - \lambda^2$ ist dies aber leicht einzusehen. Sei etwa die Determinante entlang dem Pfad $x = \lambda$ betrachtet; die Ableitung und der zugehörige Tangentialvektor lauten

$$\varphi'(x,\lambda) = \begin{pmatrix} 2x & -2\lambda \end{pmatrix} \qquad t = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Damit wechselt die Determinante $\det(\varphi') = -\sqrt{2}\lambda$ beim Übergang von $\lambda < 0$ zu $\lambda > 0$ das Vorzeichen. Für den anderen Ast $x = -\lambda$ gilt dasselbe, wie sich ebenso nachrechnet. □

Nachdem durch Vorzeichenwechsel der Determinante festgestellt wurde, daß eine Verzweigung vorhanden ist, besteht nun das Problem darin, diesen Verzweigungspunkt zu berechnen und geeignete Startpunkte für den abzweigenden Lösungsast zu finden.

Das Problem, daß kleine Störungen eine einfache Verzweigung zerstören, läßt sich nach der Methode, die von G.Moore in [29] vorgeschlagen wurde, beheben, indem die Störung als zusätzliche Variable eingeführt wird. Vereinfacht wird dies dadurch, daß wegen Bemerkung 2.6 nur Störungen in $\mathcal{R}(f'(y))^{\perp}$ berücksichtigt werden müssen. Gesucht wird eine Verzweigung von $f(y)-\alpha z = 0$, wobei die zusätzliche Variable $z \in \mathbb{R}^n$ den Raum $\mathcal{R}(f'(y))^{\perp}$ aufspannt und α die Störung darstellt. Damit liest sich das Mooresche System zur Berechnung von einfachen Verzweigungen:

$$f'(y)^* z = 0 \tag{2.11}$$

$$f(y) - \alpha z = 0 \tag{2.12}$$

$$\tfrac{1}{2}(\|z\|^2 - 1) = 0 \tag{2.13}$$

Gleichung (2.11) stellt sicher, daß $z \in \mathcal{R}(f'(y))^{\perp}$; nebenbei wird auch der erste Teil von Bedingung (2.4) erfüllt. Gleichung (2.12) bedeutet, daß y eine Lösung der um α gestörten Gleichung ist. Schließlich wird Gleichung (2.13) gebraucht, um $z \neq 0$ zu garantieren; der Faktor $\frac{1}{2}$ bewirkt, daß die Ableitung des Systems symmetrisch ist.

Das erweiterte System besteht aus $2n+2$ Gleichungen für $2n+2$ Variablen; daß die Ableitung in der Nähe von einfachen Verzweigungspunkten regulär ist, wird in Kapitel 3.4 gezeigt. Daher besitzt das System eine lokal eindeutige Lösung, welche der Verzweigungspunkt ist, und die mit einem gewöhnlichen Newton-Verfahren berechnet werden kann.

Gute Startwerte für die Newton-Iteration lassen dann finden, wenn an den Lösungspunkten y_ν und $y_{\nu+1}$ Schätzer für die Konditionszahl der Ableitung $\kappa(f')$ existieren, die die asymptotische Relation $\kappa(f') \sim 1/s$ erfüllen, wobei s den Abstand zur Verzweigung bezeichnet. Aus linearer Interpolation der Nullstelle von $1/\kappa(f')$ auf der Geraden durch y_ν und $y_{\nu+1}$ läßt sich der

Startwert

$$y^0 := \frac{1}{\kappa(f'(y_\nu)) + \kappa(f'(y_{\nu+1}))}\Big(\kappa(f'(y_\nu))y_\nu + \kappa(f'(y_{\nu+1}))y_{\nu+1}\Big)$$

finden; ist kein Konditionsschätzer zur Hand, tut es auch $y^0 := \frac{1}{2}(y_\nu + y_{\nu+1})$. Für z^0 wird jedes der in Kapitel 3 diskutierten Verfahren der linearen Algebra zur Berechnung der Pseudoinversen eine eigene Lösung vorschlagen, die jeweils einen linken Nullvektor approximiert. Sind y^0 und z^0 gegeben, läßt sich ein Startwert für α daraus motivieren, daß Gleichung (2.12) zumindest in Richtung von z^0 erfüllt sein soll, daraus ergibt sich:

$$\langle z^0, f(y^0) - \alpha^0 z^0\rangle = 0 \Rightarrow \alpha^0 := z^{0*} f(y^0)$$

Falls das Newton-Verfahren nicht konvergiert, können ähnlich wie bei Wendepunkten mit y^0 anstelle von y_ν bzw. $y_{\nu+1}$ neue Startwerte interpoliert werden; bringen mehrere Interpolationen keine Konvergenz, muß die Lösung $y_{\nu+1}$ verworfen und mit reduzierter Schrittweite die Pfadverfolgung bei y_ν neu gestartet werden.

Ist nun der Verzweigungspunkt $y^\star$ gefunden, stellt sich das Problem, wie Startwerte auf dem Pfad gefunden werden können, der von dem berechneten Pfad abzweigt. Aus der Normalform ist ersichtlich, daß die Tangente t eines abzweigenden Pfades außer $t \in \mathcal{N}(f'(y^\star))$ auch die Gleichung

$$\mathcal{P}_{\mathcal{R}}{}^{\perp} f''(y^\star)(t,t) = 0$$

erfüllen muß; in der Normalform ist nämlich für die Tangenten an die abzweigenden Pfade $t_{1/2}^* C t_{1/2} = (1\,\pm 1)\left(\begin{smallmatrix}1 & 0\\ 0 & -1\end{smallmatrix}\right)\left(\begin{smallmatrix}1\\ \pm 1\end{smallmatrix}\right) = 1^2 - (\pm 1)^2 = 0$.

Da die Auswertung von $f''(y)$ für das Newton-Verfahren ohnehin erforderlich ist und bei der Lösung des entstehenden linearen Systems die $(2,2)$-Matrix, die die zweite Ableitung auf $\mathcal{N}(f'(y))$, projiziert auf $\mathcal{R}(f'(y^\star))^\perp$ darstellt, durch eine entsprechende Koordinatentransformation explizit berechnet wird, bedeutet die Berechnung der Tangenten nur die Berechnung beider Lösungen einer quadratischen Gleichung $t^* C t = c_{11}t_1^2 + 2c_{12}t_1t_2 + c_{22}t_2^2 = 0$ mit einer indefiniten $(2,2)$-Matrix C. Die Lösung läßt sich z.B. für $c_{22} \neq 0$ durch $t^T = (\cos\theta, \sin\theta)$ mit $\tan\theta = \Big(-c_{12} \pm \sqrt{c_{12}^2 - c_{11}c_{22}}\Big)\Big/ c_{22}$ berechnen.

Nachdem die Koordinatentransformation rückgängig gemacht worden ist, ergeben sich zwei Tangentialrichtungen. Diejenige, die mit der Tangente der

Punktes y_ν den kleineren Winkel einschließt, wird als die Tangente des aktuellen Pfades genommen; die andere Tangente erlaubt dann, tangentiale Prediktoren für Lösungen auf dem abzweigenden Pfad $y^\star \pm st$ zu berechnen.

Die Schrittweite s muß dabei klein genug sein, daß der Korrektor konvergiert, aber groß genug, daß der Prediktor hinreichend weit von der Verzweigung entfernt ist und die Ableitung numerisch nicht mehr rangdefekt ist. Dazu wird als erster Versuch $s = \rho \max(\|y_\nu - y^\star\|, \|y_{\nu+1} - y^\star\|)$ gewählt mit einem Sicherheitsfaktor ρ, etwa $\rho = \frac{1}{2}$. Ist diese Schrittweite zu groß, findet die a-posterori-Schrittweitenstrategie aus Abschnitt 1.2.5 Anwendung; ist die Schrittweite zu klein, wird sie um einen konstanten Faktor, etwa 2, vergrößert.

Damit sind alle Probleme, die bei der numerischen Behandlung einfacher Verzweigungen im Kontext der Pfadverfolgung auftreten können, diskutiert worden. Falls jedoch die Bedingungen (2.4) und (2.5) nicht erfüllt sind, liegen höhere Verzweigungen vor. Diese benötigen in der Regel Informationen aus höheren Ableitungen. Falls der Algorithmus auf eine solche Verzweigung trifft, ist es möglich, daß er sie erst gar nicht entdeckt, da die Determinante ihr Vorzeichen nicht wechselt. Falls die Verzweigung dennoch entdeckt wird, wird das Mooresche System entweder nicht konvergieren oder spätestens bei der Berechnung der Tangenten scheitern, da f'' auf $\mathcal{N}(f')$ nicht indefinit sein wird.

Normalerweise sind schon einfache Verzweigungen kein generisches Phänomen, höhere Verzweigungen noch weniger. Falls jedoch Symmetrien vorhanden sind, unter denen f invariant ist, dann können von einer Lösung, die unter der Symmetrie invariant ist, an einer *symmetriebrechenden Verzweigung* (unter Unständen sehr viele) Lösungen abzweigen, die jeweils zueinander symmetrisch sind. Solche symmetriebrechenden Verzweigungen werden durch die Anwesenheit der Symmetrie zu generischen Erscheinungen. Eine Ausnutzung der Symmetrie zur Berechnung dieser Verzweigungen ist mit dem symbolisch/numerischen Programmpaket `Symcon` von Gatermann und Hohmann [17] möglich.

Tatsächlich besitzen alle der im Kapitel 5 vorgestellten Beispiele eine solche Symmetrie. Dabei handelt es sich aber nur um eine einfache Spiegelungssymmetrie. Die dort auftretenden Verzweigungen können noch berechnet werden, da ihre Normalform die der Pitchfork-Verzweigung (2.7) ist, sie also einfache Verzweigungen bilden. Die Spiegelsymmetrie entspricht dabei

der Symmetrie $x \leftrightarrow -x$, unter der der Pfad $x = 0$ invariant ist, während die beiden Zweige $x = \pm\sqrt{\lambda}$ zueinander symmetrisch sind und den abzweigenden Pfad bilden.

3. Berechnung der Pseudoinversen

3.1 Definition der Moore-Penrose-Pseudoinversen

Im Rahmen der Pfadverfolgung treten bei dem Gauß-Newton-Verfahren lineare Gleichungssysteme (1.6) auf, die keine eindeutige Lösung besitzen;. Um dennoch zu einem eindeutigen Verfahren zu gelangen, wurde das Konzept der Moore-Penrose-Pseudoinversen angewandt. Dieses Konzept wird in diesem Abschnitt in einem allgemeineren Rahmen erläutert. In den folgenen Abschnitten dieses Kapitels werden dann aus einem generellen Schema zur Berechnung der Pseudoinversen, angewandt auf verschiedene Zerlegungsverfahren der linearen Algebra, Algorithmen zur Berechnung der Pseudoinversen hergeleitet. Schließlich wird mittels des generellen Schemas eine effiziente Methode zur Lösung der im Newton-Verfahren des Mooreschen Systems (2.11) – (2.13) auftretenden Gleichungssysteme dargestellt.

Das Ziel bei der Definition der Moore-Penrose-Pseudoinversen besteht darin, jedem Gleichungsystem

$$Ax = b$$

eine Art „Lösung“ zuzuordnen, für jedes mögliche A und b.

Ein Lösungsansatz ergibt sich aus dem wohlbekannten Isomorphiesatz der Algebra, der im Fall von linearen Operatoren zwischen zwei Vektorräumen besagt:

Satz 3.1 (Isomorphiesatz) *Für jeden linearen Operator $A\colon V \to W$ zwischen zwei Vektorräumen V und W existiert ein Isomorphismus $\bar{A}\colon V/\mathcal{N}(A) \to \mathcal{R}(A)$, so daß das Diagramm*

$$\begin{tikzcd} V \arrow[r, "A"] \arrow[d, "\pi_{\mathcal{N}(A)}"'] & W \\ V/\mathcal{N}(A) \arrow[r, "\bar{A}"'] & \mathcal{R}(A) \arrow[u, "Id_{\mathcal{R}(A)}"'] \end{tikzcd}$$

kommutativ ist, wobei $\pi_{\mathcal{N}(A)}(x) = x + \mathcal{N}(A)$ die kanonische Projektion bezeichnet.

Unter der Berücksichtigung des in Hilberträumen kanonischen Isomorphismus $V/\mathcal{N}(A) \cong \mathcal{N}(A)^\perp = \mathcal{R}(A^*)$ ist also die Einschränkung von A auf $\mathcal{R}(A^*)$ ein Isomorphismus $\bar{A} : \mathcal{R}(A^*) \to \mathcal{R}(A)$; diese Relation läßt sich auch als $A = \mathcal{P}_{\mathcal{R}(A)}\bar{A}\mathcal{P}_{\mathcal{R}(A^*)}$ schreiben. Da für $\bar{A}$ eine Inverse existiert, ist es naheliegend, die Pseudoinverse als

$$A^+ := \mathcal{P}_{\mathcal{R}(A^*)}\bar{A}^{-1}\mathcal{P}_{\mathcal{R}(A)} \tag{3.1}$$

zu definieren.

A^+ ist also derjenige Operator, der auf $\mathcal{R}(A)$ eine Inverse für den auf $\mathcal{R}(A^*)$ eingeschränkten und dadurch bijektiven Operator A darstellt und auf $\mathcal{R}(A)^\perp = \mathcal{N}(A^*)$ dem Nulloperator entspricht, also $\mathcal{R}(A^+) = \mathcal{R}(A^*)$ und $\mathcal{N}(A^+) = \mathcal{N}(A^*)$. Dies ist die sogenannte „Operatordefinition“ der Pseudoinversen, die von Hestenes in [22] gegeben wurde.

Bemerkung 3.1: Aufgrund der Dualität zwischen A und A^* in der Definition ist die Pseudoinverse von A^* einfach $A^{*+} = A^{+*} = \mathcal{P}_{\mathcal{R}(A)}\bar{A}^{-*}\mathcal{P}_{\mathcal{R}(A^*)}$.

Neben der Operatordefinition existieren noch eine Reihe äquivalenter Definitionen. Die ursprüngliche Definition von E.H.Moore in [28] besagt, daß die Pseudoinverse einer Matrix A diejenige Matrix ist, die auf $\mathcal{R}(A)$ eine Rechtsinverse von A ist, bzw. $AA^+A = A$ und deren Spalten sowie Zeilen Linearkombinationen der Spalten bzw. Zeilen der Adjungierten A^* sind, also $\mathcal{R}(A^+) \subseteq \mathcal{R}(A^*)$ und $\mathcal{N}(A^+) \supseteq \mathcal{N}(A^*)$. Da aus $AA^+A = A$ folgt, daß $\operatorname{rank}(A^+) = \operatorname{rank}(A) = \operatorname{rank}(A^*)$, sind die Inklusionen tatsächlich Gleichheiten und die Definition zur Operatordefinition äquivalent.

Penrose hat unabhängig von Moore in [31] eine Definition über die nach ihm benannten Penrose-Axiome formuliert, welche lauten:

$$A^+AA^+ = A^+ \quad AA^+A = A(A^+A)^* = A^+A \quad (AA^+)^* = AA^+ \tag{3.2}$$

Durch Einsetzen der Definition (3.1) in (3.2) läßt sich zeigen, daß der Operator A^+ diese Axiome erfüllt. Andererseits besagen die ersten beiden Axiome, daß A^+ eine Links- bzw. Rechtsinverse von A auf $\mathcal{R}(A^+)$ bzw. $\mathcal{R}(A)$ darstellt, während die anderen beiden Axiome besagen, daß A^+A bzw. AA^+ orthogonale Projektoren auf $\mathcal{R}(A^+)$ bzw. $\mathcal{R}(A)$ sind, woraus $\mathcal{N}(A) = \mathcal{N}(A^+A) = \mathcal{R}(A^+A)^\perp = \mathcal{R}(A^+)^\perp$ und $\mathcal{N}(A^+) = \mathcal{N}(AA^+) = \mathcal{R}(AA^+)^\perp = \mathcal{R}(A)^\perp$ folgt.

Eine weitere Definition ergibt sich mit Hilfe der aus der Approximationstheorie bekannten Tatsache, daß für ein $b \in \mathcal{R}(A)$ dasjenige x, das in der

Lösungsmenge $x_0 + \mathcal{N}(A)$ (mit einer speziellen Lösung $Ax_0 = b$) liegt und in der Norm minimal ist, (also den Nullpunkt möglichst gut approximiert,) durch $x \in \mathcal{N}(A)^\perp = \mathcal{R}(A^*)$ gegeben ist. Ebenso stellt für ein b nicht notwendig in $\mathcal{R}(A)$ das Bild der Lösung $x = A^+b$ die Bestapproximation von b auf $\mathcal{R}(A)$ dar und minimiert daher das Residuum $b - Ax = b - \mathcal{P}_{\mathcal{R}(A)}b = \mathcal{P}_{\mathcal{R}(A)}{}^\perp b$. Also läßt sich die Pseudoinverse auch über die Minimierungsbedingungen

$$x = A^+b \iff \|Ax - b\| = \min \quad \text{und} \quad \|x\| = \min \tag{3.3}$$

definieren.

3.2 Herleitung des Lösungsverfahrens

Zentral für die weiteren Überlegungen ist der folgende Zerlegungssatz, der zu verschiedenen Möglichkeiten der Berechnung der Pseudoinversen führt. Sei für den Rest des Kapitels die Annahme gegeben, daß A eine (m,n)-Matrix mit $\operatorname{rank}(A) = p \leq \min(m,n)$ darstellt. Dann gilt der

Satz 3.2 (Zerlegungssatz) *Angenommen, es existiert eine Transformation des Urbildraumes $T_D\colon \mathbb{R}^p \times \mathbb{R}^{n-p} \to \mathbb{R}^n$ und eine Transformation des Bildraumes $T_R\colon \mathbb{R}^p \times \mathbb{R}^{m-p} \to \mathbb{R}^m$, so daß die Gleichung*

$$A = T_R \begin{pmatrix} \bar{A} & 0 \\ 0 & 0 \end{pmatrix} T_D{}^* \tag{3.4}$$

mit einer regulären (p,p)-Matrix $\bar{A}$ gilt. Sei des weiteren T_R partitioniert als:

$$T_R = \begin{pmatrix} \overbrace{W_1}^{p} & \overbrace{W_2}^{m-p} \end{pmatrix} \qquad \textit{und} \qquad T_R{}^{-1} = \begin{pmatrix} \bar{W}_1^* \\ \bar{W}_2^* \end{pmatrix} \begin{matrix} \}\, p \\ \}\, m-p \end{matrix}$$

und T_D partitioniert als:

$$T_D = \begin{pmatrix} \overbrace{V_1}^{p} & \overbrace{V_2}^{n-p} \end{pmatrix} \qquad \textit{und} \qquad T_D{}^{-1} = \begin{pmatrix} \bar{V}_1^* \\ \bar{V}_2^* \end{pmatrix} \begin{matrix} \}\, p \\ \}\, n-p \end{matrix}$$

Dann ist

$$\mathcal{R}(A) = \mathcal{R}(W_1) \quad \textit{und} \quad \mathcal{R}(A)^\perp = \mathcal{R}(\bar{W}_2)$$

sowie

$$\mathcal{R}(A^*) = \mathcal{R}(V_1) \quad \textit{und} \quad \mathcal{R}(A^*)^\perp = \mathcal{R}(\bar{V}_2)$$

Bemerkung 3.2: Die (p,p)-Matrix $\bar{A}$ stellt den Operator $\bar{A}$ in Satz 3.1 in speziellen Koordinaten, die von T_D und T_R abhängen, dar.

Beweis: Einsetzen der Partitionen von T_R und T_D in (3.4) ergibt $A = T_R \left(\begin{smallmatrix} \bar{A} & 0 \\ 0 & 0 \end{smallmatrix}\right) {T_D}^* = W_1 \bar{A} V_1^*$. Also ist $\mathcal{R}(A) = \mathcal{R}(W_1)$.

Wird die Gleichung ${T_R}^{-1} T_R = Id$ in Block-Form hingeschrieben:

$$ {T_R}^{-1} T_R = \begin{pmatrix} \bar{W}_1^* \\ \bar{W}_2^* \end{pmatrix} \begin{pmatrix} W_1 & W_2 \end{pmatrix} = \begin{pmatrix} \bar{W}_1^* W_1 & \bar{W}_1^* W_2 \\ \bar{W}_2^* W_1 & \bar{W}_2^* W_2 \end{pmatrix} = \begin{pmatrix} Id_p & 0 \\ 0 & Id_{m-p} \end{pmatrix} \tag{3.5} $$

folgt insbesondere $\bar{W}_2^* W_1 = 0$; also ist $\mathcal{R}(\bar{W}_2)$ orthogonal zu $\mathcal{R}(W_1)$. Da $\bar{W}_2$ eine Teilmatrix von T_R ist, hat sie vollen Rang. Daher gilt $\mathcal{R}(\bar{W}_2) = \mathcal{R}(A)^\perp = \mathcal{N}(A^*)$.

Aus der dualen Betrachtung, daß

$$ A^* = T_D \begin{pmatrix} \bar{A}^* & 0 \\ 0 & 0 \end{pmatrix} {T_R}^* = V_1 \bar{A}^* {W_1}^* $$

folgt $\mathcal{R}(V_1) = \mathcal{R}(A^*) = \mathcal{N}(A)^\perp$ und aus

$$ {T_D}^{-1} T_D = \begin{pmatrix} \bar{V}_1^* \\ \bar{V}_2^* \end{pmatrix} \begin{pmatrix} V_1 & V_2 \end{pmatrix} = \begin{pmatrix} \bar{V}_1^* V_1 & \bar{V}_1^* V_2 \\ \bar{V}_2^* V_1 & \bar{V}_2^* V_2 \end{pmatrix} = \begin{pmatrix} Id_p & 0 \\ 0 & Id_{m-p} \end{pmatrix} \tag{3.6} $$

folgt schließlich $\mathcal{R}(\bar{V}_2) = \mathcal{N}(A)$. □

Bemerkung 3.3: Unter der Berücksichtigung von Rundungsfehlern, wie sie bei der numerischen Berechnung einer Zerlegung unvermeidbar auftreten, ist es nicht erfüllbar, daß die Einträge, die die reguläre Matrix $\bar{A}$ in Gleichung (3.4) beranden, exakte Nullen sind. Statt dessen werden sich dort „hinreichend kleine“ Einträge befinden. Die Frage, wann diese Einträge hinreichend klein sind, führt zu dem diffizilen Problem des numerischen Rangentscheides, der bei verschiedenen Verfahren unterschiedlich ausfallen kann. Dabei kann der Unterschied dadurch entstehen, daß der Terminus „hinreichend klein“ in verschiedenen, den Verfahren angepaßten Fehlermaßen konkretisiert wird (siehe [13]).

Wird der Rang von A z.B. als das maximale p festgesetzt, für das $\mathrm{cond}(\bar{A}) < 1/\varepsilon$ ist, mit $\mathrm{cond}(\bar{A}) = \|\bar{A}\|\|\bar{A}^{-1}\|$ und einer vorgegebenen Genauigkeit ε, dann hängt die Konditionszahl und der Rangentscheid von der verwendeten Norm ab.

Im Kontext der Pfadverfolgung stellt sich dieses Problem zum Glück nicht, da der Rang aus analytischen Gründen festliegt. Während der Gauß-Newton-Iteration in der Pfadverfolgung wird von vollem Rang und bei der Berechung von einfachen Verzweigungen von einem einfachen Rangdefekt ausgegangen, solange die Zerlegung nicht einen höheren Rangdefekt zwingend nahelegt. Dies führt dazu, daß insbesondere bei der Berechnung von Verzweigungen aus analytischen Gründen eine Matrix als rangdefekt angenommen wird, obwohl sie dies numerisch nicht ist. Dabei stellt sich dann das Problem, wie diese Matrix zu einer möglichst nahegelegenen rangdefekten Matrix verändert werden kann.

Des weiteren wird angenommen, daß sich der als „hinreichend klein" angenommene Rest der Matrix A bei der Zerlegung $A = T_R \left(\begin{smallmatrix} \bar{A} & R^* \\ T & S \end{smallmatrix}\right) T_D{}^*$ auf die Matrix S konzentriert, so daß sich die Zerlegung inklusive der vernachlässigten Teile hinreichend genau als

$$A = T_R \begin{pmatrix} \bar{A} & 0 \\ 0 & S \end{pmatrix} T_D{}^* \tag{3.4'}$$

schreiben läßt. Dies ist für alle in Abschnitt 3.3 vorgeschlagenen Verfahren erfüllbar. Dabei braucht S, falls der Rang aus analytischen Gründen festgelegt wird, keinesfalls „vernachlässigbar klein" zu sein.

Mit Hilfe des Zerlegungssatzes läßt sich die Pseudoinverse $x = A^+ b$ in drei Teilschritten berechnen:

Schritt A: Projiziere b orthogonal auf $\mathcal{R}(A)$ und transformiere es mit $T_R{}^{-1}$:

$$\begin{pmatrix} c \\ 0 \end{pmatrix} = T_R{}^{-1} \mathcal{P}_{\mathcal{R}(A)} b$$

Schritt B: Löse das reguläre Gleichungssystem:

$$\bar{A} u = c$$

Schritt C: Transformiere $\left(\begin{smallmatrix} u \\ 0 \end{smallmatrix}\right) = \left(\begin{smallmatrix} \bar{A} & 0 \\ 0 & 0 \end{smallmatrix}\right)^+ \left(\begin{smallmatrix} c \\ 0 \end{smallmatrix}\right)$ mit $T_D{}^{-*}$ und projiziere das Ergebnis auf $\mathcal{R}(A^*)$:

$$x = \mathcal{P}_{\mathcal{R}(A^*)} T_D{}^{-*} \begin{pmatrix} u \\ 0 \end{pmatrix}$$

Die Projektoren in den Schritten A und C können nun mit dem folgenden Lemma aus den Informationen der zugehörigen Transformationen berechnet werden:

Lemma 3.3 *Sei V eine (n,p)-Matrix mit $rank(V) = p \leq n$. Dann ist der Projektor auf $\mathcal{R}(V)$ gegeben durch*

$$\mathcal{P}_V = V(V^*V)^{-1}V^* \tag{3.7}$$

Beweis: Da $\operatorname{rank}(V) = p$, gilt auch für die (p,p)-Matrix V^*V, daß $\operatorname{rank}(V^*V) = p$, also existiert die Inverse von V^*V und damit der Operator auf der rechten Seite von Gleichung 3.7. Des weiteren ist $\mathcal{R}(V^*) = Id_p$ und daher $\mathcal{R}(V(V^*V)^{-1}V^*) = \mathcal{R}(V(V^*V)^{-1}) = \mathcal{R}(V)$. Da der Operator die Gleichungen $P^2 = P$ und $P^* = P$ für einen orthogonalen Projektor erfüllt, und das Bild des Operators gleich $\mathcal{R}(V)$ ist, handelt es sich um den orthogonalen Projektor auf $\mathcal{R}(V)$. □

Bemerkung 3.4: Die Inverse der Matrix V^*V läßt sich zum einen dadurch berechnen, daß zuerst die symmetrische und, wegen des vollen Ranges von V, positiv definite Matrix V^*V explizit berechnet wird und dann eine Cholesky-Zerlegung $V^*V = R^*R$ berechnet wird. Da sich aber dabei die Rundungsfehler wie $\operatorname{cond}_2(V^*V) = \operatorname{cond}_2(V)^2$ verstärken, ist es vorzuziehen, eine Orthogonalisierung z.B. mittels des modifizierten Gram-Schmidt-Algorithmus $V = QR$ (siehe z.B. [19]) vorzunehmen, worauf hin sich die Inverse zu

$$(V^*V)^{-1} = (R^*Q^*QR)^{-1} = R^{-1}R^{-*}$$

berechnet. Paige hat in [30] gezeigt, daß sich hierbei ein Rundungsfehler ε nur gemäß $\operatorname{cond}_2(V)\,\varepsilon + \operatorname{cond}_2(V)^2\varepsilon^2$ bei der Auswerung von $(R^*R)^{-1}$ verstärkt – also unter der Bedingung $\operatorname{cond}_2(V)\,\varepsilon < 1$, die bedeutet, daß V numerisch vollen Rang hat, nicht stärker als bei der Auswertung von V.

Für die Berechnung der Projektors vereinfacht sich die Gleichung (3.7) durch Einsetzen der Orthogonalisierung noch zu

$$\mathcal{P}_V = QQ^*$$

wie auch aus $\mathcal{R}(V) = \mathcal{R}(Q)$ direkt ersichtlich ist.

Die Berechnung der Schritte A–C kann damit auf zwei verschiedene Weisen geschehen:

Methode $\mathbf{A}_1$: *(Normalengleichungen)* Zur Berechnung des Projektors auf $\mathcal{R}(A)$ läßt sich ausnutzen, daß $\mathcal{R}(A) = \mathcal{R}(W_1)$. Damit lautet der Projektor gemäß Lemma 3.3 $\mathcal{P}_A = W_1(W_1{}^*W_1)^{-1}W_1{}^*$ und

$$\begin{aligned}\begin{pmatrix} c \\ 0 \end{pmatrix} &= T_R{}^{-1}\mathcal{P}_{\mathcal{R}(A)}b &= \begin{pmatrix} \bar{W}_1^* \\ \bar{W}_2^* \end{pmatrix} W_1(W_1{}^*W_1)^{-1}W_1{}^*b \\ &&= \begin{pmatrix} (W_1{}^*W_1)^{-1}W_1{}^*b \\ 0 \end{pmatrix} \end{aligned} \tag{3.8}$$

da aus der Gleichung (3.5) $\bar{W}_1^*W_1 = Id$ und $\bar{W}_2^*W_1 = 0$ folgt.

Die Stabilität dieser Methode hängt in erster Linie von der Kondition der Matrix W_1 ab; der „Grad der Orthogonalität" der Spalten von W_1 kann als ein Maß für die Stabilität des Verfahrens betrachtet werden.

Der Aufwand zur Berechnung liegt vor allem in der Berechnung von $(W_1{}^*W_1)^{-1}$; der Aufwand dafür liegt bei ungefähr $1/2p^2m+1/6p^3$ flops für die Berechnung mit der Cholesky-Methode und bei p^2m flops bei der Berechnung über modifizierte Gram-Schmidt-Orthogonalisierung (ohne Berücksichtigung von sparsity-Effekten). Für den in diesem Kontext vorliegenden Fall eines nur geringen Rangdefektes $p \approx m$ liegt der Aufwand also bei etwa $2/3m^3$ bzw. m^3 flops. Der Aufwand für eine Berechnung $c = (W_1{}^*W_1)^{-1}W_1{}^*b$ liegt dann noch einmal bei etwa $2m^2$ flops.

Methode $\mathbf{A}_2$: *(Konormalengleichungen)* Wegen der Gleichung $\mathcal{R}(A)^\perp = \mathcal{R}(\bar{W}_2)$ läßt sich der orthogonale Projektor auf $\mathcal{R}(A)$ berechnen mittels:

$$\mathcal{P}_{\mathcal{R}(A)} = Id - \mathcal{P}_{\mathcal{R}(A)}{}^\perp = Id - \bar{W}_2(\bar{W}_2^*\bar{W}_2)^{-1}\bar{W}_2^*$$

und damit

$$\begin{aligned}\begin{pmatrix} c \\ 0 \end{pmatrix} &= T_R{}^{-1}\mathcal{P}_{\mathcal{R}(A)}b &= \begin{pmatrix} \bar{W}_1^* \\ \bar{W}_2^* \end{pmatrix} \left(Id - \bar{W}_2(\bar{W}_2^*\bar{W}_2)^{-1}\bar{W}_2^*\right) b \\ &&= \begin{pmatrix} \bar{W}_1^*b - \bar{W}_1^*\bar{W}_2(\bar{W}_2^*\bar{W}_2)^{-1}\bar{W}_2^*b \\ 0 \end{pmatrix} \end{aligned} \tag{3.9}$$

Die Stabilität dieser Methode hängt neben der Kondition von $\bar{W}_2$ vor allem von der Kondition der Subtraktion ab. Diese Subtraktion ist gut konditioniert, falls $\bar{W}_1^*b \approx c$ bzw. b beinahe $\in \mathcal{R}(\bar{W}_1^*)$. Da die Berechnung des Projektors nur für b beinahe $\in \mathcal{R}(A)$, d.h. für kleine Residuuen gut konditioniert ist, muß auch die Subtraktion nur in diesem Fall gut konditioniert sein,

damit das Verfahren stabil ist. Dies führt zu der zusätzlichen Stabilitätsbedingung $\mathcal{R}(\bar{W}_1^*) \approx \mathcal{R}(A)$. Dieses Argument trifft auch dann zu, wenn die Subtraktion vor der Multiplikation mit $\bar{W}_1^*$ ausgeführt wird, wie die Klammern in

$$c = \bar{W}_1^* \left(b - \bar{W}_2 \left(\bar{W}_2^* \bar{W}_2 \right)^{-1} \bar{W}_2^* b \right)$$

andeuten. Denn dann hängt die Stabitilät von der Kondition der Auswertung der Produktes $\bar{W}_1^* \mathcal{P}_{\mathcal{R}(A)} b$ ab, was wiederum zu der Bedingung $\mathcal{R}(\bar{W}_1^*) \approx \mathcal{R}(A)$ führt. Dieses Problem taucht bei Methode A_1 nicht auf.

Für die zusätzliche potentielle Quelle der Instabilität wird aber eine erhebliche Reduktion des Aufwandes eingetauscht. Der Aufwand zur Berechnung des Projektors reduziert sich zu $1/2(m-p)^2 m + 1/6(m-p)^3$ flops für die Cholesky-Methode und $(m-p)^2 m$ flops für die Orthogonalisierung. Der Gesamtaufwand beträgt also bei kleinem Rangdefekt in jedem Fall $\mathcal{O}(m)$ flops sowie nochmals $\mathcal{O}(m)$ flops für jede Auswertung von $c = \bar{W}_1^* \mathcal{P}_{\mathcal{R}(A)}$.

Zur Berechung von Schritt C ergeben sich dual zwei Möglichkeiten:
Methode C_1: $\mathcal{R}(A^*) = \mathcal{R}(V_1)$; also

$$x = \mathcal{P}_{\mathcal{R}(A^*)} {T_D}^{-1} \begin{pmatrix} u \\ 0 \end{pmatrix} = V_1 ({V_1}^* V_1)^{-1} u \tag{3.10}$$

Methode C_2: $\mathcal{R}(A^*)^\perp = \mathcal{R}(\bar{V}_2)$; also

$$x = \left(Id - \mathcal{P}_{\mathcal{R}(A^*)}{}^\perp \right) {T_D}^{-1} \begin{pmatrix} u \\ 0 \end{pmatrix} = \bar{V}_1 u - \bar{V}_2 \left(\bar{V}_2{}^* \bar{V}_2 \right)^{-1} \bar{V}_2{}^* \bar{V}_1 u \tag{3.11}$$

Eine äquivalente Aufwands- und Stabilitätsbetrachtung läßt sich auch hier durchführen mit dem Ergebnis, daß wieder für den Fall kleinen Rangdefekts die Methode C_2 erheblich geringenen Aufwand erfordert. Die Stabilität der Methode C_1 hängt von der Kondition von V_1 ab, während für die Stabilität von C_2 neben der Kondition von $\bar{V}_2$ auch die Qualität der Approximation $\mathcal{R}(\bar{V}_1) \approx \mathcal{R}(V_1) = \mathcal{R}(A^*)$ von Bedeutung ist; im Falle guter Approximation ist die Subtraktion gut konditioniert, da $\mathcal{P}_{\mathcal{R}(A^*)}{}^\perp \bar{V}_1 u \ll \bar{V}_1 u$.

Bemerkung 3.5: Falls A vollen Rang hat, vereinfacht sich zumindest einer der beiden Schritte A/C, da die zugehörige Projektion unterbleiben kann. Hat etwa A vollen Spaltenrang $p = m$, dann ist $\mathcal{P}_{\mathcal{R}(A)} = Id$ und ${T_R}^{-1} = (\bar{W}_1^*)$; Schritt A reduziert sich dann zu

$$c = \bar{W}_1^* b.$$

Bemerkung 3.6: Falls eine der Transformationen orthogonal ist, dann fallen beide Methoden zur Berechnung des zugehörigen Projektors zusammen; z.B. für T_D gilt dann

$$\begin{aligned} T_D{}^{-1} = T_D{}^* \Rightarrow \qquad & \bar{V}_1 = V_1, \quad \bar{V}_2 = V_2 \\ x \overset{C_1}{=} \; & V_1 \underbrace{(V_1{}^* V_1)}_{Id}{}^{-1} u = V_1 u \qquad \text{oder} \\ x \overset{C_2}{=} \; & V_1 b - \underbrace{V_1{}^* V_2}_{0} (V_2{}^* V_2)^{-1} V_2 u = V_1 u \end{aligned} \tag{3.12}$$

Im Sinne der obigen Analyse sind solche Transformationen auch am stabilsten.

Bemerkung 3.7: Im Kontext der Pfadverfolgung ist in der Notation dieses Kapitels $p = m = n - 1$; für Schritt A trifft also Bemerkung 3.5 zu. In Schritt C_2 entfällt bei der Stabilitätsbetrachtung das Problem mit der Kondition von $\bar{V}_2$, da die Matrix nur aus einer Spalte besteht.

Bemerkung 3.8: In der Literatur (z.B. [32, 42]) findet sich meist ein Schema, das von einer Zerlegung $A = BC$ in eine (m, p)-Matrix B und eine (p, n)-Matrix C von jeweils vollem Rang ausgeht; aus dieser Zerlegung läßt sich dann das Lösungsschema $A^+ = C^+ B^+ = C^*(CC^*)^{-1}(B^*B)^{-1}B^*$ erhalten. Werden die Matrizen B und C durch beliebig gewählte Spalten bzw. Zeilen X, Y erweitert, so daß die entstehenden Matrizen $(B\,X)$ und $\binom{C}{Y}$ vollen Rang haben, dann paßt dieses Schema in das Schema (3.4)

$$A = BC \iff A = \begin{pmatrix} B & X \end{pmatrix} \begin{pmatrix} Id_p & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} C \\ Y \end{pmatrix}$$

und das Lösungsschema erweist sich als das Verfahren $A_1/B/C_1$. Die Idee einer Zerlegung wie in Satz 3.2 findet sich hingegen in der angegebenen Literatur nicht; statt dessen wird das einfachere Lösungsschema im Spezialfall durch Anwendung der Sherman-Morrison-Woodbury-Formel auf aus $A_1/B/C_1$ entstandene Formeln hergeleitet. Hier wurde die Idee aus den Spezialfällen der QR-Zerlegung in [13] und der LU-Zerlegung in [34] entwickelt.

Bemerkung 3.9: Die Bezeichnung *Normalengleichungen* für Methode A_1 sind durch die Minimierungseigenschaft der Pseudoinversen motiviert.

Nämlich kann die Gleichung (3.8) auch über die Bedingung

$$\|T_R\begin{pmatrix} c \\ 0 \end{pmatrix} - b\|^2 = \min$$

hergeleitet werden; durch Differenzieren ergeben sich die Gleichungen

$$\frac{d}{dc}\|T_R\begin{pmatrix} c \\ 0 \end{pmatrix} - b\|^2 = \frac{d}{dc}\left(T_R\begin{pmatrix} c \\ 0 \end{pmatrix} - b\right)^* \left(T_R\begin{pmatrix} c \\ 0 \end{pmatrix} - b\right) = 2\,(W_1^* W_1 c - W_1^* b) = 0,$$

die in der Literatur unter Normalengleichungen geführt werden.

3.3 Realisierungen des Lösungsschemas

3.3.1 Singulärwertzerlegung

Als zuverlässigste Methode, die Pseudoinverse einer Matrix zu berechnen, gilt die *Singulärwertzerlegung* ([19, 12]). Dabei wird die Matrix A mittels zweier orthogonaler Transformationen $T_R = (U \;\; \Psi)$ und $T_D = (V \;\; \Phi)$ auf Diagonalform gebracht:

$$A = \begin{pmatrix} U & \Psi \end{pmatrix} \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V^* \\ \Phi^* \end{pmatrix} \qquad \text{mit} \quad \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_p) \tag{3.13}$$

wobei die $\sigma_1 \geq \ldots \geq \sigma_p > 0$ die nach der Größe sortierten Singulärwerte darstellen. Nach Bemerkung 3.6 fallen die Varianten zur Berechnung von $A^+ b = x$ zusammen zu

$$A^+ b = V \Sigma^{-1} U^* b.$$

Die Singulärwertzerlegung gilt als besonders geeignet zum Rangentscheid einer Matrix; der Rang wird dabei als das größte p festgelegt, für das $\operatorname{cond}_2(A) = \sigma_1/\sigma_p < 1/\varepsilon$ mit einer vorgegebenen Genauigkeit ε gilt. Insbesondere für die Berechnung der Pseudoinversen bei bekanntem Rang existieren jedoch weniger aufwendige Verfaheren, die ebenfalls akzeptable Resultate liefern. Besonders bei Berücksichtigung von Sparseness-Effekten ist die Singulärwertzerlegung sehr ineffektiv, da sie auf die Besetztheitsstruktur von A keinerlei Rücksicht nimmt. Auf eine Variante, die *unvollständige Singulärwertzerlegung*, die nur die Kernvektoren Φ und Ψ berechnet, wird in Unterabschnitt 3.3.4 zurückgegriffen.

3.3.2 QR-Zerlegung

Eine weniger aufwendige Methode, die auch im ursprünglichen `Alcon` implementiert ist, basiert auf einer QR-Zerlegung mit Householder-Reflexionen, wobei die Pivotstrategie von Businger und Golub Anwendung findet ([13, 12]). Dabei wird A mittels Reflexions-Matrixen auf obere Dreiecksgestalt transformiert, so daß im k-ten Schritt die Zerlegung aus:

$$A\Pi_1 \cdots \Pi_k = A^{(k)} = Q_k \cdots Q_1 \begin{pmatrix} R^{(k)} & S^{(k)} \\ 0 & T^{(k)} \end{pmatrix}$$

besteht; die Pivotstrategie besteht nun darin, die die Spalte $k+1$ mit derjenigen Spalte i unter den letzten $n-k-1$-Spalten zu tauschen, für die $T_i^{(k)}$ maximale Norm hat. Werden die Diagonalelemente von $R^{(k)}$ mit r_{jj}, $1 \leq j \leq k$ bezeichnet, dann sind die $|r_{jj}|$ gerade die Normen der nach vorne getauschten Spalten von $T^{(j)}$. Aufgrund der Pivotwahl gilt $|r_{11}| \geq \ldots \geq |r_{kk}|$ für alle k. Damit läßt sich ein Rangentscheid dadurch durchführen, daß der Rang der Matrix festgelegt wird durch dasjenige p, für das als letztes die Relation

$$\frac{|r_{11}|}{|r_{pp}|} > 1/\varepsilon$$

gilt mit einer vorgebenen Genauigkeit ε. Der dabei verwendete Quotient $\mathrm{sc}\,(A) := |r_{11}|/|r_{pp}|$ wurde von Deuflhard und Sautter in [13] als *Subkondition* von A eingeführt; der Name rührt daher, daß gezeigt werden kann, daß $\mathrm{sc}\,(A) \leq \mathrm{cond}_2\,(A)$ gilt.

Ist der Rang erst einmal festgesetzt, dann sieht die Zerlegung also folgendermaßen aus:

$$A\Pi = Q \begin{pmatrix} R & S \\ 0 & 0 \end{pmatrix}$$

mit einer regulären (p,p)-Matrix R und einer $(p, n-p)$-Matrix S. Die Zerlegung paßt zwar in das Schema (3.4) mittels $A = Q \left(\begin{smallmatrix} Id & 0 \\ 0 & 0 \end{smallmatrix}\right) \left(\begin{smallmatrix} R & S \\ 0 & Id \end{smallmatrix}\right) \Pi^*$; aber die Methode C_2 ist nicht so ohne weiteres anwendbar, da die Inverse von $T_D{}^* = \left(\begin{smallmatrix} R & S \\ 0 & Id \end{smallmatrix}\right) \Pi^*$ nicht direkt zur Verfügung steht. Statt dessen wird eine Methode verwendet, in der S eliminiert wird, so daß $\bar{A} = R$.

In [32] wird vorgeschlagen, die Matrix S durch Householder-Transformationen von rechts zu eliminieren, so daß sich die Zerlegung als $A = Q \left(\begin{smallmatrix} R & 0 \\ 0 & 0 \end{smallmatrix}\right) \hat{Q}^* \Pi^*$ schreiben läßt. Das Verfahren hat dann wegen Bemerkung 3.6

ähnliche Stabilitätsvorteile wie die Singulärwertzerlegung, ist aber fast genauso aufwendig.

Statt dessen wird, dem Artikel [13] folgend, eine Block-Gauß-Elimination verwendet. Nach Einführung der Matrix V als die eindeutige Lösung der Matrixgleichung $RV = S$ läßt sich durch die Elimination $\left(\begin{smallmatrix} R & S \\ 0 & 0 \end{smallmatrix}\right) = \left(\begin{smallmatrix} R & 0 \\ 0 & 0 \end{smallmatrix}\right)\left(\begin{smallmatrix} Id & V \\ 0 & Id \end{smallmatrix}\right)$ die Zerlegung auf die Form (3.4) des Zerlegungssatzes bringen:

$$A = Q \begin{pmatrix} R & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} Id_p & V \\ 0 & Id_{n-p} \end{pmatrix} \Pi^*$$

wobei $T_R = Q$ als orthogonale Transformation unproblematisch ist. Bezüglich $T_D = \Pi\left(\begin{smallmatrix} Id_p & 0 \\ V^* & Id_{n-p} \end{smallmatrix}\right)$, für das sich ${T_D}^{-*} = \Pi\left(\begin{smallmatrix} Id_p & -V \\ 0 & Id_{n-p} \end{smallmatrix}\right)$ explizit angeben läßt, unterscheiden sich die Verfahren C_1 und C_2 voneinander.

Die Verfahren lauten dann:

Schritt A: Berechne

$$\begin{pmatrix} c \\ d \end{pmatrix} = Q^* b$$

Schritt B: Löse

$$Ru = c$$

Schritt C: Berechne

$$\begin{aligned} x &\overset{\mathrm{C}_1}{=} \Pi \begin{pmatrix} Id_p \\ V^* \end{pmatrix} (Id_p + VV^*)^{-1} u \\ &\overset{\mathrm{C}_2}{=} \Pi \begin{pmatrix} u \\ 0 \end{pmatrix} + \Pi \begin{pmatrix} -V \\ Id_{n-p} \end{pmatrix} (Id_p + V^*V)^{-1} V^* u \\ \text{bzw.} \\ &\overset{\mathrm{C}_2}{=} \Pi \left(\begin{pmatrix} u \\ 0 \end{pmatrix} - \mathcal{P}_{\left(\begin{smallmatrix} -V \\ Id \end{smallmatrix}\right)} \begin{pmatrix} u \\ 0 \end{pmatrix} \right) \end{aligned}$$

wobei der Projektor in der letzten Zeile mit einer Orthogonalisierung von $\Pi\left(\begin{smallmatrix} -V \\ Id \end{smallmatrix}\right)$ berechnet wird. Im Artikel [13] wird die Methode C_2 vorgeschlagen mit einer Cholesky-Zerlegung von $Id + V^*V$, um die Inverse in der zweiten Zeile auszuwerten; im Falle höheren Rangdefekts $n - p > 1$ ist aber explizite Berechnung der Projektion wie in der dritten Zeile vorzuziehen, da sie stabiler und nur unwesentlich aufwendiger ist.

Für die Stabilität des Schrittes C ist die Größe von V entscheidend; denn sowohl für die Kondition von $V_1 = \Pi\left(\begin{smallmatrix} Id_p \\ V^* \end{smallmatrix}\right)$ als auch für die Qualität der

Approximation $\mathcal{R}(\bar{V}_1) = \mathcal{R}(\Pi\binom{Id_p}{0}) \approx \mathcal{R}(V_1) = \mathcal{R}(\Pi\binom{Id_p}{V^*})$ und die Kondition von $\bar{V}_2 = \Pi\binom{-V}{Id_{n-p}}$ ist ein kleines V von Vorteil. Die Pivotstrategie sorgt aber für die Relation $|r_{ii}| \geq |r_{ij}|$ für alle $i < j$, so daß sich für die Elemente von $V = (v_{ik})$ wegen

$$v_{ik} = \frac{1}{r_{ii}}\Big(r_{i,k+p} - \sum_{l=p}^{i+1} r_{il} v_{lk}\Big)$$

induktiv die Abschätzung

$$|v_{ik}| \leq 2^{p-i} \tag{3.14}$$

zeigen läßt und somit die Größe von V beschränkt bleibt. Die Abschätzung ist dabei zumindest für die QR-Zerlegung sehr pessimistisch.

Die QR-Zerlegung erweist sich somit als günstiger im Aufwand als die Singulärwertzerlegung, ohne wesentlich an Stabilität zu verlieren. Sie ist daher für kleine bis mittelgroße Matrizen, die voll belegt sind, vorzuziehen. Für Sparse-Matrizen ist sie aber ungeeignet, da schon eine Householder-Transformation zumeist jegliche Besetztheitsstruktur zerstört. Eine Idee, die die Sparsity bewahrt, besteht darin, die Householder-Reflexionen durch Givens-Rotationen zu ersetzen. Dieser Ansatz wird aber hier nicht weiter verfolgt; statt dessen wird ein Verfahren, basierend auf einer LU-Zerlegung, vorgestellt.

3.3.3 LU-Zerlegung

Wohl die populärste Methode zur Lösung von regulären Gleichungssystemen ist die Gauß-Elimination mit Pivotsuche; dies gilt insbesondere im Sparse-Kontext, da sie mit der Belegtheitsstruktur einer Matrix bei geeigneter Pivotstrategie am pfleglichsten umgeht.

Wie schon bei der Diskussion der Stabilität der Berechnung der Pseudoinversen bemerkt, sind hier orthogonale Transformationen vorteilhafter, sowohl was die Stabilität als auch die Einfachheit der Algorithmen angeht. Für den Erhalt von Sparsity ist aber Gauß-Elimination vorzuziehen; daher wurden auch hierfür Verfahren entwickelt.

Problematisch ist bei der Gauß-Elimination der Rangentscheid. Gewöhnlich wird davon ausgegangen, daß bei Matrizen mit Rang p zumindest vollständige Pivotsuche im $p+1$-ten Eliminationsschritt in dem noch nicht zerlegten Rest $A^{(p)}$ in (3.15) nur vernachlässigbar kleine Pivots findet, so daß

sich ein Rangentscheid der Form

$$|a_{ij}| < \|A\|\varepsilon \qquad \text{für alle } a_{ij} \text{ in } A^{(p)}$$

anwenden läßt. Jedoch braucht dies, insbesondere bei eingeschränkter Pivotsuche, nicht immer der Fall zu sein. Chan hat in [6] gezeigt, daß in exakter Arithmetik bei fast singulären Matrizen nur dann kleine Pivots auftauchen, wenn solche Elemente a_{ij} erst in den letzten Schritten als Pivots gewählt werden, für die die Inverse der Adjungierten an derselben Stelle entsprechend „große“ Elemente ${a_{ij}}^{-*}$ besitzt. Da die Inverse einer Sparse-Matrix im allgemeinen voll besetzt ist, ist die Wahrscheinichkeit, ein solches Element zu finden, insbesondere bei durch Erhalt der Sparsity eingeschränkter Pivotstrategie gering.

Die Probleme des Rangentscheids werden jedoch zunächst zurückgestellt, zumal der Rangentscheid in dieser Arbeit aus analytischen Erwägungen gegeben ist. Zunächst werden die verschiedenen Varianten des Algorithmus vorgestellt unter der Annahme, daß die Zerlegung im $p+1$-ten Schritt ein kleines Pivot angetroffen hat.

Gegeben ist also eine Zerlegung

$$P^*AQ = LU = \begin{pmatrix} L_{11} & 0 \\ L_{21} & Id_{m-p} \end{pmatrix} \begin{pmatrix} D_{11} & 0 \\ 0 & A^{(p)} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & Id_{n-p} \end{pmatrix} \tag{3.15}$$

wobei P und Q durch die Pivotstrategie erzeugte Permutationsmatrizen sind, L_{11} und U_{11} untere bzw. obere (p,p)-Dreiecksmatrizen sind und D_{11} eine (p,p)-Diagonalmatrix, wobei je nach dem, ob eine LU- oder LDU-Zerlegung stattgefunden hat, entweder $D = Id_p$ ist oder U auf der Diagonalen nur Einsen zu stehen hat. L_{21} ist eine $(m-p,p)$ und U_{12} eine $(p,n-p)$-Matrix. $A^{(p)} \approx 0$ bezeichnet den durch den Rangentscheid vernachlässigten Teil von A.

Durch Anwendung des Schemas (3.4) mit $\bar{A} = D_{11}$, $T_R = P\left(\begin{smallmatrix} L_{11} & 0 \\ L_{21} & Id_{m-p} \end{smallmatrix}\right)$ und $T_D = Q\left(\begin{smallmatrix} U_{11}^* & 0 \\ U_{12}^* & Id_{n-p} \end{smallmatrix}\right)$ läßt sich die Pseudoinverse $x = A^+b$ mit der Methode $\mathrm{A}_1/\mathrm{B}/\mathrm{C}_1$ berechnen:

Schritt $\mathbf{A}_1$: Löse

$$({L_{11}}^*L_{11} + {L_{21}}^*L_{21})\,c = \begin{pmatrix} {L_{11}}^* & {L_{21}}^* \end{pmatrix} P^*b$$

Schritt $\mathbf{B}_1$: Löse

$$D_{11}u = c$$

Schritt $\mathbf{C}_1$: Löse

$$(U_{11}{U_{11}}^* + U_{12}{U_{12}}^*)\, y = u; \quad x = Q \begin{pmatrix} {U_{11}}^* \\ {U_{12}}^* \end{pmatrix} y$$

Diese Methode ist wieder [32] entnommen, wobei dort aus Gründen der Stabilität im Schritt C_1 die LDU-Zerlegung empfohlen wird. Björk und Duff haben in [4] diesen Algorithmus für den überbestimmten Vollrangfall $m > n = p$, in dem sich Schritt C_1 zu $x = Q{U_{11}}^{-1}u$ vereinfacht, unter Einbeziehung von Sparse-Techniken einschließlich einer Sparse-Cholesky-Zerlegung von $(L_{11}{L_{11}}^* + L_{12}{L_{12}}^*)$ realisiert. Für den rangdefekten Fall muß aber auch noch die Belegtheitsstruktur im Schritt C_1 einbezogen werden.

In [34] hat Sautter ein anderes Verfahren vorgeschlagen, welches der Vorgehensweise bei der QR-Zerlegung ähnelt. Dazu wird mittels der durch

$${L_{11}}^*W = {L_{21}}^* \qquad U_{11}V = U_{12}$$

definierten W und V die Zerlegung (3.15) geschrieben als:

$$A = P \begin{pmatrix} Id_p & 0 \\ W^* & Id_{m-p} \end{pmatrix} \begin{pmatrix} L_{11}D_{11}U_{11} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} Id_p & V \\ 0 & Id_{n-p} \end{pmatrix} Q^* \tag{3.16}$$

Da nun die Zerlegung mit $\bar{A} = L_{11}D_{11}U_{11}$ in das Schema (3.4) paßt und sowohl $T_R = P\left(\begin{smallmatrix} Id_p & 0 \\ W^* & Id_{m-p} \end{smallmatrix}\right)$ und $T_D = Q\left(\begin{smallmatrix} Id_p & 0 \\ V^* & Id_{n-p} \end{smallmatrix}\right)$ als auch ${T_R}^{-*} = P\left(\begin{smallmatrix} Id_p & -W \\ 0 & Id_{m-p} \end{smallmatrix}\right)$ und ${T_D}^{-*} = Q\left(\begin{smallmatrix} Id_p & -V \\ 0 & Id_{n-p} \end{smallmatrix}\right)$ sich explizit angeben lassen, lassen sich beide Verfahren zur Berechnung der Pseudoinversen gemäß dem Schema A/B/C ausführen. Das Schema $\mathrm{A}_2/\mathrm{B}/\mathrm{C}_2$ ist dabei aber für kleinen Rangdefekt nicht nur effizienter, sondern benötigt auch keine Sorgfalt im Umgang mit der Sparsity-Struktur, da sowohl bei der Berechnung und Zerlegung von $\bar{W}_2^*\bar{W}_2 = Id + W^*W$ bzw. $\bar{V}_2^*\bar{V}_2 = Id + V^*V$ als auch bei der Orthogonalisierung von $\bar{W}_2$ bzw. $\bar{V}_2$ eine eventuelle Zerstörung der Belegtheitsstruktur unerheblich ist.

Damit lautet der Algorithmus $\mathrm{A}_2/\mathrm{B}/\mathrm{C}_2$:

Schritt $\mathbf{A}_2$: Berechne

$$c = b_p + (\mathcal{P}_{\binom{-W}{Id}} Pb)_p$$

wobei b_p der Vektor ist, der aus den ersten p Elementen von Pb besteht (oder in Formeln: $b_p = (Id_p\ 0)\ Pb$).

Schritt $\mathbf{B}_2$: Löse

$$L_{11} D_{11} U_{11} u = c$$

Schritt $\mathbf{C}_2$: Berechne

$$x = Q \begin{pmatrix} u \\ 0 \end{pmatrix} - Q\mathcal{P}_{\binom{-V}{Id}} \begin{pmatrix} u \\ 0 \end{pmatrix}$$

Dieser Algorithmus ist im Gegensatz zum Schema $\mathrm{A}_1/\mathrm{B}/\mathrm{C}_1$ unabhängig davon, ob eine LU oder LDU-Zerlegung durchgeführt wird; in beiden Fällen werden dieselben W und V berechnet.

Wenn vollständige Pivotsuche durchgeführt wird, ist auch dieses Verfahren stabil; denn die Pivotsuche garantiert $|l_{ji}| \leq |l_{ii}|$ und $|u_{ij}| \leq |u_{ii}|$ für $i < j$. Daher läßt sich dasselbe Argument wie bei der QR-Zerlegung anwenden, welches die Beschränktheit von W und V in einer Form wie in (3.14) ergibt.

Falls die Matrix vollen Rang hat, genügt eine Form des Pivoting; falls $m > n = p$, dann existiert V nicht und nur die Bedingung $|l_{ji}| \leq |l_{ii}|$ muß erfüllt sein, um die Größe von W zu begrenzen; daher ist hier nur Spaltenpivotsuche erforderlich. Im anderen Fall $n > m = p$ ist hingegen nur Zeilenpivotsuche erforderlich.

Bei Anwendung von Zerlegungen unter Berücksichtigung der Belegtheitsstruktur wird jedoch oft nur eine eingeschränkte Pivotsuche der Form

$$|a_{pivot}| > u * \max\{|a_{ij}| \mid (i,j) \text{ im Bereich der Pivotsuche } \}$$

mit einem Stabilitätsfaktor $u < 1$ durchgeführt. Dann lassen sich allerdings nur noch Abschätzungen der Form

$$|v_{ik}| \leq \frac{1}{u}\left(1 + \frac{1}{u}\right)^{p-i}$$

zeigen. Bei numerischen Testläufen erwies sich jedoch das Verfahren trotzdem als hinreichend zuverlässig.

3.3.4 Deflation

Um dem Problem des Rangentscheids bei einer LU-Zerlegung zu begegnen, insbesondere, wenn aufgrund eingeschränkter Pivotstrategien keine kleinen Pivots zu erwarten sind, wird in Rückgriff auf Abschnitt 3.3.1 die angedeutete Idee der unvollständigen Singulärwertzerlegung präzisiert. Die Kombination beider Techniken bildet dann das geeignete Werkzeug zur Berechnung der Pseudoinversen für große, dünnbesetzte Matrizen.

Die Idee, die aus [38] zitiert wird, besteht darin, daß für eine gegebene, quadratische Matrix A, für die trotz ihres Rangdefekts ein Löser $Ax = b$ existiert, wie es bei einer LU-Zerlegung der Fall sein kann, linke und rechte Kernvektoren berechnet werden, und mit ihrer Hilfe der Anteil von b bzw. x, der nicht in $\mathcal{R}(A)$ bzw. $\mathcal{R}(A^*)$ liegt, wegprojiziert wird.

Formalisieren läßt sich die Methode durch die Beobachtung, daß für die Singulärvektoren U in der Singulärwertzerlegung (3.13) gilt, daß $\mathcal{R}(U) \cong \mathbb{R}^p$ und die Isomorphie für $x \in \mathbb{R}^p$ durch $y = Ux$ sowie für $y \in \mathcal{R}(U)$ durch $x = U^*y$ vermittelt wird; dabei ist $UU^* = Id_p$ und $U^*U = \mathcal{P}_U = Id_{\mathcal{R}(U)}$. Des weiteren läßt sich die Projektion auf $\mathcal{R}(U)$ auch mit Hilfe nur von Ψ als

$$U^*U = Id - \Psi^*\Psi$$

berechnen. Wird jetzt die Transformation T_R anstelle als Abbildung

$$T_R\colon \mathbb{R}^p \times \mathbb{R}^{m-p} \to \mathbb{R}^m, \qquad T(x,z) = Ux + \Psi z$$

mittels dem Isomorphismus $U^*\colon \mathcal{R}(U) \to \mathbb{R}^p$ als Abbildung

$$T_R\colon \mathcal{R}(U) \times \mathbb{R}^{m-p} \to \mathbb{R}^m. \qquad T(y,z) = UU^*y + \Psi z$$

geschrieben, oder in Matrixform

$$T_R = \begin{pmatrix} UU^* & \Psi \end{pmatrix} = \begin{pmatrix} Id - \Psi\Psi^* & \Psi \end{pmatrix}$$

sowie

$$T_D = \begin{pmatrix} VV^* & \Phi \end{pmatrix} = \begin{pmatrix} Id - \Phi\Phi^* & \Phi \end{pmatrix}$$

dann gilt für die Adjungierte $T_R{}^*\colon \mathbb{R}^m \to \mathcal{R}(U) \times \mathbb{R}^{m-p}$, daß

$$\begin{aligned} T_R{}^*T_R &= \begin{pmatrix} \mathcal{P}_U{}^* \\ \Psi^* \end{pmatrix} \begin{pmatrix} \mathcal{P}_U & \Psi \end{pmatrix} \\ &= \begin{pmatrix} \mathcal{P}_U^2 & 0 \\ 0 & \Psi^*\Psi \end{pmatrix} = Id_{\mathcal{R}(U)\times\mathbb{R}^p} \end{aligned}$$

und

$$\begin{aligned} T_R T_R{}^* &= \begin{pmatrix} \mathcal{P}_U & \Psi \end{pmatrix} \begin{pmatrix} \mathcal{P}_U{}^* \\ \Psi^* \end{pmatrix} = (\mathcal{P}_U + \Psi\Psi^*) \\ &= Id_n - \Psi\Psi^* + \Psi\Psi^* = Id_n \end{aligned}$$

also $T_R{}^* = T_R{}^{-1}$ und ebenso $T_D{}^* = T_D{}^{-1}$.

Damit lautet die Zerlegung

$$A = \begin{pmatrix} \mathcal{P}_U & \Psi \end{pmatrix} \begin{pmatrix} A_d & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \mathcal{P}_V{}^* \\ \Phi^* \end{pmatrix}$$

wobei $A_d = \mathcal{P}_U A \mathcal{P}_V$.

Da die Transformationen orthogonal sind, fallen beide Lösungsschemata zusammen zu

Schritt A: Projiziere

$$c = \mathcal{P}_U b = b - \Psi\Psi^* b$$

Schritt B: Löse

$$Au = c$$

Schritt C: Projiziere

$$x = \mathcal{P}_V u = u - \Phi\Phi^* u$$

Zur Anwendung dieser Methode müssen die rechten und linken Singulärvektoren Φ, Ψ berechnet werden. Dafür wird wie in [7] eine Verallgemeinerung der inversen Vektor-Iteration, die „inverse Block-Iteration“ verwendet, die ebenfalls nur einen Löser für $Ax = b$ und zusätzlich für $A^*x = b$ benötigt.

Falls der Rang p bekannt ist, wird die Iteration mit einer zufällig gewählten $(n, n-p)$-Matrix Φ^0 gestartet. In jedem Schritt wird zunächst das Gleichungssystem $A^*\hat{\Psi}^i = \Phi^i$ gelöst und danach die Spalten von $\hat{\Psi}^i$ mittels dem modifizierten Gram-Schmidt-Algorithmus orthogonalisiert zu $\hat{\Psi}^i = \Psi^i R^i_\Psi$. Danach wird $A\hat{\Phi}^{i+1} = \Psi^i$ gelöst und zu $\hat{\Phi}^{i+1} = \Phi^{i+1} R^{i+1}_\Phi$ orthogonalisiert. Dabei streben wegen

$$\Psi^* A \Phi = \Phi^* A^* \Psi = \Delta = \operatorname{diag}(\sigma_{p+1}, \ldots, \sigma_n)$$

die R^i_Ψ und R^i_Φ gegen die Inverse der Diagonalmatrix Δ, die die vernachlässigbar kleinen Singulärwerte enthält, wenn Ψ^i und Φ^i gegen Ψ und Φ konvergieren. Die Iteration wird beendet, wenn $\Phi^i \approx \Phi^{i+1}$ und $\Psi^i \approx \Phi^{i+1}$, wobei konkret das heuristische Abbruchkriterium $\|\phi^i_j - \phi^{i+1}_j\|^2 < \varepsilon/\sigma_{p+k}$ für die Spalten von Φ^i und eine äquivalente Bedingung für die Spalten von Ψ^i erfüllt ist. Dabei müssen die Diagonalelemente von R^i_Φ und R^i_Ψ als Näherungen für die Singulärwerte herhalten. Eine ausführlichere Beschreibung des Algorithmus mit Konvergenzanalyse findet sich z.B. in [41].

Bei unbekanntem Rang p wird das Verfahren noch um eine äußere Iteration erweitert, in der der vermutete Rangdefekt $k = n - p$ beginnend mit $k = 1$ hochgezählt wird. In jedem Schleifendurchgang werden mittels der inversen Block-Iteration Approximationen für linke und rechte Singulärvektoren Φ_k, Ψ_k zu den k kleinsten Singulärwerten Δ_k berechnet. Ist dann der größte der Singulärwerte σ_{n-k+1} großer als $\varepsilon\|A\|$, wobei $\|A\|$ eine zu $\|A\|_2 = \sigma_1$ möglichst gut verträgliche Norm ist, dann werden die zu σ_{n-k+1} gehörigen Singulärvektoren als nicht mehr in $\mathcal{N}(A)$ bzw. $\mathcal{N}(A^*)$ angenommen und der Rankdefekt zu $k-1$ festgesetzt; die restlichen Spalten von Φ_k und Ψ_k spannen dann $\mathcal{N}(A)$ und $\mathcal{N}(A^*)$ auf. Nebenbei ergibt sich als Konditionsschätzer $\kappa(A) = \|A\|/\sigma_{n-k+1}$. Als Norm wird dabei eine normalisierte Frobenius-Norm $\|A\|_F := (\sum a^2_{ij}/n)^{1/2}$ verwendet. Der Faktor n statt n^2 führt dabei zu einer zur 2-Norm besser verträglichen Norm, da insbesondere bei dünn besetzen Matrizen die Anzahl der Einträge eher $\mathcal{O}(n)$ anstelle von $\mathcal{O}(n^2)$ ist.

Somit lautet der Algorithmus zur Berechnung der linken und rechten Singulärvektoren Φ und Ψ:

Beginne mit $k = 1$ und einen Zufallsvektor $\Phi^0_1 = (\phi^0)$
  Iteriere mit $i = 1, 2, \ldots$

Löse	$A^*\hat{\Psi}^i = \Phi^{i-1}$
Orthogonalisiere	$\Psi^i R = \hat{\Psi}^i$
falls $i > 1$	$\max_1 = \max\{\|\psi^i_j - \psi^{i-1}_j\|^2/r_{jj} \mid j = 1 \ldots k\}$
Löse	$A\hat{\Phi}^i = \Psi^i$
Orthogonalisiere	$\Phi^i R = \hat{\Phi}^i$
falls $i > 1$	$\max_2 = \max\{\|\phi^i_j - \phi^{i-1}_j\|^2/r_{jj} \mid j = 1 \ldots k\}$

  bis $i > 1$ und $\max_1, \max_2 < \varepsilon\|A\|$

  Partitioniere $R = \left(\begin{smallmatrix} r_{11} & r_{12} \\ 0 & R_{22} \end{smallmatrix}\right)$

falls $1/r_{11} > \varepsilon\|A\|$
dann
setze $\operatorname{rank}(A) = n - k + 1$
$(\phi_2^i, \dots \phi_k^i) = \Phi$
$(\psi_2^i, \dots \psi_k^i) = \Psi$
$\Delta = {R_{22}}^{-1}$
anderenfalls
erhöhe $k \to k+1$
setze neue Startvektoren
$\Phi_{k+1}^0 = (\Phi_k^i \ \phi_{k+1}^0)$
wobei ϕ_{k+1}^0 ein Zufallsvektor orthogonal zu Φ_k^i ist
und durchlaufe die Schleife von i nochmal

Der Algorithmus wurde zunächst nur für quadratische Matrizen entwickelt. Im Zusammenhang mit der LU-Zerlegung von nicht quadratischen Matrizen wird zuerst eine Zerlegung

$$A = P \begin{pmatrix} Id_p & 0 \\ W^* & Id_{m-p} \end{pmatrix} \begin{pmatrix} L_{11}U_{11} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} Id_p & V \\ 0 & Id_{n-p} \end{pmatrix} Q^*$$

durchgeführt mit einer (quadratischen) Matrix $\bar{A} = L_{11}U_{11}$, in der keine kleinen Pivots aufgetreten sind. Danach wird mit $\bar{A}$ anstelle von A der Deflationsalgorithmus durchgeführt.

Stoppt der Algorithmus schon bei $k = 1$, ist die Matrix $\bar{A}$ nicht rangdefekt und die LU-Zerlegung hat einen akzeptablen Rangentscheid durchgeführt; anderenfalls berechnet das Verfahren neben $\binom{-W}{Id}$ und $\binom{-V}{Id}$ noch zusätzliche Kernvektoren Ψ bzw. Φ, deren Anteile dann in den Schritten A_2 und C_2 auch wegprojiziert werden.

Damit ist auch für eine LU-Zerlegung mit Berücksichtigung der Belegtheitsstruktur ein Konditionsschätzer sowie die Möglichkeit eines Rangentscheids über den Umweg der Deflationstechnik möglich. Insbesondere ergibt sich die Möglichkeit eines analytischen Rangentscheides, der nicht durch das Weglassen der jeweils letzten Eliminationsschritte realisiert werden konnte. Allerdings bildet diese Methode eine Notlösung, da zum Beispiel ein Sparse-Rückwärtsfehler in der Approximation $A_d \approx A$ nicht gewährleistet ist.

3.3.5 Block-Metalöser

Die folgende Diskussion betrifft spezifische Gleichungssysteme, wie sie nur beim Lösen parameterabhängiger Gleichungen auftauchen. In diesem Fall ist es bisweilen günstig, zum Beispiel bei Stabilitätsbetrachtungen der Lösung, von einer Aufteilung

$$A = \begin{pmatrix} A_x & A_\lambda \end{pmatrix}$$

auszugehen, wobei A_x eine quadratische (n_x, n_x)-Matrix ist, die zu den n_x Variablen gehört, und A_λ eine (n_x, n_λ)-Matrix ist, die zu den n_λ Parametern gehört. Dabei besitzt A_x oft eine spezielle Struktur, etwa Selbstadjungiertheit oder eine sehr dünne Besetztheitsstruktur. Daher wird es in der Literatur, insbesondere im Kontext der Pseudo-Bogenlängen-Fortsetzung, als günstig betrachtet, nur mit einem „Black-Box"-Löser für $A_x x = b$ einen *Meta-Löser* für die auftretenden Gleichungssysteme $Ax = b$ zu konstruieren.

Zum Beispiel kommt es durch Linearisierung der in diesem Zugang definierten erweiteren Funktion G in Gleichung (1.7) im zugehörigen Newtonverfahren zu dem Gleichungssystem

$$\begin{pmatrix} A_x & A_\lambda \\ {u_x}^* & u_\lambda \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = - \begin{pmatrix} f \\ g \end{pmatrix}$$

Durch die Block-Elimination $\left(\begin{smallmatrix} A_x & A_\lambda \\ {u_x}^* & u_\lambda \end{smallmatrix}\right) = \left(\begin{smallmatrix} A_x & 0 \\ {u_x}^* & S \end{smallmatrix}\right)\left(\begin{smallmatrix} Id & V \\ 0 & Id \end{smallmatrix}\right)$ mit $V = {A_x}^{-1} A_\lambda$ und $S = u_\lambda - {u_x}^* V$ läßt sich das Gleichungssystem mittels Block-Substitution

Schritt 1: Löse $A_x u = -f$

Schritt 2: Löse $S\lambda = -g - {u_x}^* u$

Schritt 3: Berechne $x = u - V\lambda$

lösen. Diese Methode läuft in Schwierigkeiten, wenn $\operatorname{rank}(A_x) < \operatorname{rank}(A)$ gilt, also A_x singulär wird, wie es zum Beispiel bei Wendepunkten der Fall ist.

Bemerkung 3.10: Wie in Abschnitt 1.2.3 bemerkt, läßt sich das Gauß-Newton-Verfahren auch als Pseudo-Bogenlängen-Verfahren mit einer speziellen Zusatzbedingung auffassen. Für das entstehende Gleichungssystem (1.11) ist dann $u_x = -V$ und $u_\lambda = Id$, da $\left(\begin{smallmatrix} -V \\ Id \end{smallmatrix}\right)$ ein Kernvektor von A ist; das Verfahren der Block-Substitution entspricht dann genau dem Verfahren A/B/C_2 für die Zerlegung $A = (A_x\ 0)\left(\begin{smallmatrix} Id & -V \\ 0 & Id \end{smallmatrix}\right)$, z.B. ist $S = Id + V^* V$.

Diese Probleme verschwinden, wenn angenommen wird, daß für A_x nicht nur ein Löser, sondern eine Zerlegung wie in (3.4) gegeben ist. Die größte Schwierigkeit besteht dann im Aufschreiben der Formeln.

Sei also angenommen, daß eine Zerlegung

$$A_x = T_{R_x} \begin{pmatrix} \bar{A}_x & 0 \\ 0 & S_x \end{pmatrix} {T_{D_x}}^*$$

existiert mit $\operatorname{rank}(A_x) = p_x$, wobei S_x aus Gründen, die weiter unter erklärt werden, nicht mehr unbedingt als vernachlässigbar klein angesehen wird. Dann sieht die Zerlegung von A zunächst so aus:

$$A = T_{R_x} \begin{pmatrix} \bar{A}_x & 0 & {W_1}^* A_\lambda \\ 0 & S_x & {W_2}^* A_\lambda \end{pmatrix} \begin{pmatrix} {T_{D_x}}^* & 0 \\ 0 & Id_{n_\lambda} \end{pmatrix} \tag{3.17}$$

Falls nun A_x einen kleinen Rangdefekt $q_x = n_x - p_x$ hat, dann ist eine Zerlegung der $(q_x, q_x + n_\lambda)$-Matrix $(S_x \;\; {W_2}^* A_\lambda)$ leicht z.B. mittels QR-Zerlegung möglich. Sei also eine zusätzliche Zerlegung

$$\begin{pmatrix} S_x & {W_2}^* A_\lambda \end{pmatrix} = T_{R_\lambda} \begin{pmatrix} \bar{A}_\lambda & 0 \\ 0 & S_\lambda \end{pmatrix} {T_{D_\lambda}}^*$$

gegeben mit $\operatorname{rank}\left(\bar{A}_\lambda\right) = p_\lambda$; dann läßt dich die Zerlegung (3.17) auf die Form (3.4) bringen. Denn die Matrix ${W_1}^* A_\lambda$ läßt sich einfach mittels Block-Elimination entfernen, wenn V_λ als Lösung von $\bar{A}_x V_\lambda = {W_1}^* A_\lambda$ definiert wird.

$$\begin{aligned}
A &= T_{R_x} \begin{pmatrix} \bar{A}_x & 0 & {W_1}^* A_\lambda \\ 0 & S_x & {W_2}^* A_\lambda \end{pmatrix} \begin{pmatrix} {T_{D_x}}^* & 0 \\ 0 & Id_{n_\lambda} \end{pmatrix} \\
&= T_{R_x} \begin{pmatrix} \bar{A}_x & 0 & 0 \\ 0 & S_x & {W_2}^* A_\lambda \end{pmatrix} \begin{pmatrix} Id_{p_x} & 0 & V_\lambda \\ 0 & Id_{q_x} & 0 \\ 0 & 0 & Id_{n_\lambda} \end{pmatrix} \begin{pmatrix} {T_{D_x}}^* & 0 \\ 0 & Id_{n_\lambda} \end{pmatrix} \\
&= T_{R_x} \begin{pmatrix} Id_{p_x} & 0 \\ 0 & T_{R_\lambda} \end{pmatrix} \begin{pmatrix} \bar{A}_x & 0 & 0 \\ 0 & \bar{A}_\lambda & 0 \\ 0 & 0 & S_\lambda \end{pmatrix} \\
&\qquad \begin{pmatrix} Id_{p_x} & 0 \\ 0 & {T_{D_\lambda}}^* \end{pmatrix} \begin{pmatrix} Id_{p_x} & 0 & V_\lambda \\ 0 & Id_{q_x} & 0 \\ 0 & 0 & Id_{n_\lambda} \end{pmatrix} \begin{pmatrix} {T_{D_x}}^* & 0 \\ 0 & Id_{n_\lambda} \end{pmatrix}
\end{aligned}$$

Damit ist also

$$\begin{aligned} T_R &= T_{R_x} \begin{pmatrix} Id_{p_x} & 0 \\ 0 & T_{R_\lambda} \end{pmatrix} \\ T_D &= \begin{pmatrix} T_{D_x} & 0 \\ 0 & Id_{n_\lambda} \end{pmatrix} \begin{pmatrix} Id_{p_x} & 0 & 0 \\ 0 & Id_{q_x} & 0 \\ V_\lambda{}^* & 0 & Id_{n_\lambda} \end{pmatrix} \begin{pmatrix} Id_{p_x} & 0 \\ 0 & T_{D_\lambda} \end{pmatrix} \end{aligned}$$

und

$$\bar{A} = \begin{pmatrix} \bar{A}_x & 0 \\ 0 & \bar{A}_\lambda \end{pmatrix}, \quad S = S_\lambda$$

Die Berechnung nach Methode $A_2/B/C_2$ erfolgt nun nicht dadurch, daß die Matrizen T_R, T_D oder Teilmatrizen wie $\bar{V}_1$ explizit ausgerechnet werden, sondern indem T_{R_x}, T_{R_λ}, T_{D_x}, T_{D_λ} und V_λ abgespeichert werden; eine Auswertung von z.B. $c = T_R{}^{-1}b$ erfolgt dann durch Auswerten der Teilmatrizen:

$$c = T_R{}^{-1}b \quad \Leftrightarrow \quad T_{R_x}{}^{-1}b = \hat{b} = \begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \end{pmatrix} = \begin{pmatrix} \bar{W}_1^* b \\ \bar{W}_2^* b \end{pmatrix} \quad \text{und} \quad c = \begin{pmatrix} \hat{b}_1 \\ T_{R_\lambda}{}^{-1}\hat{b}_2 \end{pmatrix}$$

Die für die Projektoren erforderlichen Matrizen werden mittels

$$\bar{W}_2 = T_R{}^{-*}\begin{pmatrix} 0 \\ Id_{m-p} \end{pmatrix} \quad \text{und} \quad \bar{V}_2 = T_D{}^{-*}\begin{pmatrix} 0 \\ Id_{n-p} \end{pmatrix}$$

berechnet. Der Lösungsalgorithmus zur Berechnung von $x = A^+b$ mit der Block-Meta-Zerlegung lautet also

Schritt $\mathbf{A}_2$:

$$\begin{pmatrix} c_1 \\ c_2 \\ 0 \end{pmatrix} = T_R{}^{-1}\mathcal{P}_{\bar{W}_2}{}^{\perp}b \qquad \text{mit } c_1 \in \mathbb{R}^{p_x}, c_2 \in \mathbb{R}^{p_\lambda}$$

Schritt $\mathbf{B}_2$: Löse

$$\begin{aligned} \bar{A}_x u_1 &= c_1 \\ \bar{A}_\lambda u_2 &= c_2 \end{aligned}$$

Schritt C_2:

$$x = \mathcal{P}_{\bar{V}_2}{}^{\perp} T_D{}^{-*} \begin{pmatrix} u_1 \\ u_2 \\ 0 \end{pmatrix}$$

Ein Problem, das dabei auftritt, ist der Rangentscheid von A_x, insbesondere, wenn die Entscheidung aus analytischen Gründen gefällt wird. Dazu werden zwei Vorschläge untersucht, der eine von Govaerts und Pryce in [21] gemacht, der andere von Keller in [26] zunächst für den Spezialfall der LU-Zerlegung entwickelt und von Chan (zitiert nach dem den allgemeineren Fall $n_\lambda \geq 1$ behandelnden Artikel [7]) für der Fall der Deflation in einer Weise übertragen, die eine einfache Verallgemeinerung erlaubt.

Der Vorschlag von Govaerts und Pryce hat den Vorzug, sowohl gedanklich als auch von der algorithmischen Umsetzung her sehr einfach zu sein. Die Idee besteht darin, immer $\operatorname{rank}(A_x) = \operatorname{rank}(A)$ zu fordern; ist diese Forderung nicht zu erfüllen, wird die Matrix A_x im Löser durch eine Approximation $\tilde{A}_x \doteq A_x$ ersetzt, die vollen Rang besitzt. Der Fehler, der dadurch in der Lösung entsteht, wird durch Nachiteration sukzessive verkleinert. Der Algorithmus wird dadurch sehr vereinfacht, da die Teilmatrix $(S_x \;\; \bar{W}_2^* A_\lambda)$ wegfällt.

Mittels einer Fehleranalyse (basierend auf einer von Pryce entwickelten Methode) kommen sie dabei in [21] zu dem Schluß, daß bei einem stabilen Löser für $\tilde{A}_x x = b$ eine Nachiteration über A_x genügt, um ein ausreichend exaktes Ergebnis zu erhalten, auch wenn wegen der Approximation $\tilde{A}_x \doteq A_x$ die Matrix $\tilde{A}_x$ extrem schlecht konditioniert ist.

Im Rahmen der Pfadverfolgung schlagen sie vor, an Wendepunkten für $\tilde{A}_x$ die Ableitung eines Lösungspunktes auf dem Pfad nahe dem Wendepunkt zu nehmen. In Verbindung mit der Schrittweitenwahl von Abschnitt 1.2.5 werden allerdings normalerweise keine Punkte berechnet, die nahe genug an dem Wendepunkt liegen; in numerischen Experimenten führte diese Art der Approximation zu starken Schrittweiteneinschränkungen. Statt dessen wird für jeden der implementierten Löser eine Möglichkeit verwendet, bei Entdecken von Singularitäten die Zerlegung so zu verändern, daß eine numerisch nicht singuläre Zerlegung entsteht. Bei der QR- und LU-Zerlegung besteht diese Veränderung in dem Ersetzen von Pivots r_{jj} mit $|r_{jj}| < \|A\|\varepsilon$ durch $\tilde{r}_{jj} = \operatorname{sgn}(r_{jj})\|A\|\varepsilon$; bei der Deflation einfach in Abbrechen der Iteration, wenn der gewünschte Rangdefekt erreicht ist. Ein solches Verfahren läßt sich

auf iterative Löser natürlich nicht so einfach zu übertragen; aber Govaerts und Pryce bemerken in ihrem Artikel auch, daß sie nicht erwarten, daß ihre Methode für iterative Löser funktioniert. Zudem taucht bei fast jedem iterativen Löser in der Analyse der Konvergenzgeschwindigkeit ein Faktor proportional zu $\mathrm{cond}_2(A_x)$ auf, der in diesem Fall zu sehr langsamer Konvergenz führen würde.

Der andere Vorschlag lautet in der ursprünglichen Form bei Keller in [26], eine Rangreduktion bei A_x durchzuführen, wenn es numerisch erforderlich ist, und dann den Algorithmus des Block-Meta-Lösers durchzuführen; Chan machte dann den Vorschlag, grundsätzlich $\mathrm{rank}(A_x) = \mathrm{rank}(A) - n_\lambda$ zu fordern; dadurch entsteht bei der Zerlegung von A_x ein vernachlässigter Rest S_x mit $\|S_x\| \gg 0$, falls numerisch $\mathrm{rank}(A_x) > \mathrm{rank}(A) - n_\lambda$. Der Rang dieses Restes geht aber bei der Zerlegung von $(S_x \;\; \bar{W}_2^* A_\lambda)$ in $\bar{A}_\lambda$ ein.

Um diesen Vorschlag zu motivieren, möge die geneigte Leserin und der geneigte Leser annehmen, daß sowohl die Matrix A_x als auch das kleine System $(S_x \;\; \bar{W}_2^* A_\lambda)$ mit demselben Lösungsalgorithmus, z.B. Gauß-Elimination, zerlegt wird. Dann entspricht die Block-Meta-Zerlegung genau einer Gauß-Elimination von ganz A, wobei jedoch vor den letzten n_λ Schritten Pivotelemente aus dem Parameterblock A_λ nicht erlaubt sind. Ist also A_x sehr dünn besetzt, während A_λ eher voll ist, wie es bei parameterabhängigen Gleichungssystemen durchaus anzunehmen ist, dann verhindert diese Pivotstrategie, daß durch frühzeitige Wahl von Pivots aus dem Parameterblock viele zusätzliche Nicht-Null-Elemente entstehen. Um dann den Rang von A und $\mathcal{N}(A)$ bzw. $\mathcal{N}(A^*)$ zu bestimmen, bleibt in den verbleibenden Eliminationsschritten noch genug Möglichkeit, da $\mathrm{rank}(A)$ nicht größer sein kann als $\mathrm{rank}(A_x) + n_\lambda$.

Bei numerischen Tests erwiesen sich beide Verfahren im Kontext der Pfadverfolgung als stabil auch bei Berechnung von Wendepunkten; bei Verwendung eines Sparse-Solvers in Verbindung mit der Deflation zur Erzwingung des Rangdefekts von A_x erwies sich die Methode von Chan aber als signifikant langsamer, da die Deflation viele Iterationen benötigt, um brauchbare Singulärvektoren zu erzeugen, wenn A_x numerisch gut konditioniert ist.

Bei Verwendung der Zerlegung im Mooreschen System (2.11) - (2.13) ist es allerdings erforderlich, daß das Verfahren $\mathcal{N}(A)$ gut approximiert, damit die zweite Ableitung auf den richtigen Unterraum projiziert wird. Hier gerät das Verfahren von Govaerts und Pryce ins Hintertreffen, da bei diesem Verfahren der berechnete Nullraum von A immer eine Komponente in Richtung der

Parameter hat. Damit lassen sich Verzweigungen von Typ (2.8) nicht korrekt behandeln, da hier $\mathcal{N}(A) = \mathcal{N}(A_x) \times 0^{n_\lambda}$. Bei der Methode von Keller oder Chan taucht diese Schwierigkeit nicht auf, da für diesen Fall bei der Zerlegung von $(S_x \;\; \bar{W}_2^* A_\lambda)$ die Matrix S_x als vernachlässigt klein betrachtet werden kann; damit ergibt sich ein korrekter Nullraum $\mathcal{N}(A) \subseteq \mathbb{R}^{n_x} \times 0^{n_\lambda}$.

3.4 Lösung des Mooreschen Systems

3.4.1 Transformation auf Block-Dreiecksgestalt

Unter Zuhilfenahme des Zerlegungsschemas (3.4) wird in diesem Abschnitt ein effizientes Lösungsverfahren für die Gleichungssysteme, die bei der Newton-Iteration zur Lösung des Mooreschen Systems entstehen, entwickelt in Analogie zu dem Verfahren, das in [8] für den Spezialfall der QR-Zerlegung vorgeschlagen wurde.

Zuächst muß erst einmal die linearisierte Gleichung hingeschrieben werden:

$$\mathcal{M}\Delta x = \begin{pmatrix} C & A^* & 0 \\ A & -\alpha Id_n & z \\ 0 & z^* & 0 \end{pmatrix} \begin{pmatrix} \Delta y \\ \Delta z \\ \Delta\alpha \end{pmatrix} = \begin{pmatrix} r_y \\ r_z \\ r_\alpha \end{pmatrix} = - \begin{pmatrix} f'(y)z^* \\ f(y) - \alpha z \\ \frac{1}{2}(\|z\|^2 - 1) \end{pmatrix} = -F(x) \tag{3.18}$$

wobei $x = (y, z, \alpha)$, $A = f'(y)$ und $C = (f'(y)^* z)'$. Da

$$C = (c_{ij}) = (\frac{\partial^2}{\partial y_i \partial y_j} f^* z) = {}^1\sum_{i=1}^{n} \left(\frac{\partial^2 f_i}{\partial y_j \partial y_k} z_i \right)$$

eine selbstadjungierte Matrix ist, und daher auch die gesamte Matrix des Systems selbstadjungiert ist, werden nur Ähnlichkeitstransformationen verwendet, die diese Selbstadjungiertheit erhalten.

Zunächst werden aber die folgenden Vereinfachungen gemacht, um ein leichter lösbares System $\tilde{\mathcal{M}}\Delta x = -F$ zu erhalten:

1. In der Matrix wird das α als vernachlässigbar klein angenommen und daher $-\alpha Id_n = 0$ gesetzt; d.h. es wird die Annahme gemacht, daß die zu berechnende Verzweigung nicht all zu stark gestört ist.

[1] nur im Fall des kanonischen Skalarproduktes

2. Es wird der analytische Rangentscheid rank $(A) = n-1$ gefällt; besteht also eine Zerlegung
$$A = T_R \begin{pmatrix} \bar{A} & 0 \\ 0 & S \end{pmatrix} {T_D}^*$$
dann wird S vernachlässigt. Anstelle der Matrix $A = W_1 \bar{A} {V_1}^* + W_2 S {V_2}^*$ wird also die Matrix $\tilde{A} = W_1 \bar{A} {V_1}^*$ benutzt. Falls die Iterierten der Newtoniteration nahe am Verzweigungspunkt liegen, sollte dies keine große Veränderung sein.

Nach diesen Vereinfachungen läßt sich $\tilde{\mathcal{M}}$ mittels der Transformation
$$\mathcal{T} = \begin{pmatrix} {T_D}^{-1} & 0 & 0 \\ 0 & {T_R}^{-1} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
auf die Gestalt:
$$\mathcal{T} \begin{pmatrix} C & \tilde{A}^* & 0 \\ \tilde{A} & 0 & z \\ 0 & z^* & 0 \end{pmatrix} \mathcal{T}^* = \begin{array}{c} \begin{array}{ccccc} \overbrace{\phantom{C_{11}}}^{p} & \overbrace{\phantom{C_{12}}}^{2} & \overbrace{\phantom{\bar{A}^*}}^{p} & \overbrace{}^{1} & \overbrace{}^{1} \end{array} \\ \begin{pmatrix} C_{11} & C_{12} & \bar{A}^* & 0 & 0 \\ {C_{12}}^* & C_{22} & 0 & 0 & 0 \\ \bar{A} & 0 & 0 & 0 & w_1 \\ 0 & 0 & 0 & 0 & w_2 \\ 0 & 0 & {w_1}^* & w_2 & 0 \end{pmatrix} \begin{array}{l} \}\, p \\ \}\, 2 \\ \}\, p \\ \}\, 1 \\ \}\, 1 \end{array} \end{array} \tag{3.19}$$
bringen, wobei
$$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = {T_R}^{-1} z = \begin{pmatrix} \bar{W}_1^* z \\ \bar{W}_2^* z \end{pmatrix}$$
und
$$\begin{pmatrix} C_{11} & C_{12} \\ {C_{12}}^* & C_{22} \end{pmatrix} = {T_D}^{-1} C {T_D}^{-*} = \begin{pmatrix} {\bar{V}_1}^* C \bar{V}_1 & {\bar{V}_1}^* C \bar{V}_2 \\ {\bar{V}_2}^* C \bar{V}_1 & {\bar{V}_2}^* C \bar{V}_2 \end{pmatrix}.$$
Wird die letzte Zeile nach oben und die rechteste Spalte nach links getauscht, ist das Gleichungssystem (3.19) dann in Block-Dreiecksgestalt:
$$\begin{pmatrix} 0 & 0 & 0 & {w_1}^* & w_2 \\ 0 & C_{11} & C_{12} & \bar{A}^* & 0 \\ 0 & {C_{12}}^* & C_{22} & 0 & 0 \\ w_1 & \bar{A} & 0 & 0 & 0 \\ w_2 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta\alpha \\ \Delta u_1 \\ \Delta u_2 \\ \Delta w_1 \\ \Delta w_2 \end{pmatrix} = \begin{pmatrix} r_\alpha \\ g_1 \\ g_2 \\ h_1 \\ h_2 \end{pmatrix}$$
mit

$$\begin{pmatrix}\Delta u_1\\ \Delta u_2\end{pmatrix} = {T_D}^*\Delta y = \begin{pmatrix}{V_1}^*\Delta y\\ {V_2}^*\Delta y\end{pmatrix} \quad \text{und} \quad \begin{pmatrix}\Delta w_1\\ \Delta w_2\end{pmatrix} = {T_R}^*\Delta z = \begin{pmatrix}{W_1}^*\Delta z\\ {W_2}^*\Delta z\end{pmatrix},$$

$$\begin{pmatrix}g_1\\ g_2\end{pmatrix} = {T_D}^{-1}r_y = \begin{pmatrix}\bar{V}_1^* r_y\\ \bar{V}_2^* r_y\end{pmatrix} \quad \text{und} \quad \begin{pmatrix}h_1\\ h_2\end{pmatrix} = {T_R}^{-1}r_z = \begin{pmatrix}\bar{W}_1^* r_z\\ \bar{W}_2^* r_z\end{pmatrix}$$

Dieses Gleichungssystem ist dann lösbar, wenn die Blöcke auf der Diagonalen vollen Rang haben; dies ist aber für $\bar{A}$ an einer Verzweigung per Definition erfüllt; dort ist ebenfalls $z \in \mathcal{R}(A)^\perp$ und daher $\binom{w_1}{w_2} = \binom{0}{\pm 1}$. Zu guter letzt ist die Matrix C_{22} exakt die in Definition (2.5) benutzte Matrix und daher indefinit und damit regulär. Folglich ist das gesamte Gleichungssystem an einem Verzweigungspunkt und wegen des Banchschen Störungslemmas auch in einer ganzen Umgebung der Verzweigung regulär. Damit ist auch die in Abschnitt 2.4 offengebliebene Frage der lokalen Eindeutigkeit des Mooreschen Systems positiv beantwortet.

Bemerkung 3.11: Um diese Dreiecksform zu erhalten, muß A rangdefekt sein; anderenfalls würde die Transformation $\mathcal{T}$ nicht zu einem solchen gestaffelten System führen. Des weiteren würde die Matrix C_{22} nicht berechnet werden können, obwohl sie zum Starten der Pfadverfolgung von einer Verzweigung unerläßlich ist. Daher muß im Mooreschen System von einer rangdefekten Matrix ausgegangen werden; dies ist gemeint, wenn in Bemerkung 3.3 von einem Rangentscheid aus analytischen Gründen gesprochen wird.

Zur Umsetzung in einen Algorithmus sind neben der Zerlegung von A auch noch die Transformationen zur Berechnung der w_i und die Transformation von C zu berechnen. Die Transformation $\binom{w_1}{w_2} = {T_R}^{-1}z$ und die Berechnungen von C_{12} und C_{22} sind dabei von geringem Aufwand; der Erhalt von Besetztheitsstrukturen ist auch nicht von Relevanz. Bei der expliziten Berechnung von $C_{11} = \bar{V}_1^* C \bar{V}_1$ kann jedoch die Besetztheitsstruktur von C völlig zerstört werden. Da aber für C_{11} nur die Auswertung von Matrix-Vektor-Produkten $C_{11}z = b$ erforderlich ist, braucht C_{11} auch nicht explizit berechnet zu werden; statt dessen wird die Auswertung von $C_{11}z = b$ über

$$u = \bar{V}_1 z \qquad c = Cu \qquad b = \bar{V}_1^* c$$

berechnet.

Bemerkung 3.12: Falls $n = 1$ ist, dann läßt sich die Berechnung erheblich vereinfachen, indem $z^0 = z \equiv 1.0$ gesetzt wird, so daß die Gleichung (2.13) im Mooreschen System immer erfüllt ist. Daraus ergibt sich α unmittelbar zu $\alpha^k = f(y^k)$. Damit muß wegen $C_{22} = C$ und $\bar{A} = 0$ nur das $(2,2)$-Gleichungssystem $C\Delta y^k = -f'(y^k)^*$ gelöst werden.

3.4.2 Varianten der Nachiteration

Anstelle des Gleichungssystems $\mathcal{M}x = b$ wurde im vorherigen Abschnitt ein modifiziertes Gleichungssystem $\tilde{\mathcal{M}}x^0 = b$ gelöst. Um zu einer Lösung des ursprünglichen Systems zu gelangen, bietet sich die Nachiteration:

$$\begin{aligned} i = 0, \ldots & \\ \tilde{\mathcal{M}}\delta x^i &= r^i \\ r^{i+1} &= \left(\tilde{\mathcal{M}} - \mathcal{M}\right)\delta x^i \\ x^{i+1} &= x^i + \delta x^i \end{aligned}$$

an, die mit $r^0 = b$, $x^0 = 0$ gestartet wird.

Die Iteration wird abgebrochen, wenn $\|\delta x^{i+1}\| \geq C\|\delta x^i\|$ mit einem $C < 1$, da Divergenz oder zumindest zu langsame Konvergenz vermutet wird; falls für die Korrekturen $\|\delta x^i\|/\|x^i\| < \varepsilon$ gilt, wird die Iteration beendet und $x^{i+1} \doteq x^\infty$ als Lösung von $\mathcal{M}x = b$ akzeptiert.

Für die Realisierung der Nachiteration ergeben sich verschiedene Varianten, je nach dem welche der Modifikationen von $\tilde{\mathcal{M}}$ dabei berücksichtigt werden.

Natürlich läßt sich daß Residuum über die Auswertung sowohl von $\mathcal{M}\delta x^i$ als auch $\tilde{\mathcal{M}}\delta x^i = r^i$ realisieren. Diese *vollständige Nachiteration* läßt sich aber durch Vernachlässigung von Rundungsfehlern in der Zerlegung stark vereinfachen. Denn nach der Transformation mit $\mathcal{T}$ sieht die Differenz wie folgt aus:

$$\mathcal{T}(\tilde{\mathcal{M}} - \mathcal{M})\mathcal{T}^* = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -S^* & 0 \\ 0 & 0 & \alpha Id & 0 & 0 \\ 0 & -S & 0 & \alpha Id & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

so daß sich das Residuum zu

$$\begin{pmatrix} r_y^{i+1} \\ r_z^{i+1} \\ r_\alpha^{i+1} \end{pmatrix} = \begin{pmatrix} -V_2 S^* W_2{}^* \delta z^i \\ -W_2 S V_2{}^* \delta y^i + \alpha I d_m \delta z^i \\ 0 \end{pmatrix} \tag{3.20}$$

berechnet. Dabei können auch je nach dem die Anteile von α und S vernachlässigt werden. Diese Methode der *teilweisen Nachiteration* verringert die Auswertung des Residuums von $\mathcal{O}(n^2)$ auf $\mathcal{O}(n)$ Operationen, berücksichtigt dafür aber nicht eventuelle Instabilitäten in der Zerlegung von A.

4. Implementierung

4.1 Realisierung der Skalierung

Wie schon in Bemerkung 1.1 festgestellt wurde, ist das Verfahren der Pfadverfolgung mit dem Gauß-Newton-Verfahren als Korrektor nicht nur von dem zugrundeliegenden Problem abhängig, sondern auch von dem verwendeten inneren Produkt auf dem Urbildraum $\mathbb{R}^{n+1}$. Falls kein aus der Problemstellung vorgegebenes Produkt vorhanden ist, ist es daher sinnvoll, ein vom Problem abhängiges inneres Produkt zu verwenden. Üblich ist es dabei, sich auf eine Skalierung der Variablen zu beschränken.

Auswirkung der Skalierung

Die Skalierung kann dabei über Verwendung einer Skalierungsmatrix D eingeführt werden, so daß das bezüglich D skalierte Produkt $\langle x, y\rangle_D$ als das kanonische Skalarprodukt im $\mathbb{R}^{n+1}$ der Bilder unter D ergibt. Formal ist D also eine reguläre Abblidung zwischen zwei Hilberträumen:

$$D\colon V \to \mathbb{R}^{n+1} \qquad \text{mit} \quad V = (\mathbb{R}^{n+1}, \langle\cdot,\cdot\rangle_D) \quad \text{und} \quad \mathbb{R}^{n+1} = (\mathbb{R}^{n+1}, \langle\cdot,\cdot\rangle_{Id})$$

wobei $\langle x, y\rangle_{Id} = x^T y = \sum\limits_{i=1}^{n+1} x_i y_i$ das kanonische Skalarprodukt bezeichnet, so daß

$$\langle x, y\rangle_D := \langle Dx, Dy\rangle_{Id} = x^T D^T D y$$

Dabei wird D als Diagonalmatrix gewählt, so daß $D^T = D$.

Ist $A\colon V \to W$ eine Abbildung zwischen zwei skalierten Räumen, wobei V durch D_d und W durch D_r skaliert ist, dann berechnet sich die Adjungierte A^* aus der Definitionsgleichung

$$\langle Ax, y\rangle_{D_r} = \langle x, A^* y\rangle_{D_d}$$

zu

$$A^* = D_d^{-2} A^T D_r^2 \tag{4.1}$$

Für die direkten Löser ist es vorteilhaft, eine lineare Abbildung als Matrix bezüglich der kanonischen Normen darzustellen, da dann Rangentscheid und Pivotstrategie bezüglich unskalierten Größen berechnet werden können.

Daher wird eine skalierte Abbildung $\tilde{A}\colon \mathbb{R}^m \to \mathbb{R}^n$ über

$$\tilde{A} D_d x = D_r A x \iff \tilde{A} = D_r A D_d^{-1}$$

eingeführt. Bei sämtlichen direkten Lösern wird zuerst aus A die skalierte Matrix $\tilde{A}$ berechnet und dann mit $\tilde{A}$ die Zerlegung durchgeführt. Zum Lösen von Gleichungssystemen müssen dann auch die Variablen entsprechend skaliert werden. Dazu wird die Skalierung in die Transformationen T_D, T_R geschoben.

Bezeichne zum Beispiel T_R die Transformation ohne Skalierung und $\tilde{T_R}$ die Transformation mit Skalierung, dann ist

$$\tilde{T_R} = D_r^{-1} T_R$$

Da $\tilde{T_R}\colon \mathbb{R}^m \to V$, berechnet sich $\tilde{T_R}^*$ über (4.1) zu

$$\tilde{T_R}^* = (D_r^{-1} T_R)^T D_r^2 = T_R{}^* D_r$$

und weiter

$$\begin{aligned} \tilde{T_R}^{-1} &= T_R{}^{-1} D_r \\ \tilde{T_R}^{-*} &= D_r{}^{-1} T_R{}^{-*} \end{aligned}$$

Insbesondere ist $\tilde{T_R}^{-1} = \tilde{T_R}^*$, falls $T_R{}^{-1} = T_R{}^*$. Für $\tilde{T_D} = D_d{}^{-1} T_D$ gelten dieselben Formeln.

Da im Mooreschen System Adjungierte und Normen vorkommen, ist es auch von der Skalierung betroffen, so daß eine sorgfältige Überprüfung dieses Einflusses notwendig ist. Die Jacobi-Matrix in Gleichung (3.18):

$$\mathcal{M} = \begin{pmatrix} C & A^* & 0 \\ A & -\alpha Id_n & z \\ 0 & z^* & 0 \end{pmatrix} = \begin{pmatrix} C & D_d^{-2} A^T D_r^2 & 0 \\ A & -\alpha Id_n & z \\ 0 & (D_r^2 z)^T & 0 \end{pmatrix}$$

bleibt aber als Selbstabbildung von $V \times W \times \mathbb{R}$ mit der Skalierung

$$\mathcal{D} = \begin{pmatrix} D_d & 0 & 0 \\ 0 & D_r & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.2}$$

selbstadjungiert. Dies läßt sich unter Berücksichtigung, daß $C = \frac{\partial}{\partial y}(f'(y)^* z) = D_d^{-2} \frac{\partial}{\partial y} f'(y)^T D_r^2 z$ und daher

$$C^* = D_d^{-2} C^T D_d^2 = D_d^{-2} \underbrace{(\tfrac{\partial}{\partial y} f'(y)^T D_r^2 z)}_{\text{symmetrisch}}^T D_d^{-2} D_d^2 = C$$

bezüglich der Skalierung D_d selbstadjungiert ist, mittels

$$\mathcal{M}^* = \mathcal{D}^{-2} \mathcal{M}^{\mathcal{T}} \mathcal{D}^2 = \mathcal{M}$$

direkt nachrechnen; auch ergibt die Ausführung mit der skalierten Transformation

$$\tilde{\mathcal{T}} = \begin{pmatrix} \tilde{T_D}^{-1} & 0 & 0 \\ 0 & \tilde{T_R}^{-1} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

eine symmetrische, d.h. bezüglich des kanonischen Skalarproduktes selbstadjungierte Matrix $\tilde{\mathcal{M}} = \tilde{\mathcal{T}} \mathcal{M} \tilde{\mathcal{T}}^*$.

Verwendete Skalierungen

Urbildskalierung In [37] hat Skeel gezeigt, daß für die Stabilität der Lösung eine Skalierung am günstigsten ist, bei der alle Komponenten der skalierten Lösung von derselbe Größenordnung sind. Da eine solche Skalierung ohne Kenntnis der Lösung unmöglich zu erreichen ist, wird die Urbildskalierung, die im Weiteren der Kürze halber mit $D_d = D$ bezeichnet wird, in jedem Punkt auf dem Lösungspfad so geändert, daß alle Komponenten des skalierten Punktes $\tilde{y} = Dy$ ungefähr gleich groß sind. Dann ist die i-te Komponente der Newton-Korrektur in Falle $|\Delta y_i| \ll |y_i|$ zwar möglicherweise mit einem relativ großen Fehler behaftet; dies ist aber unproblematisch, da sie zur Verbesserung der Lösung aber wenig beiträgt, weil diese Komponente anscheinend schon recht genau ist.

Zur Realisierung der Skalierung wird an Startpunkten y zunächst

$$\hat{D}_0 = diag\left(\frac{1}{|y_1|}, \ldots, \frac{1}{|y_{n+1}|}\right)$$

gesetzt. Nach Berechnung eines neuen Punktes wird dann von der alten Skalierung $D_{\nu-1}$ zu einer neuen Skalierung mittels $\hat{D}_\nu^{-1} = \frac{1}{2}\left(D_{\nu-1}{}^{-1} + \mathrm{diag}(|y_i|)\right)$

gewechselt. Da sich für $y_i \to 0$ und $y_i \to \infty$ numerische Probleme ergeben, werden die folgenden elementweisen Randbedingungen für ${D_\nu}^{-1}$ berücksichtigt:

Falls $\hat{d}_i^{-1} < scal_{min}\|\hat{D}_\nu^{-1}\|$, setze $d_i^{-1} := scal_{min}\|\hat{D}_\nu^{-1}\|$
Falls $\hat{d}_i^{-1} > scal_{max}\|\hat{D}_\nu^{-1}\|$, setze $d_i^{-1} := scal_{max}\|\hat{D}_\nu^{-1}\|$

Dabei wird als Norm für $\nu > 0$ die durch die letzte Skalierung $D_{\nu-1}$ induzierte Norm verwendet; im Fall $\nu = 0$ wird mangels besserer Alternativen die kanonische Norm genommen. Durch diese Art von Randbedingungen bleiben die skalierten Werte unter Transformationen $y \to \alpha y$ invariant.

Bildskalierung Für die Pfadverfolung ist eine Skalierung des Bildraumes nicht notwendig, da der Pfadverfolgungsalgorithmus nirgendwo davon Gebrauch macht, daß der Bildraum normiert ist. Lediglich die Stabilität der Löser der linearen Gleichungssysteme wird beeinflußt. Erst bei der Berechnung von Verzweigungen macht sich die Bildskalierung bemerkbar.

Für die Bildskalierung D_r sind optional drei Möglichkeiten verfügbar:

1. Da die Bildskalierung während der Pfadverfolgung nebensächlich ist, kann bei Problemen, bei denen keine Verzweigungen auftreten, die Bildskalierung weggelassen werden, d.h.

$$D_r := Id$$

2. Falls $f: X \times \mathbb{R} \to X$ als Abbildung des Produktraumes des Variablenraumes X und des Parameterraumes $\mathbb{R}$ in den Variablenraum X aufgefaßt werden kann, iat es sinnvoll, dias Bild genauso zu skalieren wie die Variablen im Urbildraum; nach der Partition

$$D_d = \begin{pmatrix} D_{d_x} & 0 \\ 0 & D_{d_\lambda} \end{pmatrix}$$

wird also gesetzt

$$D_r := D_{d_x}$$

3. Falls die Ableitung nach den Variablen $A_x = \frac{\partial}{\partial x} f$ bezüglich des kanonischen Produkts selbstadjungiert ist, sollte dies auch für die skalierte Jacobi-Matrix gelten; aus der Bedingung

$$\tilde{A}_x^{\ *} = \left(D_r A_x {D_{d_x}}^{-1}\right)^T = {D_{d_x}}^{-1} A_x D_r = \tilde{A}_x = D_r A_x {D_{d_x}}^{-1}$$

folgt dann

$$D_r = {D_{d_x}}^{-1}$$

Bemerkung 4.1: Durch die Änderung der Skalierung nach Berechnung eines Lösungspunktes ist die zugehörige Tangente t nicht mehr normiert. Für die Schrittweitensteuerung ist es aber wichtig, daß die Tangente normiert ist, weil sonst der Prediktor $\hat{y} = y + st$ nahezu beliebig ist. Beim Renormieren der Tangente auf die neue Skalierung muß dann die Schrittweite so geändert werden, daß der Prediktor invariant bleibt; d.h.

$$\text{falls} \quad \|t_\nu\|_{D_\nu} = \tau \neq 1, \qquad \text{ersetze} \qquad t_\nu \to \frac{1}{\tau} t_\nu \quad \text{und} \quad s_\nu \to \tau s_\nu$$

Des weiteren enthält nach dem Wechsel der Skalierung der Schätzer (1.28) für ω_t aus Abschnitt 1.2.5 keine relevante Information mehr, da die verwendeten Normen und Produkte bezüglich der alten Skalierung nicht mit den Werten bezüglich der neuen Skalierung übereinstimmen. Im Tangententest Satz 1.5 läßt sich dieser Schätzer daher nicht mehr verwenden.

4.2 Weitere Details des Algorithmus

Orientierung der Tangente

Nach Berechnung einer Lösung y_ν auf dem Lösungspfad muß die zugehörige Tangente t_ν so orientiert werden, daß sie entlang dem Pfad in dieselbe Richtung zeigt, wie die Tangente des vorherigen Punktes $y_{\nu-1}$. Anderenfalls läuft das Pfadverfolgungsverfahren den Pfad wieder zurück; sollen auch Verzweigungen berechnet werden, wird das Verfahren versuchen, bei einer falsch orientierten Tangente zwischen den beiden Lösungspunkten eine Verzweigung zu berechnen, wenn zwischen den Punkten keine Singularität liegt, da gemäß Bemerkung 2.9 die Determinante das Vorzeichen wechselt. Dies führt zu unnötigen Rechenaufwand.

Liegt tatsächlich eine Verzweigung zwischen den Punkten, so wird das Verfahren bei falsch orientierten Tangente diese nicht entdecken, was zu einem unvollständigem Verzweigungsdiagramm führt.

Um möglichst die richtige Orientierung der Tangente zu erreichen, wird gefordert, daß die Tangente mit der Strecke zwischen den Lösungspunkten

$y_{\nu-1}$ und y_ν einen positiven Winkel einschließt:

$$\langle t_\nu, y_\nu - y_{\nu-1} \rangle \geq 0 \tag{4.3}$$

und daß sie mit der Tangente des letzten Lösungspunktes wenigstens einen gewissen Winkel einschlïeßt:

$$\langle t_\nu, t_{\nu-1} \rangle \geq \text{tangent}_{min} \tag{4.4}$$

wobei in der Regel $\text{tangent}_{min} = 0.0$ gesetzt wird; bei schwierigen Beispielen kann auch tangent_{min} auf größere Werte gesetzt werden, während ein Wert von $\text{tangent}_{min} = -1.0$ diese Test abschaltet.

Können nicht beide Bedingungen (4.3) und (4.4) gleichzeitig erfüllt werden, gilt die Tangente als nicht eindeutig orientierbar; der Lösungspunkt y_ν wird daher verworfen und ein neuer Lösungspunkt mit einer kleineren Schrittweite berechnet.

Berechnung von Randpunkten des Definitionsbereichs

Im allgemeinen kann nicht davon ausgegangen werden, daß die Funktion f auf ganz $\mathbb{R}^{n+1}$ definiert ist; insbesondere der Parameter wird nur in einem Bereich $\lambda \in [\lambda_{min}, \lambda_{max}]$ variieren können. Daher stellt sich das Problem, daß Randwerte des Definitionsbereiches berechnet werden müsen, wenn der Pfad den Definitionsbereich verläßt, indem er z.B. bestimmte Parameterwerte überschreitet. Unter der Annahme, daß $f: Y \subset \mathbb{R}^{n+1} \to \mathbb{R}^n$, wobei der Definitionsbereich von der Form $Y = \prod_{i=1}^{n+1} [y_{i,min}, y_{i,max}]$ ist, läßt sich dieses Problem jedoch einfach lösen.

Sei etwa der Prediktor in der i-ten Komponente von $\hat{y}$ größer als $y_{i,max}$; dann wird zunächst die Schrittweite so reduziert, daß der Prediktor auf der Grenze $\hat{y}_i = y_{i,max}$ liegt; danach wird im Korrektorschritt die Pseudoinverse durch einen Löser ersetzt, der immer $\Delta y_i = 0$ garantiert, so daß die Korrektur auf der Hyperebene $y_i = y_{i,max}$ erfolgt.

Dies ist ohne Änderung der Lineare-Algebra Routinen möglich, indem nach Berechnung von $\widehat{\Delta y} = -A^+ f$ und der Tangente t von A der Schritt

$$\alpha = -\frac{(\widehat{\Delta y})_i}{t_i}, \qquad \Delta y = \widehat{\Delta y} + \alpha t$$

nachgeschaltet wird. Ist $t_i \ll (\widehat{\Delta y})_i$, dann wird $\Delta y \equiv 0$ gesetzt, was zur Beendigung des Korrektorschrittes führt. Denn die Berechnung des Randpunktes

ist unter dieser Bedingung schlecht konditioniert, was die Verwendung des Prediktors als Korrektor rechtfertigt.

Wiederfinden von Verzweigungen

Falls die Pfadverfolgung eine Verzweigung entdeckt, muß getestet werden, ob es dieselbe Verzweigung nicht schon einmal aus einer anderen Richtung berechnet hat. Es ist sinnvoll, diesen Test möglicht frühzeitig durchzuführen, damit nicht unnötige Iterationen zur Berechnung eines Punktes, der schon bekannt ist, durchgeführt werden.

Für diesen Test wird zunächst von dem Startpunkt y_0 aus die erste Newton-Iteration Δy^0 mitsamt der vereinfachten Newton-Iteration $\overline{\Delta y}^1$ berechnet. Überlebt das Verfahren dann den Monotonietest $\Theta_0 = \|\overline{\Delta y}^1\|/\|\Delta y^0\| \leq \overline{\Theta} < 1$, läßt sich aus Teil c) von Satz 1.2 oder einfacher aus der linearen Konvergenzannahme $\Theta_0 \geq \|\overline{\Delta y}^{i+1}\|/\|\overline{\Delta y}^i\|$ für das vereinfachte Newtonverfahren, das hier wegen der lokalen Eindeutigkeit gegen denselben Punkt konvergiert, die Abschätzung

$$\|y^0 - y^\star\| \leq \frac{1}{1-\Theta_0}\|\Delta y^0\| \tag{4.5}$$

für die Entfernung zum Konvergenzpunkt zeigen. Befindet sich nun ein bereits berechneter Verzweigungspunkt in einer kleineren Entfernung von Startpunkt, so wird er als Lösung akzeptiert, wenn noch ein abzweigenden Ast mit einer akzeptablen Tangente frei ist. Dabei wird eine Tangente akzeptiert, falls sie einen kleineren Winkel mit der Tangente des letzten Punktes einschließt als die Tangenten der beiden Punkte, zwischen denen die Verzweigung entdeckt wurde.

4.3 Kurzbeschreibung des verwendeten Sparse-Lösers

Zur Berechnung einer LU-Zerlegung unter Berücksichtigung der Belegtheitsstruktur wurde der Sparse-Löser `MA28` benutzt, der über die NetLib als Fortran-Code verfügbar ist. Der Code wurde unter Einfügung der zur Berechnung der Pseudoinversen nötigen Zusatzroutinen in C übersetzt, was aufgrund des für Fortran sehr übersichtlichen Programmierstils und der guten Dokumentation in [15] und den Kommentaren im Code relativ leicht zu bewerkstelligen war.

(Eine automatische „Übersetzung“ oder ein Linken der Maschinencodes an die restlichen, in C geschriebenen Routinen war wegen der nötigen Modifikationen im Code ungünstiger.) Das Auswahlkriterium zugunsten des `MA28` war einfach: Es ist der einzige Code für allgemeine dünnbesetzte Matrizen, der auch Gleichungssysteme lösen kann, wenn die rechte Seite vor der Zerlegung nicht bekannt ist, wie es hier z.B. beim Berechnen der vereinfachten Newton-Korrekturen der Fall ist. Alle anderen Sparse-Löser berechnen den L-Teil der Zerlegung nicht explizit, sondern transformieren die rechte Seite beim Eliminationsprozeß mit.

Es folgt nun eine kurze Beschreibung des Algorithmus des Sparse-Lösers `MA28`; eine ausführlichere Diskussion der Details läßt sich in [15] finden.

Zunächst zu den Datenstrukturen, in denen dünn besetzte Matrizen abgespeichert werden. Für die Kommunikation nach außen ist beim `MA28` eine einfach zu belegende Datenstruktur vorgesehen; sie besteht in einem eindimensionalen Vektor $a[k]$ mit Indexbereich $k = 1 \ldots nz$, der die Werte der Einträge einer Matrix A enthält, und zwei Index-Vektoren $rowIndex[k]$, $colIndex[k]$, $k = 1 \ldots nz$, die die Zeilen- und Spaltenindizes der jeweiligen Einträge enthält; der Wert a_{ij} ist dabei in denjenigem $a[k]$ gespeichert, für das $rowIndex[k] = i$ und $colIndex[k] = j$ ist. Dabei ist nz die maximale Anzahl von Nicht-Null Elementen, die in der Struktur abgespeichert werden können.

Existieren mehrere Indexpaare $(rowIndex[k], colIndex[k]) = (i, j)$, besteht der Wert des zugehörigen Eintrags in der Summe der ugehörigen $a[k]$. Gelöschte Einträge werden durch $colIndex[k] = 0$ gekennzeichnet.

Intern werden die Daten zunächst umsortiert, so daß sie zeilenweise gepackt vorliegen; die zugehörige Datenstruktur lautet:

$$\begin{aligned} a[k] \quad & k = 1 \ldots nz \\ colIndex[k] \quad & k = 1 \ldots nz \\ rowPtr[i] \quad & i = 1 \ldots n+1 \\ nzRow[i] \quad & i = 1 \ldots n \end{aligned} \tag{4.6}$$

Dabei fängt die Zeile i beim Index $k = rowPtr[i]$ an und geht bis $k = rowPtr[i] + nzRow[i] - 1$, $a[k]$ enthält die Werte und $colIndex[k]$ die Spaltenindizes der Einträge. $rowPtr[n+1]$ enthält den Index hintern dem letzten Nicht-Null Element in a.

Falls die Werte in a, $colIndex$ ohne Lücke aufeinanderfolgen, ist eine der Datenstrukturen $rowPtr$, $nzRow$ im Prinzip überflüssig; da aber im Laufe der Elimination Lücken in die Struktur gerissen werden, sind während der Eliminationsphase beide erforderlich.

Um die Elimination durchzuführen, sind noch einige Hilfsstrukturen notwendig. Zum einen werden die Zeilen des L-Teils und des U-Teils der Zerlegung zusammen gepackt abgespeichert. Dazu wird ein Vektor $nzLRow[i]$, $i = 1 \ldots n$ eingeführt; die ersten $nzLRow[i]$ Elemente einer Zeile i bilden dann den L-Teil (ohne die Einsen auf der Diagonale), der Rest den U-Teil.

Des weiteren wird eine spaltenweise gepackte Struktur der Nicht-Null-Belegung angelegt:

$$\begin{array}{ll} rowIndex[k] & k = 1 \ldots nz \\ colPtr[i] & i = 1 \ldots n+1 \\ nzCol[i] & i = 1 \ldots n \end{array}$$

die nur die Indizes der Nicht-Null-Elemente enthält, die während der Elimination in $A^{(k)}$ enthalten sind.

Für die schnelle Pivot-Suche werden doppelt verkettete Listen von Indizes angelegt, deren Zeilen bzw. Spalten dieselbe Anzahl von Nicht-Null-Elementen enthalten; die Listen sind in fünf Index-Vektoren:

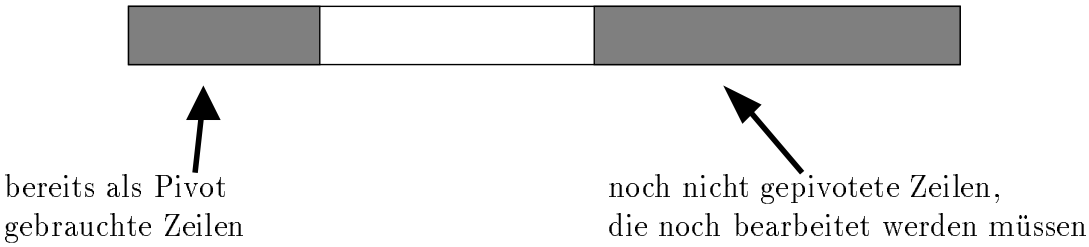
$$\left.\begin{array}{l} First[i] \\ NextRow[i] \\ LastRow[i] \\ NextCol[i] \\ LastCol[i] \end{array}\right\} \quad i = 1 \ldots n$$

abgespeichert. Dabei enthält $j = First[i]$ einen Index für eine Zeile mit i Nicht-Null-Elementen, falls $j > 0$ oder einen Index, so daß die $-j$-te Spalte i Nicht-Null-Elemente hat, falls $j < 0$. $nextRow/Col[|j|]$ enthalten dann den Nachfolger und $lastRow/Col[|j|]$ den Vorgänger in der Liste der Zeilen/Spalten mit i Nicht-Null-Elementen. Eine „0“ bezeichnet das Ende einer Liste.

Außerdem werden noch Index-Vektoren benötigt, in denen die Pivotstrategie abgespeichert wird.

Der `MA28` braucht zur Berechnung der Zerlegung mehr Speicher, als nur die Nicht-Null Elemente von A belegen. An Anfang der Elimination werden die Nicht-Null Elemente von A in das Ende des Indexbereiches von a

und $colIndex$ geschoben; während eines Eliminationsschrittes werden dann die Zeilen, die das Pivotelement enthalten, in den vorderen Teil des Speichers geschoben. Der Speicher sieht also zwischendurch so aus:



Am Ende der Elimination befindet sich die gesamte zerlegte Matrix wieder in gepackter Form am Anfang des Speichers.

Ein Eliminationschritt besteht dabei aus zwei vom Aufwand her etwa gleich großen Hauptschritten: zuerst wird ein geeignetes Pivotelement gesucht und dann mit diesem Pivot die Elimination durchgeführt.

Bei der Pivotsuche wird die Pivotstrategie von Markowitz angewand. Dabei wird das Element a_{ij} als Pivot akzeptiert, für das

- die zugehörige Anzahl der Operationen im Eliminationsschritt, die sich als das Produkt $(nzRow[i] - 1) * (nzCol[j] - 1)$ berechnet und eine obere Schranke für das Fill-In, d.h. die in diesem Schritt entstehenden zusätzlichen Nicht-Null-Elemente, bildet, minimal ist

- und zusätzlich die Stabilitätsbedingung

$$|a_{ij}| > u * \max\{|a_{ik}| \mid a_{ik} \text{ ist in derselben Zeile wie } a_{ij}\} \tag{4.7}$$

  sowie, falls a_{ij} das einzige Element in der Zeile ist, dann auch

$$|a_{ij}| > u * \max\{|a_{kj}| \mid a_{kj} \text{ ist in derselben Spalte wie } a_{ij}\}$$

  erfüllt.

Der Parameter $0 \leq u < 1$ gewichtet dabei den Kompromiß zwischen Erhalt der dünnen Belegtheitsstruktur und der Stabilität. Bei Werten von u nahe 1 tendiert die Pivotsuche zur normalen Spaltenpivotsuche und ist entsprechend stabil; für $u = 0$ findet die Pivotsuche nur nach dem Kriterium der Belegtheitsstruktur statt. Duff empfiehlt in [15] aus der Erfahrung heraus

Werte von $0.1 \leq u \leq 0.25$ als guten Kompromiß; alle Beispiele in Kapitel 5 sind mit $u = 0.1$ berechnet.

Damit nicht der gesamte Teil von $A^{(k)}$ nach dem Pivot durchsucht werden muß, werden die verketteten Listen eingesetzt; dabei werden zunächst die Spalten und die Zeilen in der mit $First[1]$ startenden Liste nach dem günstigste Pivotkandidat durchsucht, danach die in $First[2]$ usw. Die Suche wird bei $First[n_{Such}]$ abgebrochen, falls der laufende Pivotkandidat weniger als $(n_{Such} - 1)^2$ Operationen benötigt, da alle in den folgenden Spalten und Zeilen stehenden Elemente entweder eine größere Anzahl von Operationen verursachen würden oder bereits durchsucht worden sind.

Nachdem das Pivot gefunden worden ist, wird die Pivotzeile in den vorderen Teil des Speichers kopiert. Über die spaltenweise gepackte Struktur ist es leicht herauszufinden, in welchen Zeilen eliminiert werden muß, da sie in der Pivotspalte ein Nicht-Null Element besitzen.

Für jede solche Zeile i wird zunächst das zu eliminierende Element a_{ij} an die Spitze des U-Teils getauscht und durch den Multiplikator a_{ij}/a_{pivot} ersetzt; sodann wird $nzLRow[i]$ um eins erhöht, so daß dieses Element nun dem L-Teil der Zeile angehört. Danach wird für jedes Element in U-Teil der Nichtpivot-Zeile, für das sich in der Pivot-Zeile ein Element mit demselben Spaltenindex befindet, die entsprechende Elimination durchgeführt und das zugehörige Element in der Pivotzeile markiert. Sind danach nicht alle Elemente der Pivotzeile markiert, bewirken die nicht markierten Elemente neue Nicht-Null Elemente in der Nichtpivotzeile, also Fill-In. Falls sowohl unmittelbar hinter als auch unmittelbar vor der Zeile kein Platz für das Fill-In ist, wird die Zeile an den Anfang des noch nicht bearbeiteten Teils von A kopiert. Ist dort nicht genügend Platz, so daß die Zeile in die bearbeiteten Zeilen von A hineinragen würde, so wird der Speicherbereich, der von dem noch nicht bearbeiteten Teil belegt wird, durch eine Garbage-Kollektion komprimiert, indem alle durch Kopieren von Zeilen in den vorderen Teil entstandenen Lücken geschlossen werden.

Ist dann noch nicht genügend Platz vorhanden, muß die Zerlegung abgebrochen werdern. Im anderen Fall können die noch ausstehenden Eliminationsschritte für diese Zeile durch Absuchen der Pivotzeile nach unmarkierten Elementen berechnet werden.

Während der Elimination müssen an den angebrachten Stellen auch die Datenstrukturen der spaltenweise gepackten Belegtheitsstruktur und der verketteten Listen entsprechend den Veränderungen durch die Elimination an-

gepaßt werden. Dies wurde hier aber zur Bewahrung der Übersichtlichkeit weggelassen.

Falls der `MA28` eine Matrix mit derselben Nicht-Null Verteilung schon einmal zerlegt hat, wie es im Rahmen der Pfadverfolgung gegeben ist, kann er optional auf die aufwendige Pivotsuche verzichten und auf die alte Pivotstrategie zurückgreifen. Dazu muß neben den Vektoren, die die Pivotstrategie speichern, auch noch die Datenstruktur

$$\begin{array}{l} colIndex \\ nzRow \\ nzLRow \end{array}$$

von der alten Zerlegung aufbewahrt werden.

Zur Elimination wird dann die Matrix gemäß der alten Pivotstrategie permutiert und in der Datenstruktur $a/colIndex/nzRow$ abgespeichert; danach kann die Elimination ohne zusätzliche Speicherbewegungen durchgeführt werden, da der Platz für das Fill-In schon aus der vorherigen Elimination bereitsteht. Diese Option ist erheblich schneller als die Zerlegung mit neuer Pivotsuche (bei einem Testbeispiel der Dimension 5000 entstand ein Unterschied von fünf Minuten mit alter gegenüber zwei Stunden mit neuer Pivotstrategie), birgt aber auch die Gefahr der Instabilität mit sich, da keinerlei Stabilitätsbedingungen die Zerlegung beeinflussen.

Daher sind im `MA28` Stabilitätsindikatoren eingebaut, die nach der Elimination Aussagen über die Stabilität der Zerlegung machen. Zum einen wird das maximale u berechnet, für das die Pivotstrategie gerade noch die Stabilitätsbedingung (4.7) erfüllt; zum anderen wird aus [16] eine Abschätzung der bezüglich der berechneten Faktoren $\tilde{L}$ und $\tilde{U}$ verursachten Fehlers $\tilde{L}\tilde{U} = A+E$ übernommen, nämlich daß sich E elementweise durch $\varepsilon\mathcal{O}(\rho)$ abschätzen läßt, wobei ρ das maximale während der Elimination aufgetretene Element bezeichnet.

In [24] werden diese Stabilitätsindikatoren untersucht und festgestellt, daß sie häufig zu pessimistisch sind; statt dessen wird empfohlen, die Stabilität der Zerlegung durch Nachiteration zu überprüfen. Konvergiert für ein Gleichungssystem die Nachiteration nicht oder zu langsam, wird die alte Pivotstrategie verworfen und eine neue Zerlegung mit neuer Pivotstrategie durchgeführt. Außerdem wird optional eine neue Pivotstrategie entwickelt, wenn der gewünschte Rang der Zerlegung wechselt, d.h. im Kontext der Pfadverfolgung an Verzweigungen.

Zur Berechnung der Nachiteration muß allerdings neben der Zerlegung auch die ursprüngliche Matrix abgespeichert werden; da aber die Zerlegung zwischen dem 3-fachen und 30-fachen des Speicherplatzes der ursprünglichen Matrix belegt je nach Größe und Verteilung der Nicht-Null-Elemente, ist dieser Mehrbedarf an Speicher erträglich.

4.4 Objekt-orientierte Implementierung in C

Die Implementierung des Pfadverfolgungsalgorithmus erfolgte im Programm `Alcon-S` in objekt-orientiertem Programmierstil, um eine größt mögliche Unabhängigkeit der einzelnen funktionalen Teile zu gewährleisten und den Aufwand bei Änderungen im Programm möglichst gering zu halten. Die Implementierung lehnte sich dabei an [25] an, wodurch auch Teile des Programmcodes von `Percon` übernommen werden konnten; andere Teile stammen von dem Programm `Symcon`, auf das die Entwicklung von `Alcon-S` aufbaute, wobei aber die Symmetrieausnutzung aus `Symcon` nicht übernommen wurde. Da die Programmierung in C erfolgte, das objekt-orientierte Programmierung nur bedingt unterstützt, mußten einige zusätzliche Überlegungen zur Organisierung der Objekte gemacht werden.

Die einfachsten Datenstrukturen sind in klassischer Trennung Struktur und darauf operierende Prozeduren gehalten, da den Vorteilen der objekt-orientierten Programmierung hier der Nachteil des umständlichen Programmierstils gegenüberstand. Als primitive Datenstrukturen sind definiert:

```
typedef enum {false, true} Bool;
typedef long Int;
typedef double Real;
typedef void* Generic;
```

dabei ist der Index-Typ `Int` deshalb als `long` implementiert, weil für die Indizierung der Nicht-Null-Elemente einer Matrix im Sparse-Format bisweilen mehr als 2^{15} Indizes benötigt werden.

Weiterhin sind die einfachen Datenstrukturen der Vektors, Indexvektors, (voller) Matrix und Sparse-Matrix:

```
typedef Real*    RealVec;
typedef Int*     IntVec;
```

```
typedef RealVec* RealMat;
typedef struct {
   Int nz, nzMax;
   RealVec a;
   IntVec  rowIndex,colIndex;
}              *SparseMat;
```

implementiert und mit einer Reihe von Hilfsroutinen (Kopieren, Vektoraddition, Matrix-Vektor-Multiplikation, Umwandlung zwischen `RealMat` und `SparseMat`, ...) versehen. Außerdem ist ein Typ `List` implementiert, der zur Verwaltung von Daten in einer doppelt verketteten Liste dient.

Die im objekt-orientierten Stil gehaltenen Datenstrukturen sind in drei `struct`'s enthalten, die jeweils getrennt die Methoden, die öffentlichen und die privaten Daten enthalten. Für ein typisches Objekt `XY` sind die *öffentlichen Daten* in der Struktur:

```
typedef struct {
   Int someInterestingData;
           .
           .
           .
   XYType type;
   Generic data;
}                    *XYData;
```

enthalten; `XYData->type` enthält dabei den Typ des Objekts, für ein Objekt der Linearen Algebra also zum Beispiel den Typ der Zerlegung. `XYData->data` ist ein Pointer auf die *privaten Daten* der Objekts; da diese für verschiedene Typen von unterschiedlicher Struktur sein können und von außen nicht sichtbar sein sollen, werden sie durch einen generischen Pointer referenziert.

Die *Methoden* des Objekts sind in einer Struktur

```
typedef struct {
   void (*MakeThis) (XYData , ...),
        (*MakeThat) (XYData , ...);
   Real (*SomeValue)(XYData , ...);
```

```
            .
            .
            .
    XYData* data;
}                       *XY;
```

zusammengefaßt; dabei ist `XY->data` ein Pointer auf die öffentlichen Daten des Objekts; alle Methoden eines Objekts müssen in der Form `XY->Methode( XY->data, ...)` aufgerufen werden, da die Methoden des Objekts sonst ihre eigenen Daten nicht kennen.

Zu dem Objekt gehören noch als Konstruktoren und Destruktoren die global gehaltenen Prozeduren

`XYData NewXYData(void)`	konstruiert `XYData`
`XY NewXY(XYData data)`	konstruiert `XY` und die privaten Daten gemäß `data->type`
`void ResetXY(XY xy)`	paßt `XY` und die privaten Daten nach Veränderung von `XYdata` den öffentlichen Daten an
`XYData Clear(XY xy)`	Destruktor für die Methoden und privaten Variablen; gibt die öffentlichen Daten `xy->data` zurück
`void QuitXYData(XYData data)`	Destruktor für die öffentlichen Daten.

Zum Konstruieren eines Objekts werden zuerst mit `NewXYData` die öffentlichen Daten, initialisiert mit irgendwelchen neutralen Werten, geschaffen und diese von aufrufenden Programmteil mit gewünschten Werten belegt; insbesondere muß der Typ spezifiziert werden. Mit den so belegten Daten wird dann über `NewXY` das eigentliche Objekt konstruiert. Beim Destruieren wird der umgekehrte Weg über `ClearXY` und `QuitXYData` gegangen. Soll der Typ des Objekts geändert werden, müssen zuerst die alten Methoden mit `ClearXY` verabschiedet werden; dann kann in den Daten der Typ gesetzt und neue Methoden mit `NewXY` konstruiert werden.

Von diesen Schema gibt es eine Reihe von Variationen. Falls die privaten Daten nicht variieren, können sie einfachheitshalber in die öffentlichen Variablen integriert werden; Methoden, die nicht von Typ abhängen, können global gehalten werden; falls alle Methoden des Objekts global sind, fällt die Struktur `XY` mitsamt den Kon-/Destruktoren weg.

Das einfachste Objekt ist

```
typedef enum {eEmpty, eFull, eSparse, eEvalOnly} LinOpType;
typedef struct {
  LinOpType type;
  Bool symmetric;
  union {
    RealMat A;
    SparseMat S;
    Bool (*f)(RealVec x, RealVec b, Generic extra,
                                        Bool adjoint);
   } d;
   Int m,n;
   Generic extra;
} *LinOp;
```

mit den globalen Methoden

```
Bool LinOpEval(LinOp l, RealVec x, RealVec b);
Bool LinOpTransEval(LinOp l, RealVec x, RealVec b);
```

welches die einheitliche Auswertung eines (m, n)-Operators ermöglicht. Neben den Typen `eFull` für einen als `RealMat` und `eSparse` für einen in `SparseMat` abgespeicherten Operator sind noch die Datentypen `eEmpty` für einen nicht initalisierten Operator und `eEvalOnly` für einen Operator, für den nur Auswertungen der Form Ax und $A^T x$ möglich sind, gegeben. Beispiel für einen solchen Operator ist etwa die Matrix $C_{11} = \bar{V}_1^* C \bar{V}_1$ in Mooreschen System (3.19), die nicht explizit berechnet wird; der Datentyp ist auch bei der Verwendung von iterativen Lösern sinnvoll.

Die Skalierung ist in dem Objekt

```
typedef enum {eEmptyProd, eStandard, eNormalized} EuclidType;

typedef struct {
  EuclidType type;
  Int n;
  RealVec weight;
} *EuclidData;
```

```
typedef Real (*Scalarfunction)(EuclidData d, RealVec x,
                               RealVec y);
typedef Real (*Form)(EuclidData d, RealVec x);
typedef void (*Operator)(EuclidData d, RealVec x);

typedef struct {
  Scalarfunction Product, Cosinus, Distance, SquaredDistance;
  Form Norm, SquaredNorm;
  Operator Scale, Descale, Normalize;
  EuclidData data;
} *Euclid;
```

untergebracht, wobei das private Datum `weight`, welches den Skalierungsvektor enthält, einfachheitshalber in die öffentlichen Daten integriert wurde. Die Operatoren `Scale` und `Descale` berechnen Dx und $D^{-1}x$; die anderen Methoden tun das, was ihr Name andeutet. Neben dem Typ `eStandard` ist auch der Typ `eNormalized` implementiert, der das innere Produkt $\langle x, y\rangle_D = x^T D^2 y$ noch durch die Anzahl der Variablen teilt. Dies führt bei großen Systemen zu Normen, die in derselben Größenordnung wie die Einträge der jeweilgen Vektoren liegen.

Für dieses Objekt stehen neben den Methoden `InitScale` und `ChangeScale`, die die Änderung des Skalierungsvektors besorgen, noch die Hilfsroutinen `Orthogonalize`, `Project` und `OrthoProject` bereit, die die Spalten eine Matrix bezüglich des Objektes orthogonalisieren und den Projektor auf ihr Bild bzw. dessen orthogonales Komplement berechnen.

Das umfangreichste Objekt ist das für die lineare Algebra:

```
typedef enum {eNoSolver, eLUMat, eQRMat, eMA28Mat,
              eBlockMat, eMoore} LinAlgType;

typedef struct {
/*
 *    this data is to be set by the user
 */
  Int n;           /* dimension of domain */
  Int m;           /* dimension of range */
```

```
  LinOp T;         /* the linear operator of the equation */
  LinAlgType type; /* the type of the solver */
/*
 *    this data can be set by the user,
 *    otherwise it is set to some resonable standart values
 */
  Euclid domainScale; /* just what it says */
  Euclid imageScale;
  Bool domainOwner, imageOwner; /* do not touch;
        true, if domain/imageScale is allocated by NewLinAlg,
        in this case do not quit this methods by your own */
  Real eps;       /* accuracy of the given input data */
  Bool fullRank;  /* if true, then the linear operator is
                   treated as exactly of rank p and perturbed
                   to a nearby operator of this rank,
                   if necessary */
  Bool iteration; /* Solve tries iterative refinement,
                   if iteration is true and the solver is a
                   direct solver */
  Bool pm,pw; /* print messages and/or warnings */
/*
 *    while this is read only
 */
  Int p;      /* pseudorank */
  Int mp,np;  /* dimensions of regular space */
  RealMat S;  /* the neglected part of the decomposition */
  Real test;  /* test function; changes sign
               if orientation changes */
  Real cond;  /* condition estimate; the output error can
               expected to be of order of eps*cond */
  Real norm;  /* estimation of the norm of the
               regular part of A */
  Bool subOp; /* is true if the LinAlg is part
               of a bigger LinAlg */
  Generic data; /* data used by the solver */
} *LinAlgData;
```

```
typedef Bool (*Procedure)(LinAlgData data);
typedef Bool (*Computation)(LinAlgData data, RealVec b,
                            RealVec x);
typedef void (*Transformation)(LinAlgData data, RealVec b,
                               RealVec x);
typedef Bool (*TransSymMat)(LinAlgData data, LinOp c,
                        LinOp c11, RealMat c12, RealMat c22);
typedef void (*GiveData)(LinAlgData data, RealMat t);
typedef void (*IntSet)(LinAlgData data, Int );
typedef void (*MoveData)(LinAlgData data, Generic *place);
typedef eFileError (*FileIO)(File f, LinAlgData data);

typedef struct {
  Procedure Prepare;
  IntSet NewRank;
  Computation Solve,      /* x = A^+ b */
              SolveAd;    /* x = A^*+ b */
  GiveData KernelBasis,   /* T: A T == 0 */
           KernelAdBasis; /* T: A^* T == 0 */

  Transformation
    SolveRegular,    /* u = A-1 c */
    SolveAdRegular,  /* c = A-* u */
    ProjectRange,    /* c = Tr^-1 *Pr*b */
    EmbedRange,      /* b = Pr*Tr^-* *c */
    EmbedDomain,     /* x = Pd*Td^-* *u */
    ProjectDomain;   /* u = Td^-1 *Pd*x */
  Transformation     /* for use with block solver */
    TransformOnRange,             /* c = W1*b */
    TransformOnRangeCompl,        /* c = W2*b */
    TransformFromRangeOrthoCompl, /* b = W1 c */
    TransformFromRangeOrtho,      /* b = W2 c */
    TransformFromDomainOrthoCompl,/* x = V1_u */
    TransformFromDomainOrtho,     /* x = V2_u */
  /* for use with the system of moore */
    TransformOnDomain,            /* u = V1_*x */
    TransformOnDomainCompl;       /* u = V2_*x */
```

```
    TransSymMat  TransformDomainSymMat;

    Procedure SolverSetUp;
    MoveData  ResetSolver;
    MoveData  SaveData, LoadData;
    Procedure ForgetHistory;
    FileIO    SaveToFile, LoadFromFile;

    LinAlgData data;
  } *LinAlg;
```

Die Programmkommentare erklären den meisten Teil hoffentlich.

Zur `LinAlgData` ist zu kommentieren, daß das Flag `fullRank` für den Typ `eBlock` gesetzt werden muß, wenn der Algorithmus nach Govaerts und Pryce berechnet werden soll. Das Flag `subOp` beeinflußt den Rangentscheid; ist es gesetzt, wird bei Beginn der Zerlegung angenommen, daß `norm` die Norm der restlichen Teil des Operators enthält, von den das aktuelle `LinAlg` einen Teil zerlegt; ebenso enthält die `test`-Funktion die Determinante des regulären Teils und nicht Null im Fall einer rangdefekten Matrix. Die Flag wird nur von dem Block-Löser aus Abschnitt 3.3.5 für die von ihm erzeugten Unter-`LinAlg`'s gesetzt.

Die Variable `np` enthält die Länge eines Vektors, der Bild der Methoden `ProjectDomain` und `TransformOnDomain` ist. Dies ist z.B. im Fall der Deflation nicht p, sondern n. Der Bildvektor der Methode `TransformOnDomainCompl` wird hingegen immer als von der Länge $n - p$ angesehen.

Bei den Methoden berechnet `Prepare` für direkte Löser die Zerlegung; für der Fall eines iterativen Lösers wäre diese Methode also leer. `Solve` und `SolveAd` berechnen die Pseudoinverse von A und A^*. `KernelBasis` und `KernelAdBasis` berechnen eine orthogonale Basis der entsprechenden Nullräume. Die folgenden Methoden zerlegen den Lösungsvorgang in einzelne Schritte und sind nur für den Block-Löser und das Mooresche System von Belang. Die Methode `TransformDomainSymMat` berechnet die Teilmatrizen von $\begin{pmatrix} C_{11} & C_{12} \\ C_{12}^* & C_{22} \end{pmatrix} = T_D{}^{-*} C T_D{}^{-1} = \begin{pmatrix} \bar{V}_1{}^* C \bar{V}_1 & \bar{V}_1{}^* C \bar{V}_2 \\ \bar{V}_2{}^* C \bar{V}_1 & \bar{V}_2{}^* C \bar{V}_2 \end{pmatrix}$ explizit, falls dies einfach möglich ist; anderenfalls kann sie ohne Schaden auf `nil` gesetzt werden.

Das `SolverSetUp` enthält ein interaktives Menu für die privaten Daten. `SaveData` speichert für die Belegung wichtige Daten für einen späteren Ge-

brauch ab. Dies ist für die Berechnung an Verzweigungen gedacht; wird die Pfadverfolgung zu einem späteren Zeitpunkt wieder von der Verzweigung gestartet, nachdem ein entfernterer Teil des Verzweigungsdiagramms berechnet wurde, können diese Daten mit `LoadData` wieder geladen werden. Beim `MA28` ist dies die Pivotstrategie; bei einem iterativen Löser könnte dies z.B. ein Vorkonditionierer, der sich dem Pfad entlang ändert, sein. `ForgetHistory` dient zum Löschen dieser Daten, falls sie nicht verwendbar sind, etwa die Pfadverfolgung mit ihnen zusammenbricht. `SaveToFile` und `LoadFromFile` dienen dem Abspeichern bzw. Laden der Flags in einer Datei.

Die privaten Daten z.B. des `MA28` sehen so aus:

```
typedef struct {
  SparseMat A;
  IntVec colIndex;
  IntVec rowPtr,nzRow,nzLRow;
  IntVec colPtr,nzCol;
  IntVec first,lastRow,nextRow,lastCol,nextCol;
  IntVec ipivot, dpivot;
  Int signum;     /* of the permutations */
  Int n, luBegin, aBegin, dp;
  Real u;
  Real stability,placeFactor;
  Bool transposed,remove,deflate,tryOld,useOld,switchAtRank;
  Bool lbig, stable;
  RealMat W, V;
  RealMat phi,psi;
  RealMat kernel, kernelAd;
  LinAlg lin;
} *MA28Mat;
```

Neben den aus Abschnitt bekannten Datenstrukturen (der neben dem `A->colIndex` zusätzliche `colIndex` enthält die Indizes einer alten Zerlegung) und den Matrizen `W,V` aus der Methode 3.3.3 und den Matrizen `phi,psi` aus der Methode 3.3.4, deren Dimension `dp` ist, enthält die Struktur noch verschiedene Flags. Diese bedeuten:

Flag:	Bedeutung, wenn gesetzt:
`transposed`	berechne Zerlegung von A^T anstelle von A
`remove`	setze Elemente kleiner als $\varepsilon\|A\|$ gleich Null
`deflate`	berechne Rangdefekt über Deflationsalgorithmus, anderenfalls durch Weglassen kleiner Pivots
`tryOld`	benutze alte Pivotstrategie, falls möglich
`useOld`	alte Pivotstrategie ist brauchbar
`switchAtRank`	berechne neue Pivotstrategie, falls der Rang wechselt
`lbig`	berechne maximales Element während der Zerlegung, anstelle es nacher abzuschätzen
`stable`	berechne Stabilitätsindikator in `stability`

Daneben ist ein Löser auf der Basis einer QR-Zerlegung aus dem Programm `Percon` und eine normale LU-Zerlegung sowie der Block-Löser, der alle drei Löser als Löser von A_x akzeptiert, implementiert. Der Löser für das Mooresche System wurde auch in das Schema gezwängt; allerdings sind alle außer der `Prepare/Solve`-Methoden nicht gesetzt.

Die Struktur für das Gauß-Newton-Verfahren wurde aus `Percon` übernommen; die öffentlichen und privaten Daten sind zusammen in der Struktur `GNProblem`:

```
typedef enum GNMODE {
  eFullJacobian, eBroyden, eSimplified, eKeepJacobian
} GNMode;
typedef enum GNERROR {
  eNoGNError, eGNFunctionFailed, eGNJacobianFailed,
  eGNSolveFailed, eGNMonoTestFailed, eGNTooManySteps
} GNError;
typedef struct {
  Int m, n;
  GNMode mode;
  Bool global, exact;
  Real tol;
  RealVec xStart;
/*
 *  functions defining the problem
 */
  Bool (*f)(RealVec x, Real tol, Generic data, RealVec fx);
```

```
    Bool (*df)(RealVec x, Real tol, Generic data, LinOp dfx);
    Bool (*Jacobian)(RealVec x, Real tol, Generic data);
    Bool (*Solve)(RealVec b, Real tol,Generic data, RealVec x);
    void (*Kernel)(Generic data, RealMat t);
  /*
   *  corresponding data
   */
    Generic fData;
    Generic dfData;
    Generic JacobianData;
    Generic SolveData;
    Generic KernelData;
  /*
   *  functions for scaling
   */
    ScaleMode scaleMode;
    Real (*ScaledNorm)(RealVec x, Generic data);
    void (*Rescale)(RealVec x, Generic data);
    Generic ScaledNormData;
    Generic RescaleData;
  /*
   *  output data
   */
    GNError error;
    Real snc, nc, theta, convergenceRadius;
    Int nIter;
    Bool pm, pw;
  /*
   *  the rest should be hidden, but ...
   */
    RealVec x, xLast, dx, dxLast, dxb, dxbLast, b;
    RealMat dxSave;
    Real conv;
    Int lastJacobian;
  } *GNProblem;
```

enthalten; es stehen die globalen Methoden

```
Bool GNStart(GNProblem gnProblem, RealVec xOut);
Bool GNContinue(GNProblem gnProblem);
Bool GNSolve(GNProblem gnProblem, RealVec xOut);
```

zur Verfügung, wobei die ersten beiden Prozeduren den Lösungsvorgang der dritten Prozedur in den ersten Iterationsschritt und die restlichen Schritte zerlegt. Dies ist sinnvoll wenn nach den ersten Schritt noch zusätzliche Test durchgeführt werden sollen, siehe etwa den Abschnitt „Wiederfinden von Verzweigungen“ in 4.2.

Das Verzweigungsdiagramm ist in der Struktur `Net`:

```
typedef enum {
  eUnknown, eRegular, eTurning, eBifurcation, eBoundary,
  eDead, eHanging, eStart
} PointType;
typedef struct {
  PointType type;
  Int n, nIter;
  RealVec y, scale;
  Generic data;
  Real nc, theta, cond, curve, dist, test;
} *Point;
typedef struct {
  Point p0, p1;
  RealVec t0, t1;
  Real s;
} *Edge;
typedef struct {
  char name[40];
  List edges,ghostEdges,
       points, bifPoints, turnPoints, boundPoints,deadPoints;
} *Net;
```

abgespeichert. Dabei enthält jeder `Point` neben den Koordinaten `y` noch Informationen über den Konvergenzprozeß und eventuell von dem `LinAlg` abgespeicherte Daten (in `data`). Eine `Edge` verbindet zwei Punkte auf dem Pfad mittels in Pfadverfolgungsrichtung orientierten Tangenten. Das `Net` enthält

neben einer Liste aller Punkte noch Listen von Punkten von speziellem Typ; die zwei verschiedenen Listen von Kanten entstehen durch den Pfadverfolgungsalgorithmus.

Nach Berechnung eines regulären Punktes wird eine Kante angelegt, deren Punkt `p0` der eben berechnete Punkt ist. In den Punkt `p1`, der von Typ `eHanging` ist, wird der tangentiale Prediktor geschrieben. Eine solche Kante kommt in die Liste der `ghostEdges`, da eines ihrer Enden in der Gegend herumgeistert. Der Korrektor arbeitet diese Liste von Geisterecken ab; gelingt es, für den Punkt `p1` eine Lösung auf dem Pfad zu berechnen, so wird die Kante von den `ghostEdges` zu den fertigen `edges` geschoben.

Falls der berechnete Punkt eine Verzweigung ist, werden den abzweigenden Ästen ensprechend mehr Kanten in der Liste der `ghostEdges` angelegt.

Globale Methoden stehen bereit zum Einfügen, Löschen und Suchen von Punkten und Ecken in die jeweiligen Listen und zur Ausgabe insbesondere von graphischer Information über Punkte und Ecken sowie des ganzen Netzes.

Die Struktur

```
typedef Bool (* Func)(RealVec y, Real tol, Generic data,
                      RealVec b);
typedef Bool (* Deriv)(RealVec y, Real tol, Generic data,
                       LinOp a);
typedef Bool (* Augment)(RealVec y, RealVec z, Real tol,
                         Generic data, LinOp c);
typedef struct AP {
  char* name;
  Func f;
  Deriv df;
  Augment cf;
  Generic fData;
  Generic dfData;
  Generic cfData;
  LinOpType type, ctype;
  Int m,n,nz,nzc;
  Real tol;
  RealVec yGuess, tauMin, tauMax, parameter;
  void (*ProblemMenue)(struct AP *p);
  void (*ResetParameter)(struct AP *p);
```

```
    void (*ResetInitalGuess)(struct AP *p);
    void (*ResetProblem)(struct AP *p);
    void (*ClearProblem)(struct AP *p);
    Int numberOfParameter;
    char** parameterNames;
    Real versionNumber;
    List solverList;
  } *AlconProblem;
```

enthält die Daten und Methoden zur Formulierung des Problems; `f, df, cf` dienen dabei zur Auswertung von $b = f(y)$, $A = f'(y)$ und $C = (f'(y)^* z)'$ und müssen vom aufrufenden Programm bereitgestellt werden. Die `fData, dfData, cfData` können dabei für die aufgerufenen Funktionen wichtige zusätzliche Daten enthalten. `m` ist die Anzahl der Variablen und `n` die Anzahl der Variablem inklusive des Parameters, also $= m + 1$. Der empfehlenswerteste Typ für A und C steht in `type` und `cType` und die maximale Anzahl an Nicht-Null Elementen der beiden Operatoren in `nz` und `nzc`. Die Prozeduren `df` und `cf` müssen allerdings damit rechnen, daß sie auch `LinOp`'s von anderem Typ auswerten sollen – ist dies nicht möglich, kann als boolescher Funktionswert `false` zurückgegeben werden.

`tol` enthält die gewünschte Genauigkeit der Lösung. `yGuess` muß einen Startwert nahe dem Lösungspfad enthalten. `tauMin` und `tauMax` enthalten die unteren und oberen Schranken des Definitionsbereiches. `parameter` enthält zusätzliche Parameter der Problems, bezüglich denen *nicht* fortgesetzt wird. Die solverList enthält eine Liste von `AlconSolver`'n; derzeit ist allerdings nur die Verwendung eines Lösers pro Problem implementiert.

Der `AlconSolver` organisiert die für die Lösung notwendigen Objekte in der folgenden Struktur:

```
  typedef enum {eNone, eSameAsDomain, eSymmetric} IScale;
  typedef struct {
    AlconProblem problem; /* das zu loesende Problem */
    Net net;              /* zum Abspeichern der Loesung */
    GNProblem GN, bifGN;  /* Newton-Vefahren zur Pfadverfolgung
                           und fuer Verzweigungen */
    LinAlg jacobi, moore; /* Lineare Algebra Routinen fuer die
                           zugehoerigen linearen Systeme */
    Euclid inner, full;   /* Skalierungen */
```

```
    RealVec scale;
    Real steplength, tangentMin, rho;
    Bool cautious, scaling, steplengthControl,tangentControl,
         bifurcations, turnings, normalized, bifNormalized;
    IScale imagescaling;
    Real versionNumber;
    Bool pm, pw;
  } *AlconSolver;
```

Dabei enthält `inner` die Skalierung des Urbildraums, während `full` die Skalierung (4.2) der Raumes von (y, z, α) enthält. `scale` enthält den von beiden Skalierungen als Gewichtung benutzten Skalierungsvektor. `steplenght` enthält einen Vorschlag für die aktuelle Schrittweite, `tangentMin` ist der Sicherheitsfaktor aus (4.4) und `rho` der Sicherheitsfaktor aus (1.27) und (1.29). `imagescaling` enthält eine der drei Alternativen zur Bildskalierung aus Abschnitt 4.1. `pm` und `pw` bwirken die Ausgabe von Meldungen und Warnungen an die Standard-Ausgabeeinheit.

Die Flags bedeuten

Flag:	Bedeutung, wenn gesetzt:
`cautious`	berechne den Tangentenschätzer (1.31) mit a-posterori-Schätzer und schalte den Test (4.5) ab
`scaling`	verwende skalierte Produkte und Normen
`steplengthControl`	führe die Schrittweitensteuerung aus Abschnitt 1.2.5 durch; sonst wird eine konstante Schrittweite genommen
`tangentControl`	verwende die Schrittweiteneinschränkung (1.32)
`bifurcations`	berechne einfache Verzweigungen
`turnings`	berechne Wendepunkte
`normalized`	verwende für `inner` die normalisierte Norm
`bifNormalized`	verwende für `full` die normalisierte Norm

Die Flags `scaling` bis `bifurcations` sind dabei empfehlenswert zu setzen; die anderen sind standartmäßig abgeschaltet.

Zur Bearbeitung des Problems stehen für die Pfadverfolgung, die Berechnung von Verzweigungen und die Berechnung von Wendepunkten je zwei globale Methoden zur Verfügung, die auf den einzigen aktiven Instances der Alcon-Objekte operieren, die in den globalen Variablen `gProblem` und `gSolution` gehalten werden:

```
void ComputeFirstSolution(void);
Bool ComputeNextSolution(Edge edge);

Bool DetectBifurcation(Edge e);
Bool ComputeBifurcation(Edge e, Bool* old);

Bool DetectTurningPoint(Edge e, Int i);
Bool ComputeTurningPoint(Edge e, Int i);
```

`ComputeFirstSolution` berechnet dabei eine Startlösung und legt zwei davon abzweigende `ghostEdges` in `gSolution->Net` an. `ComputeNextSolution` berechnet den Korrektor für die `ghostEdge egde`.

`DetectBifurcation` entdeckt eine Verzweigung zwischen `e->p0` und `e->p1`; `ComputeBifurcation` berechnet diese und setzt die Variable `old`, falls eine alte Verzweigung zum wiederholten Male wiedergefunden wurde.

`DetectTurningPoint` entdeckt einen Wendepunkt bezüglich der `i`-ten Variable zwischen `e->p0` und `e->p1`, der mittels `ComputeTurningPoint` berechnet werden kann.

Das Programm `Alcon-S` stellt nach außen hin einige Prozeduren zur Verfügung zur Berechnung eines Startpunktes, eines einzelnen Punktes, eines ganzen Zweiges oder des gesamten Verzweigungsdiagramms, sowie einige Initialisierungsroutinen:

```
void InitAlcon(void);            /* initalizer for alcon */
void InitConnection(char *name); /* load problem 'name' */
void MainEventLoop(void);        /* interactive main menu */
void NewDiagram(void);           /* methods of the main menu */
void FirstSolution(void);
void NextPoint(void);
void NextBranch(void);
void CompleteDiagram(void);
void QuitConnection(void);       /* quit actual problem */
void QuitAlcon(void);            /* exit routine for alcon */
```

Des weiteren existieren noch Prozeduren, die die berechneten Lösungen zur graphischen oder sonstigen Weiterverarbeitung auf Dateien schreiben. Die Prozedur `MainEventLoop` enthält einen interaktiven Dialog, der auch beim

eigenständigen Start von `Alcon-S` aufgerufen wird und das Programm mittels einer einfachen Menustruktur steuert; die Befehle bestehen dabei jeweils aus einzelnen Buchstaben, so daß sich zumindest Benutzer des `UCSD`-Pascal Betriebssystems schnell zurecht finden werden.

Auf eine Anbindung an eine bestimmte graphische Benutzeroberfläche wurde verzichtet, da wegen der geringen Portabilität das Programm schnell veraltet wäre. Zur graphischen Darstellung wurden statt dessen einige Scripts für MATLAB© geschrieben, die insbesondere eine auf eine Datei geschriebene Lösung mittels kubischer Hermite-Interpolation und Bezier-Techniken darzustellen vermögen.

5. Beispiele

5.1 Ein kleines, aber schwieriges Beispiel: Der Brusselator

Als erstes Beispiel werden für ein Modell einer chemischen Reaktion die Gleichgewichtslagen berechnet; es handelt sich also um ein Modell vom Typ (1.2).

Die Reaktion ist ein künstliches Modell für eine Reaktion, die auch Zustände außerhalb des statischen Gleichgewichts kennt. Sie wurde in [33] vorgeschlagen und mit dem Namen „Brusselator“ versehen. Die Reaktion besteht darin, daß in Reaktionszellen die Substanzen A, B, X, Y, E gemäß den Elementarreaktionen:

$$\begin{aligned} A &\rightarrow X \\ B + X &\rightarrow Y \\ 2X + Y &\rightarrow 3X \\ X &\rightarrow E \end{aligned}$$

zur Reaktion gebracht werden. Dabei werden die Konzentrationen der Edukte A und B und des Produkts E durch Regelung von außen konstant gehalten. Nach dem Massenwirkungsgesetz ergeben sich daraus nach Normierung einiger Reaktionskonstanten für die Änderungen der Konzentrationen $x_i = [X_i]$ und $y_i = [Y_i]$ der Substanzen X und Y in der Reaktionszelle i durch die Reaktion:

$$\begin{aligned} \frac{d}{dt} x_{i,react} &= A - (B+1)x_i + x_i^2 y_i \\ \frac{d}{dt} y_{i,react} &= B x_i - x_i^2 y_i \end{aligned}$$

Zusätzlich sind einige der Zellen noch miteinander verbunden, wodurch bei unterschiedlicher Konzentration der Stoffe Diffusion durch die Verbindung zwischen den Zellen der Unterschied auszugleichen sucht. Sind die Zellen i und j miteinander verbunden, dann verändert sich die Konzentration durch die Diffusion gemäß:

$$\begin{aligned} \frac{d}{dt} x_{i,diff_j} &= \frac{1}{\lambda^2}(x_j - x_i) \\ \frac{d}{dt} y_{i,diff_j} &= \frac{10}{\lambda^2}(y_j - y_i) \end{aligned}$$

Bezeichnet $K = \{(i,j) \mid \text{Zelle } i \text{ ist mit Zelle } j \text{ verbunden}\}$ die Kanten des Graphen der Verbindungen, dann lautet das Gleichungssystem für die Gleichgewichtsbedingung in Zelle i

$$\begin{pmatrix} \frac{d}{dt}x_{i,react} \\ \frac{d}{dt}y_{i,react} \end{pmatrix} + \sum_{(i,j)\in K} \begin{pmatrix} \frac{d}{dt}x_{i,diff_j} \\ \frac{d}{dt}y_{i,diff_j} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Bei n Zellen ergibt dies $2n$ Gleichungen für $2n$ Variablen und einen Parameter λ, der die Diffusion steuert.

Die Werte für A und B sind externe Parameter, die für die folgenden Rechnungen zu $A = 2$ und $B = 6$ gesetzt werden; es läßt sich dann leicht nachrechnen, daß unabhängig von der Diffusion immer die Lösung

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} \equiv \begin{pmatrix} 2 \\ 3 \end{pmatrix} \tag{5.1}$$

existiert; für eine isolierte Zelle ohne Diffusion ist es sogar die einzige Lösung. Falls nun $\lambda \to 0$, dann wird die Diffusion beliebig stark, so daß sich unterschiedliche Konzentrationen in miteinander verbundenen Zellen verbieten; es ergibt sich also nur die Lösung aus Gleichung (5.1). Falls $\lambda \to \infty$, werden die Zellen voneinander isoliert, da die Diffusion unwichtig wird; daher existiert auch hier nur die für isolierte Zellen eindeutige Lösung. Zwischen diesen Grenzfällen kann es jedoch, abhängig von der Konfiguration der Zellen, zu einem komplizierten Verzweigungsdiagramm kommen.

In den weiteren Rechnungen wird angenommen, daß die Zellen in linearer Kette angeordnet sind, so daß die Zelle i nur mit den Zellen $i-1$ und $i+1$ (falls diese existieren) verbunden ist. In [17] werden auch kompliziertere Geometrien mit Hilfe von Symmetrieausnutzung berechnet; die hier gegebene Symmetrie $(x_i, y_i) \leftrightarrow (x_{n-i+1}, y_{n-i+1})$ ist aber auch ohne Symmetriebetrachtungen noch berechenbar.

Da das Verzweigungsdiagramm recht umfangreich ist, ist dieses Beispiel als Test für die Zuverlässigkeit der Schrittweitensteuerung und der Modifikationen des Gauß-Newton-Verfahren geeignet. Zunächst wird für $n = 4$ das vollständige Verzweigungsdiagramm, das von der Lösung (5.1) abzweigt, berechnet. Dabei werden alle Rechnungen mit der Genauigkeit $\varepsilon = 10^{-7}$ über dem Parameterintervall $\lambda = [0.01, 7.5]$ durchgeführt. Als Startwert wird die Lösung (5.1) bei $\lambda = 0.25$ uns als Anfangsschritweite $s_0 = 0.1$ genommen.

Der Pfadverfolgungsalgorithmus berechnet 10 Verzweigungspunkte und 23 Wendepunkte. Die Lösung ist in Abbildung 5.1 dargestellt.

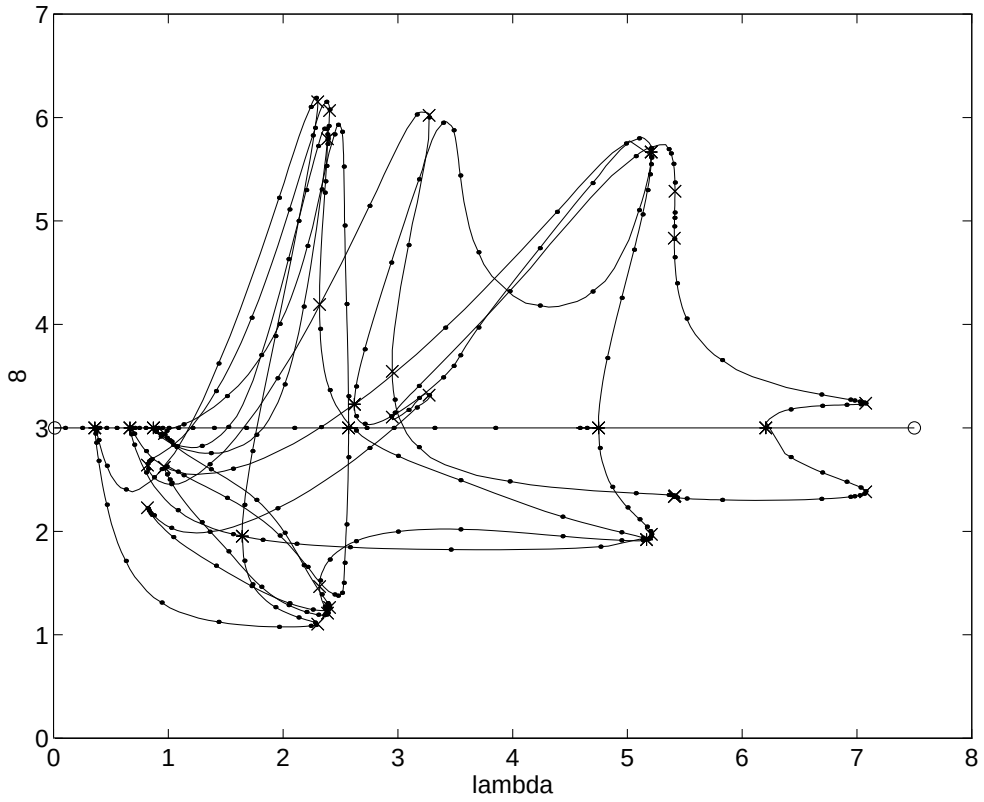


Abbildung 5.1: Verzweigungsdiagramm für den Brusselator mit 4 Zellen

An diesem Beispiel werden einige Varianten der Pfadverfolgung getestet, deren Einfluß auf den Aufwand unabhängig von der Größe des betrachteten Systems ist.

Zunächst wird der Einfluß der Skalierung auf die Berechnung betrachtet; dazu wird das Verzweigungsdiagramm unter Verwendung der QR-Zerlegung zur Berechnung der Pseudoinversen ohne Modifikation des Gauß-Newton-Verfahrens berechnet; dabei wird einmal ohne Skalierung gerechnet und mit Skalierung des Urbildes wie in 4.1 die drei Varianten der Bildskalierung berechnet. Dabei ergibt sich folgendes Diagramm für den Aufwand:

	Skalierung			
	ohne	mit		
		$D_r = Id$	$D_r = D_{d_x}$	$D_r = {D_{d_x}}^{-1}$
#f	1672	1282	1284	1279
#df	1264	978	980	974
#cf	50	45	47	44
#df-cmp	1188	912	912	910
#solve	2273	1739	1739	1739
#gn-iter	1156	888	888	888
#bif-iter	50	45	47	44
#punkte	420	322	322	322

Dabei bedeuten die Zeilen #f, #df und #cf die Anzahl der nötigen f, f' und $(f'(y)^* z)'$ Auswertungen. #df-cmp gibt die Anzahl der benötigten Zerlegungen des Lösers der linearen Gleichungssysteme an und #solve die Anzahl der gelösten Gleichungssysteme. Für die Pfadverfolgung wurden #gn-iter Gauß-Newton-Iterationen und für die Verzweigungen #bif-iter Newton-Iterationen benötigt; das berechnete Verzweigungsdiagramm besteht aus #punkte Punkten.

Die Skalierung senkt den Aufwand in diesem Beispiel um rund 23% für die Pfadverfolgung; für die Berechnung von Verzweigungen ergibt sich keine so große Reduktion des Aufwandes, da die Vorteile der Skalierung teilweise dadurch aufgehoben werden, daß wegen der größeren Schrittweite in der Pfadverfolgung schlechtere Startwerte für die Verzweigung interpoliert werden können. Der Einfluß der Bildskalierung ist hingegen verschwindend gering.

Die weiteren Beispiele sind mit Urbildskalierung und der dritten Variante $D_r = {D_{d_x}}^{-1}$ für die Bildskalierung gerechnet.

Die modifizierten Gauß-Newton-Verfahren bewirken eine Verringerung des Aufwandes für eine einzelne Newton-Iteration und gleichzeitig eine Vergrößerung der Anzahl der Iterationsschritte. Bei Testläufen an diesem Beispiel ergab sich, daß sich die Einsparung des Aufwandes im Einzelschritt durchaus lohnt, wobei die Broyden-Updates am besten abschnitten.

Zunächst wird der Einfluß der Modifikation auf die Pfadverfolgung getestet; dazu wurden die Verzweigungen mit einem normalen Newton-Verfahren berechnet:

	Gauß-Newton		
	normal	Broyden	vereinf.
#f	1279	1808	2399
#df	974	756	798
#cf	44	44	42
#df-cmp	910	692	736
#solve	1739	1718	3907
#gn-iter	888	1410	1985
#bif-iter	44	44	42
#punkte	322	324	339

Umfangreichere Tests ergaben, daß die Modifikationen des Gauß-Newton-Verfahrens für die Pfadverfolgung und des Newton-Verfahrens für die Verzweigungen den Aufwand fast unabhängig voneinander beeinflussen; daher werden hier nur die Ergebnisse der Modifikation des Newton-Verfahrens für Verzweigungen angegeben, die sich bei der Berechnung der Pfadverfolgung mit den Broyden-Updates ergaben:

	Newton		
	normal	Broyden	vereinf.
#f	1808	1819	1830
#df	756	770	781
#cf	44	30	30
#df-cmp	692	694	694
#solve	1718	1716	1716
#gn-iter	1410	1411	1411
#bif-iter	44	76	87
#punkte	324	324	324

Bei der Berechnung von Verzweigungen haben die Modifikationen einen weniger ausgeprägten Vorteil als bei der Pfadverfolgung; insbesondere das vereinfachte Newton-Verfahren fällt stark ab.

Bei allen Berechnungen wurde die Pseudoinverse mittels einer QR-Zerlegung berechnet, ohne daß dies ausdrücklich erwähnt wurde; dies liegt daran, daß die Berechnung auf eine andere Weise den Aufwand kaum beeinflußt.

Zum Vergleich wird angegeben, wie sich der Aufwand verändert, wenn anstelle der QR-Zerlegung eine Zerlegung auf Basis des Sparse-Solvers `MA28` vorgenommen wird; dabei werden von den verschiedenen Varianten der Pivotstrategie die Variante $\texttt{MA28}_1$, die für jede Zerlegung eine neue Pivotstrategie entwickelt, und $\texttt{MA28}_0$, die immer die Pivotstrategie der ersten Zerlegung benutzt, bis der Löser versagt, getestet.

Die Berechnung erfolgt bei beiden Newton-Verfahren über Broyden-Updates. Zur Abwechslung werden die Wendepunkte mit berechnet.

	Lineare Algebra		
	QR	$\texttt{MA28}_1$	$\texttt{MA28}_0$
#f	2426	2547	2572
#df	1378	1428	1453
#cf	30	31	36
#df-cmp	1300	1349	1370
#solve	2321	2440	2456
#gn-iter	1946	2044	2055
#bif-iter	78	79	83
#punkte	464	502	504

Das Ergebnis ist frappierenderweise, daß der MA28 mit einer einzigen Pivotstrategie das gesamte Diagramm fast genauso gut berechnet, wie mit einer immer neuen Pivotstrategie. Lediglich an den Verzweigungen zeigen sich bei der alten Pivotstrategie Schwächen.

Dementsprechend nehmen sich auch die verschiedenen Varianten des Block-Lösers nicht viel. Als Block-Löser mit QR-Zerlegung wurde die Variante von Govaerts & Pryce gewählt, während die Block-Zerlegung mit dem MA28 sowohl mit dem Verfahren von Chan als auch dem von Govaerts & Pryce berechnet wurde. Beim Verfahren von Chan wurde das auftretende kleine System mittels QR-Zerlegung gelöst. Entsprechend den Erfahrungen der vorherigen Beispielrechnung wurde beim MA28 die Pivotstrategie entlang eines Pfades beibehalten und nur an Verzweigungen eine neue Pivotstrategie entwickelt.

	Lineare Algebra		
	QR-Block	MA28+Chan	MA28 + Gov. & Pry.
#f	2561	2576	2573
#df	1444	1443	1455
#cf	33	31	37
#df-cmp	1353	1365	1372
#solve	2440	2471	2456
#gn-iter	2041	2070	2055
#bif-iter	91	78	83
#punkte	502	509	504

Insgesamt erweist sich der Sparse-Solver MA28 für den Brusselator mit 4 Zellen zwar als brauchbar; allerdings ist er aufgrund des Overheads in den Datenstrukturern für dieses kleine System noch langsamer als z.B. die QR-Zerlegung.

Die beiden Varianten der Block-Löser ergeben gleich gute Ergebnisse im Aufwand, wobei bei der Berechnung von Verzweigungen die Variante von Chan besser abschneidet, diesen Vorteil bei der Pfadverfolgung wieder verliert. Denn es muß berücksichtigt werden, daß die Variante von Chan aufgrund des komplizierteren Algorithmus und der vielen inversen Vektoriterationen zum Erzwingen des Rangdefektes fast doppelt so lange für die Rechnung braucht als die Variante von Govaerts & Pryce.

Wird versucht, die Dimension des Systems durch Anschluß von mehr Zellen in die Höhe zu treiben, ergeben sich bald (nämlich schon für $n = 5$)

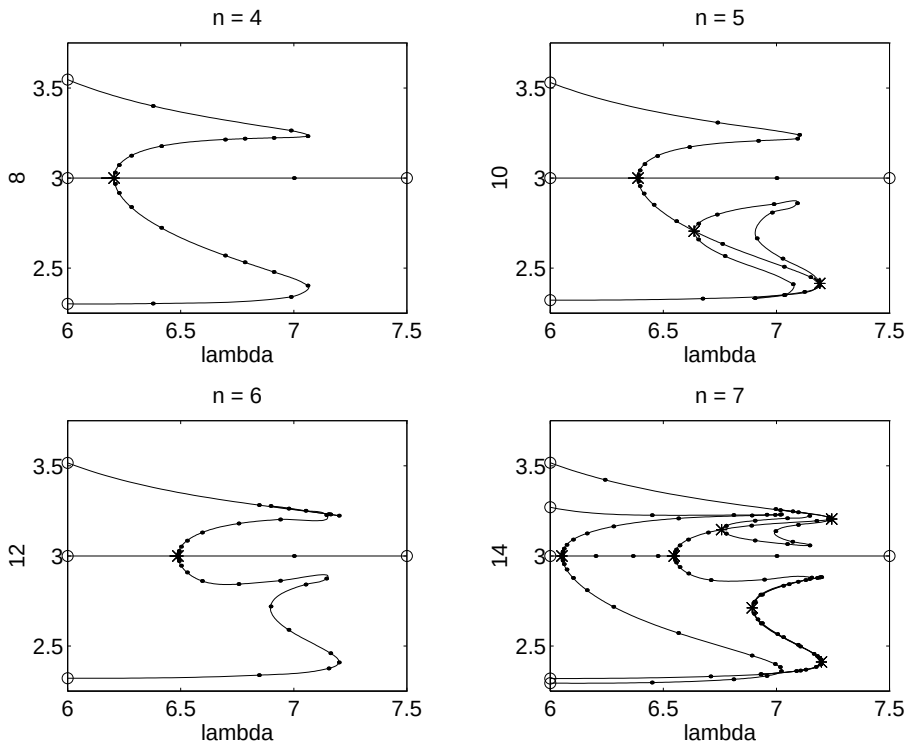


Abbildung 5.2: Detailstudie für 4,5,6 und 7 Zellen

sehr unübersichtliche Verzweigungsdiagramme. Ohne die Schrittweiteneinschränkung, die durch Zusammenziehen der Schrittweite bei der Berechnung von Wendepunkten ergibt, läßt sich das Diagramm mit 22 Verzweigungen und 88 Wendepunkten nur noch durch Einschalten des Testes (4.4) mit $\text{tangent}_{min} \geq 0.8$ berechnen.

Anhand einer kleinen Detailstudie (Abbildung 5.2) läßt sich erkennen, wie sich das Verzweigungsdiagramm mit steigender Dimension schnell verkompliziert. Sei dazu der Parameterbereich zwischen $\lambda = 6.0$ und $\lambda = 7.5$ eingeschränkt. Im Verzweigungsdiagramm ist auch deutlich zu erkennen, daß abhängig davon, ob die Anzahl der Zellen gerade oder ungerade ist, die Zweige, die bei $\lambda \approx 6.5$ abzweigen, symmetrisch zueinander sind, oder nicht. Im ersteren Fall läßt sich die Symmetrie an den gewählten Punkten erkennen; da

für den Pfadverfolungsalgorithmus beide Pfade bezüglich des Konvergenzverhaltens gleich aussehen, werden auch zueinander symmetrische Punkte auf den beiden Zweigen gewählt, was an der jeweils gleichen λ-Koordinate der Punkte erkannt werden kann. Im ungeraden Fall sind beide Zweige jeweils zu sich selbst symmetrisch; daher kann von dem einen Zweig eine weitere Verzweigung anzweigen, aber von dem anderen nicht.

Die Verkomplizierung mit steigender Anzahl von Zellen läßt sich dadurch erklären, daß bei den inneren Zellen Verzweigungen auftreten, die die Lösung in den äußeren Zellen kaum beeinflussen; etwa sieht das Verzweigungsdiagramm für $n = 12$, wie von `Alcon-S` mit den Sicherheitsfaktoren $\rho = 0.25$ und $\text{tangent}_{min} = 0.7$ in Abbildung 5.3 berechnet, für eine mittlere Zelle sehr ausgedehnt aus, während für eine Randzelle viele der Lösungen kaum noch voneinander zu unterscheiden sind.

Das Programm hat hier schon bei diesem kleinen Ausschnitt einen Fehler begangen, allerdings einen verzeihlichen; die Verzweigungen bei $\lambda \approx 6.7$ wurden aus einer Richtung als gestörte Verzweigungen berechnet, aus einer anderen Richtung aber nicht bemerkt; statt dessen wurden die Pfade der gestörten Verzweigung getrennt berechnet. Die führte dazu, daß Teile des Verzweigungsdiagrammes zweimal berechnet wurden.

An diesem Beispiel läßt sich auch die Problematik der Berechnung von Verzweigungen mit dem Block-Meta-Löser in der Variante von Govaerts und Pryce feststellen, da auch Verzweigungen von Typ (2.8) auftreten, die mit diesem Löser nicht korrekt behandelt werden können, wie in Abschnitt 3.3.5 diskutiert wurde.

Tatsächlich fällt der Fehler bei den Verzweigungen für $n = 7$ nicht ins Gewicht; die Lösung ist trotzdem hinreichend korrekt. Bei $n = 12$ wurde das Verzweigungsdiagramm aber nicht vollständig berechnet; falls zum Beispiel der `MA28` als Block-Löser eingesetzt wird, ergibt sich als Aufwand:

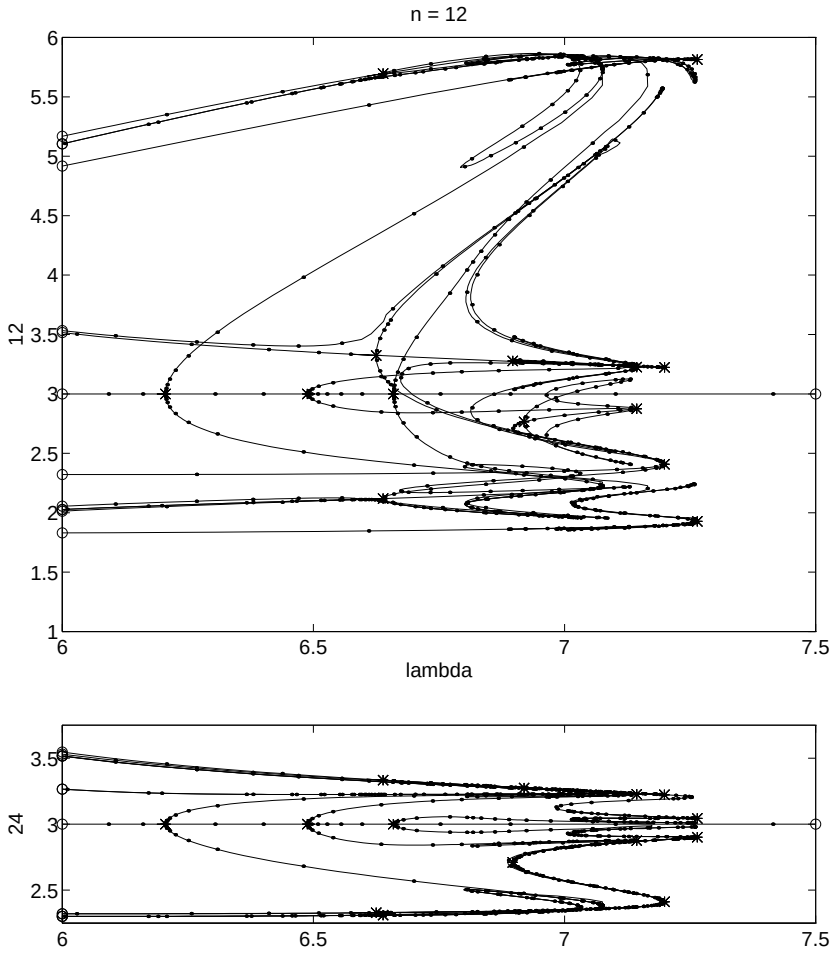


Abbildung 5.3: Detailstudie für 12 Zellen

	ohne Block	mit Block
#f	4071	3852
#df	2050	1991
#cf	32	49
#df-cmp	1972	1881
#solve	3896	3624
#gn-iter	78	2840
#bif-iter	91	110
#punkte	959	891
#bif-pkt	14	12

Die Verzweigungen bei $\lambda \approx 7.3$ wurden dabei nicht berechnet; bei Ersetzen des `MA28` durch eine QR-Zerlegung wurden sogar noch zwei Verzweigungen weniger berechnet.

5.2 Invariante Mannigfaltigkeiten gekoppelter Oszillatoren

Ein Quelle immmer wieder dankbarer Testbeispiele für Löser von Problemen hoher Dimension ergibt sich aus der Diskretisierung partieller Differentialgleichungen. Zwar sind diese Gleichungen eigentlich unendlichdimensionaler Natur, und es ist nicht zweckmäßig, diese im Kontext der Pfadverfolgung mit einem festen Gitter zu diskretisieren, da sich die Lösung durch Änderung des Parameters oft qualitativ verändert. Jedoch um zu studieren, wie sich der Algorithmus verhält, wenn die Dimension des Problems steigt, sind diese Beispiele sehr geeignet, da sich durch Verfeinerung der Diskretisierung die Dimension beliebig hoch schrauben läßt. Da bei Berechnung einer diskreten Lösung der Nachweis, daß diese Lösung noch mit dem kontinuierlichen Problem in Verbindung steht, zusätzliche Betrachtungen erfordert, die den hier gegebenen Rahmen sprengen, das Problem aber auch nur zur Demonstration der Leistungsfähigkeit des Algorithmus vorgestellt wird, wird eine zwar wenig zweckmäßige, aber sehr einfache Diskretisierung verwendet.

Das Beispiel ist in [1] als ein generisches Modell für die schwache Kopplung zwischen harmonischen Oszillatoren eingeführt worden; in der vorgestellten Formulierung ist es aus [39] entnommen.

Die Bewegungsgleichungen für die n Oszillatoren werden in Polarkoordinaten formuliert; dabei gelten für den i-ten Oszillator die Bewegungsgleichungen:

$$\begin{aligned} \dot{\theta}_i &= \omega \\ \dot{r}_i &= r_i(1 - r_i^2) \end{aligned}$$

wenn die Oszillatoren ungekoppelt sind. Außer der instabilen Lösung $r_i = 0$ nähern sich alle Lösungen an eine periodische Lösung mit $r_i = 1$ und dem Winkel $\theta_i(t) = \theta_i(0) + \omega t$ an. Die Kopplung zweier Oszillatoren i und j findet über Kopplungsterme $\lambda\Gamma_{ij}$ für θ und λC_{ij} für r statt, wobei die Γ_{ij} und C_{ij} lauten:

$$\begin{aligned} \Gamma_{ij} &= -\cos 2\theta_j + \frac{r_j}{r_i}\left(\cos\left(\theta_i + \theta_j\right) - \sin\left(\theta_i - \theta_j\right)\right) \\ C_{ij} &= r_j\left(\sin\left(\theta_i + \theta_j\right) + \cos\left(\theta_i - \theta_j\right)\right) - r_i\left(1 + \sin 2\theta_i\right) \end{aligned}$$

Damit lauten die Bewegungsgleichungen nun

$$\begin{aligned} \dot{\Theta} = h(\Theta, R) \quad &\text{mit} \quad h_i = \omega + \lambda \sum_{(i,j)\in K} \Gamma_{ij} \\ \dot{R} = g(\Theta, R) \quad &\text{mit} \quad g_i = r_i(1 - r_i^2) + \lambda \sum_{(i,j)\in K} C_{ij} \end{aligned}$$

Für $\lambda = 0$ existiert ein n-dimensionaler Torus $(\Theta, R(\Theta))$ mit $r_i(\Theta) \equiv 1$, der unter dem Fluß der Differentialgleichung invariant bleibt und sogar einen Attraktor bildet; für schwache Kopplung, d.h. für kleine Werte von λ, ist es zu erwarten, daß der Torus mit der Parametrisierung $R = R(\Theta)$ weiterexistiert. Für stärkere Kopplung ist es hingegen an einem einfacheren Modell (z.B. in [35], Cp. 6) beobachtet worden, daß sich der Torus stark wellt, bevor er durch die Kopplung zerstört wird; dies ist gerade dann der Fall, wenn die Möglichkeit der Parametrisierung über Θ verlorengeht. In [14], wo der Torus für spezielle Parameterwerte berechnet wurde, konnte ein solches Verhalten auch für das vorliegende Modell beobachtet werden.

Der invariante Torus $R(\Theta)$ läßt sich dabei durch die Gleichung:

$$g(\Theta, R(\Theta)) = \dot{R} = \frac{dR}{d\Theta}\dot{\Theta} = \frac{dR}{d\Theta}h(\Theta, R(\Theta))$$

als Lösung eines Systems von nichtlinearen partiellen Differentialgleichungen auffassen. Da $R(\Theta,\lambda) = 1 + \mathcal{O}(\lambda)$ für kleine λ, ist es zur numerischen Berechnung günstig, eine neue Variable $S(\Theta,\lambda)$ über $R(\Theta,\lambda) = 1 + \lambda S(\Theta,\lambda)$ einzuführen. Mit dieser neuen Variable lautet die zu lösende Gleichung:

$$f(\Theta, S(\Theta,\lambda)) = g(\Theta, 1+\lambda S) - \lambda \frac{dS}{d\Theta} h(\Theta, 1+\lambda S) = 0 \tag{5.2}$$

Diese Gleichung wird nun durch finite Differenzen mit äquidistanten Knoten diskretisiert. Dazu wird die kontinuierliche Variable $\Theta \in [0, 2\pi)^n$ ersetzt durch

$$\theta_i \to \theta_k^i := \frac{2\pi}{n_{grid}} k \qquad k = 0 \ldots n_{grid} - 1$$

wobei n_{grid} die Anzahl der Knoten und damit die Feinheit des Gitters bezeichnet. Die Variable S an der Stelle Θ wird an Gitterpunkten durch

$$s_i(\Theta) \to s_i(\theta_{k_1}^1, \ldots, \theta_{k_n}^n) =: s_{k_1,\ldots,k_n}^i$$

ersetzt.

Die Ableitung $\frac{dS}{d\Theta}$ wird durch symmetrische Differenzen:

$$\begin{aligned}
\left(\frac{dS}{d\Theta}\right)_{ij} = \frac{\partial s_i}{\partial \theta_j} \quad \to \quad & \frac{\Delta s_{k_1,\ldots,k_n}^i}{\Delta \theta_{k_j}^j} \\
& := \frac{1}{2}\left(\frac{s_{k_1,\ldots,k_j+1,\ldots,k_n}^i - s_{k_1,\ldots,k_n}^i}{\theta_{k_j+1}^j - \theta_{k_j}^j} + \frac{s_{k_1,\ldots,k_j-1,\ldots,k_n}^i - s_{k_1,\ldots,k_n}^i}{\theta_{k_j-1}^j - \theta_{k_j}^j}\right) \\
& = \frac{s_{k_1,\ldots,k_j+1,\ldots,k_n}^i - s_{k_1,\ldots,k_j-1,\ldots,k_n}^i}{2h}
\end{aligned}$$

approximiert; dabei ist für äquidistante Knoten $h = 2\pi/n_{grid}$ die Knotendistanz; die Addition und Subtraktion der Indizes $k_j + 1$ und $k_j - 1$ ist modulo n_{grid} zu nehmen, da aufgrund der Periodizität $\theta_{n_{grid}}^j = \theta_0^j$ ist.

Damit lautet die diskretisierte Gleichung (5.2) komponentenweise

$$\begin{aligned}
f_{k_1\ldots k_n}^i(S,\lambda) \quad = \quad & -\lambda s_{k_1\ldots k_n}^i (1 + \lambda s_{k_1\ldots k_n}^i)(2 + \lambda s_{k_1\ldots k_n}^i) + \lambda \sum_{(i,l)\in K} C_{i,l} \\
& -\lambda \sum_{j=1}^n \frac{\Delta s_{k_1\ldots k_n}^i}{\Delta \theta_{k_j}^j} \left(\omega + \lambda \sum_{(j,l)\in K} \Gamma_{j,l}\right)
\end{aligned}$$

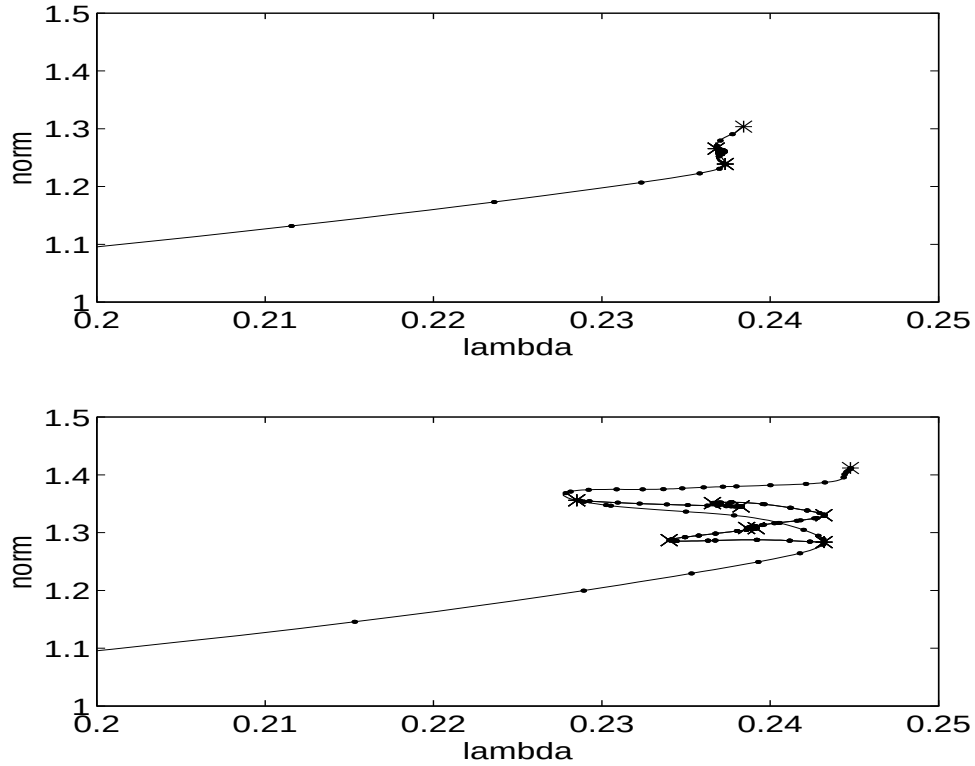


Abbildung 5.4: Verzweigungsdiagramm für $n_{grid} = 15$ und 25

Dies sind $n * n_{grid}^n$ Gleichungen für die $n * n_{grid}^n$ Variablen $s^i_{k_1,\ldots,k_n}$ und den einen Parameter λ.

Aufgrund der Transformation $R = 1+\lambda S$ ist für $\lambda = 0$ die Lösung S beliebig; als Startpunkt für die Pfadverfolgung wird der Punkt verwendet, an dem der Lösungsast von dieser Hyperebene abzweigt; er läßt sich aus $\frac{\partial}{\partial\lambda} f\Big|_{\lambda=0} = 0$ bestimmen, was nur die Lösung einer linearen Differentialgleichung erfordert.

Da der Rechenaufwand exponentiell schnell mit der Anzahl der Oszillatoren steigt, wird nur die einfachste Kombination, nämlich die zweier gekoppelter Oszillatoren, betrachtet; in diesem Fall läßt sich der invariante Torus, der ein zweidimensionales Objekt $(r_1(\theta_1,\theta_2), r_2(\theta_1,\theta_2))$ bildet, graphisch leicht veranschaulichen.

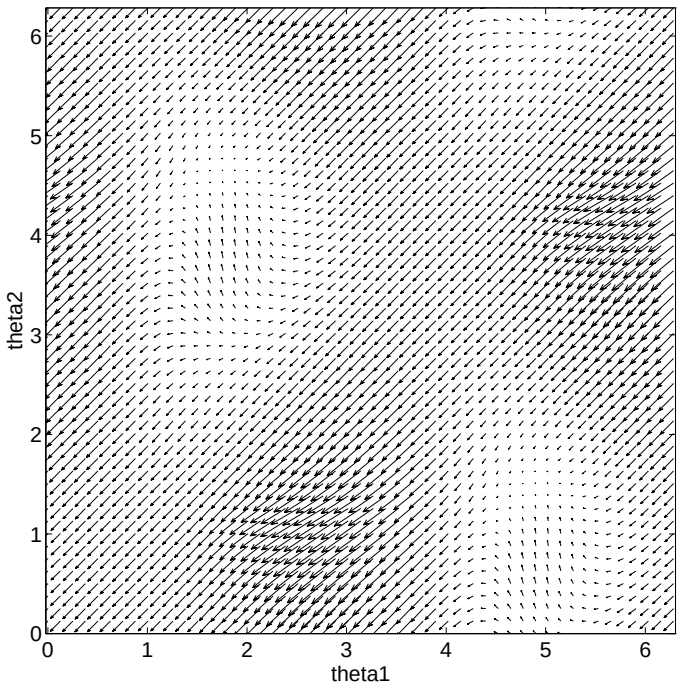


Abbildung 5.5: Fluß auf den invarianten Tori vor der Verzweigung

Abhängig von der Feinheit des Gitters veränderte sich das Verzweigungsdiagramm hinter der zweiten Verzweigung stark; offensichtlich stehen die diskreten Lösungen in keinem Zusammenhang zueinander und zur kontinuierlichen Lösung. Daher wird für verschiedene Gitterfeinheiten die Lösung nur bis zur zweiten Verzweigung dargestellt, bis zu der sich die Lösungen qualitativ nur wenig unterscheiden. Aufgetragen ist jeweils die normalisierte Norm über den Parameter.

Es ist zu erkennen, daß eine (symmetriebrechende) Verzweigung im Bereich des ersten Wendepunktes stattfindet; innerhalb der numerischen Genauigkeit handelt es sich um eine Verzweigung von Typ (2.8). Die beiden unsymmetrischen Äste treffen sich in einer Verzweigung desselben Types wieder.

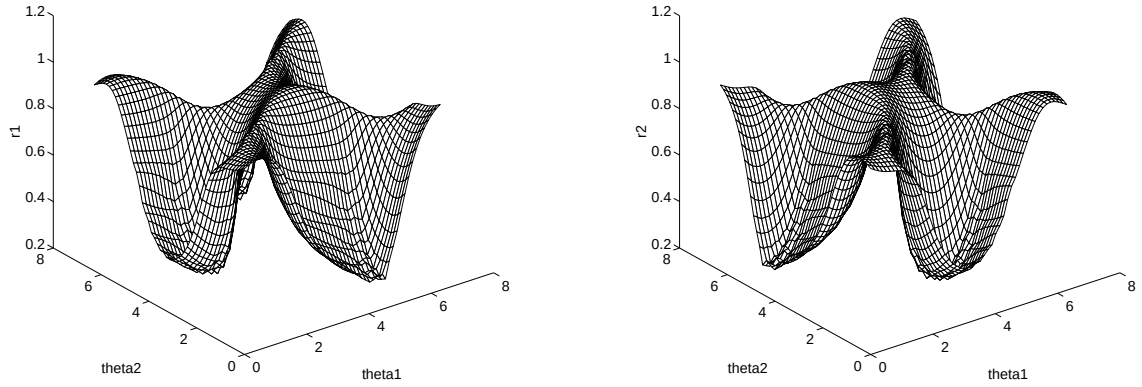


Abbildung 5.6: Invariante Tori vor der Verzweigung

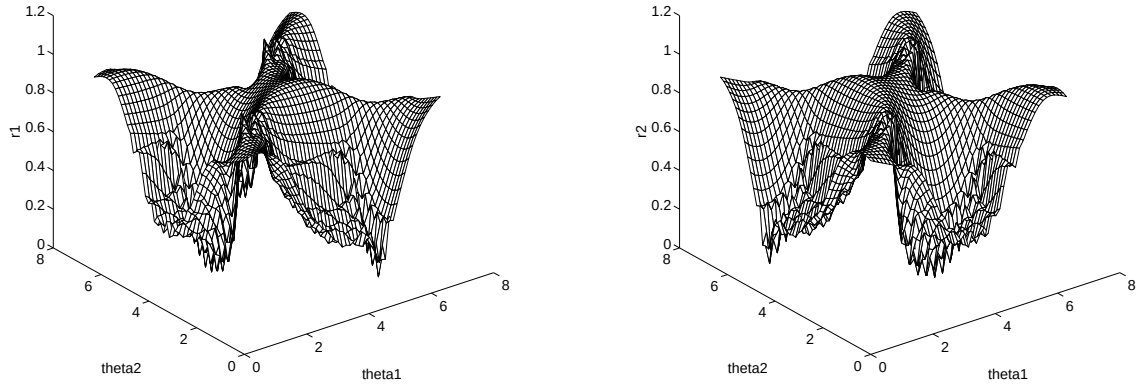


Abbildung 5.7: Invariante Tori nach der Verzweigung (symmetrischer Ast)

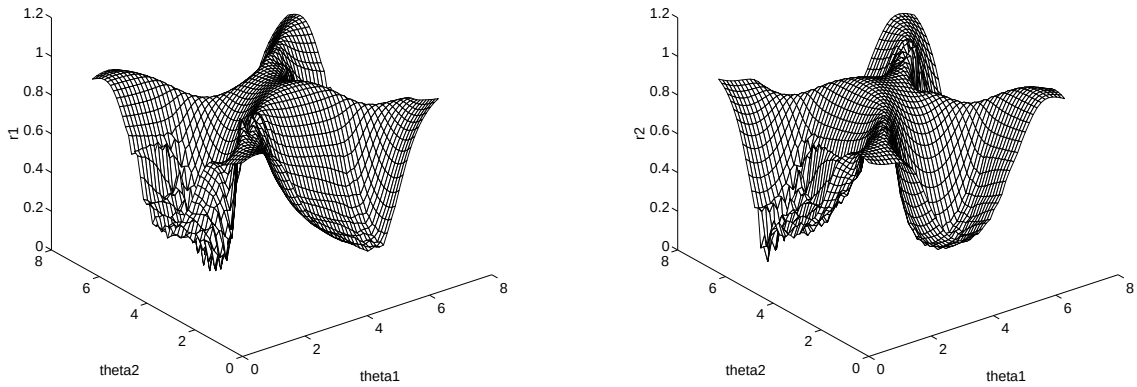


Abbildung 5.8: Invariante Tori nach der Verzweigung (unsymmetrischer Ast)

Bei Betrachtung der invarianten Tori, die in den Figuren 5.6-5.8 für $\lambda \approx 0.2480$ vor der Verzweigung und nach der Verzweigung auf den symmetrischen und einem der unsymmetrischen Ästen dargestellt sind, (wobei die Tori wegen der Parametrisierungsmöglichkeit $r_i = r_i(\Theta)$ entlang der Koordinatenachsen $\theta_i = 0$ aufgeschlitzt und auf eine Ebene abgewickelt wurden,) fällt es auf, daß die Lösung des diskreten Problems vor der Verzweigung glatt aussieht, während sie nachher teilweise stark aufgerauht scheint. Es ist daher plausibel, mit [39] die Vermutung aufzustellen, daß zwar die Lösung vor der Singularität die kontinuierliche Lösung gut approximiert, aber dahinter keine solche Approximation stattfindet, vielleicht weil hinter der Verzweigung gar keine kontinuierliche Lösung mehr vorhanden ist. Ein adaptiver Löser des Problems würde sich in diesem Fall an der Verzweigung festfressen, da er an den aufgerauhten Stellen ad infinitum verfeinern würde, ohne eine zufriedenstellende Approximation zu erreichen.

An diesem Beispiel wird mit $n_{grid} = 25$ zunächst der Einfluß der Nachiteration (3.4.2) bei der Berechnung von Verzweigungen untersucht und dann die Bedeutung der Block-Löser gezeigt.

Für die Nachiteration ergibt sich ernüchterndes; für den Fall eines gewöhnlichen Newtonverfahrens ergibt sich gar keine Änderung des Aufwandes. Für den Fall eines vereinfachten Newton-Verfahrens ergibt sich folgendes Ergebnis für den Aufwand bis zur Berechnung der ersten Verzweigung, wobei zur Zerlegung der Ableitung von f der Block-Löser in der Variante von Chan benutzt wird: (angegeben sind nur die zur Berechnung von Verzweigungen relevanten Aufwandszahlen)

	Nachiteration		
	ohne	teilweise	vollständig
#f	100	102	99
#df	40	43	39
#cf	2	3	2
#bif-iter	11	13	10

Während die vollständige Nachiteration eine minimale Aufwandverringerung erbringt, führt die Verwendung der teilweisen Nachiteration sogar zu Nachteilen; das Verfahren bricht einmal die Iteration wegen Verletzung des Monotonietestes ab. Dies liegt vermutlich darin begründet, daß während der Newton-Iteration die vernachlässigten Größen α und S klein werden, während zur Nachiteration die Werte für α und S am Startpunkt verwendet werden.

Bei der Verwendung von Broyden-Uptates ergibt sich ein ähnliches Bild.

Da die Gleichungen aus der Diskretisierung einer eigentlich unendlichdimensionalen Gleichung stammen, wo der einzelne Parameter gegenüber den kontinuierlichen Variablen ausgezeichnet ist, ist es zu vermuten, daß die Verwendung eines Block-Lösers von Vorteil ist. Dies wird auch durch die Aufwandsbetrachtung gerechtfertigt. Dabei erweist sich die Variante von Govaerts und Pryce als nicht viel instabiler als die von Chan, obwohl sie die Verzweigung theoretisch gar nicht berechnen könnte. Der Einfluß von Rundungsfehlern läßt jedoch anscheinend die Interpretation dieser Verzweigung als nicht vom Typ (2.8) zu.

	Block-Löser		
	ohne	Gov.& Pry.	Chan
#f	105	83	83
#df	57	43	40
#cf	5	4	2
#df-cmp	43	33	29
#solve	71	65	65
#gn-iter	69	57	59
#bif-iter	14	10	11
#punkte	17	14	14

Als Löser für das Block-System, wie auch für das gesamte System ohne Verwendung der Block-Lösers, wurde der `MA28` genommen. Im Diagramm ist nicht angegeben, daß der `MA28` für das ganze System auf dem Pfad zur ersten Verzweigung zwei neue Pivotstrategien benötigte, da einmal die Nachiteration scheiterte, während bei der Verwendung von Block-Lösern eine Pivot-Strategie für den ganzen Pfad reichte.

5.3 Stabwerkkonstruktionen

Stabwerke sind Konstruktionen aus elastisch dehnbaren Stäben, die an Knotenpunkten fest miteinander verbunden sind; das Gewicht der Stäbe wird als vernachlässigbar angenommen. Wenn diese Stabwerke von äußeren Kräften beeinflußt werden, werden sie sich aus ihrer ursprünglichen Gleichgewichtslage verschieben. Von den Kräften wird angenommen, daß sie nur an den Knotenpunkten angreifen; ein Durchbiegen der Stäbe ist also nicht im Modell enthalten. Einige Koordinaten der Knotenpunkte werden als unverschiebbar angenommen, z.B. die z-Koordinaten der auf dem Boden aufliegenden Knoten.

Das Modell

Werden die Stäbe durch Verschiebung aus ihrer Gleichgewichtslage gestaucht oder gestreckt, so üben sie auf die Konstruktion eine Rückstellkraft aus. Die Modellierung dieser Rückstellkraft erfolgt wie in [18].

Verbindet dabei etwa der Stab k die Knoten i und j miteinander, dann übt er auf den Knoten i die Rückstellkraft

$$\vec{F}_{k,i} = -\sigma(\varepsilon_{i,j}) A_k \frac{\vec{x}_i - \vec{x}_j}{r_{i,j}} \tag{5.3}$$

aus; dabei ist $r_{i,j} = \|\vec{x}_i - \vec{x}_j\|$ die Länge des Stabes, und wenn $r^0_{i,j} = \|\vec{x}^0_i - \vec{x}^0_j\|$ die Länge des Stabes in Ruhelage ist, dann wird mit $\varepsilon_{i,j} = (r_{i,j} - r^0_{i,j})/r^0_{i,j}$ seine relative Längenänderung bezeichnet. Für kleine Längenänderungen läßt sich die Spannungskurve hinreichend genau durch $\sigma = \varepsilon E$ mit dem Elastizitätsmodul E als Proportionalitätskonstante beschreiben; für größere Längenänderungen bleibt aber die Rückstellkraft hinter der Längenänderung zurück, was im Modell durch Modifikation zu $\sigma(\varepsilon) = (\varepsilon - \gamma\varepsilon^3)E$ berücksichtigt wird. Dabei wurde γ als 10.70 gewählt; als Elastizitätsmodul wurde $E = 2 \cdot 10^5$ N/mm^2 verwendet. Das A_k ist die Querschnittsfläche des Stabes.

Als Bedingung, daß sich eine Stabwerkskonstruktion unter Einfluß äußerer Kräfte im Gleichgewicht befindet, muß gelten, daß die Summe aller in jedem Knoten angreifenden Kräfte gleich Null ist; dies ergibt das zu lösende Gleichungssystem. Der Parameter λ steuert dabei in den Beispielen die äußeren Kräfte.

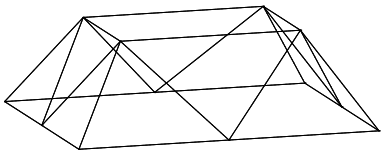
Zur Auswertung der Funktion und der Ableitung müssen die Koordinaten der Knoten in Ruhelage, eine Liste der verbindenden Stäbe, eine Liste der äußeren Kräfte und eine Liste der unverschiebbaren Koordinaten vorhanden sein. Die Verschiebungen aus der Ruhelage bilden die Unbekannten des Gleichungssystems.

Durchlaufen der Liste der Stäbe ergibt die Summe der inneren Kräfte; verbindet etwa der Stab k die Knoten i und j, dann wird $\vec{F}_{k,i}$ gemäß (5.3) berechnet und auf $\vec{F}_i$ aufaddiert, ebenso $\vec{F}_{k,j} = -\vec{F}_{k,i}$ auf $\vec{F}_j$. Bei der Ableitung werden jeweils die 3×3-Blöcke auf der Diagonalen für $\frac{\partial}{\partial \vec{x}_i}\vec{F}_i$ und $\frac{\partial}{\partial \vec{x}_j}\vec{F}_j$ sowie ausserhalb der Diagonalen für $\frac{\partial}{\partial \vec{x}_j}\vec{F}_i$ und $\frac{\partial}{\partial \vec{x}_i}\vec{F}_j$ eingetragen. Die äußeren Kräfte, von denen angenommen wird, daß sie stets die Form $\vec{F}_{i,ext} = \lambda\vec{F}_{i,0}$ haben, werden danach aufaddiert; nur sie liefern bei Berechnung der Ableitung einen Eintrag in die Parameterspalte. Für die zweite Ableitung entstehen in der Parameterspalte keine Einträge; ein 3×3-Block-Eintrag entsteht bei den Variablen außer auf der Diagonalen nur für solche Indexpaare (i, j), für die die zugehöringen Knoten über höchstens einen weiteren Knoten durch Stäbe verbunden sind.

Zuletzt werden die Kräfte, die an fixierten Koordinaten angreifen, durch genügend große Rückstellkräfte ersetzt, etwa $F_{z_i} = \kappa(z_i - z_i^0)$; dabei ist κ eine Skalierungsgröße, die in derselben Größenordnung wie die anderen Einträge in der Ableitung liegen muß, wenn der Rangentscheid nicht κ oder den Rest der Matrix für vernachlässigbar klein halten soll. Bei den Beispielen wurde $\kappa = E$ verwendet.

Eine kleine Zeltkonstruktion

Als erstes Beispiel wird eine kleine Zeltkonstruktion berechnet, die dieselbe Symmetrie wie das danach folgende größere Beispiel hat; daher sind gewisse Ähnlichkeiten in den Verzweigungsdiagrammen, was die Symmetrie brechenden Pitchfork-Verzweigungen angeht, zu erwarten.



Zeltkonstruktion

Das Zelt ist laut nebenstehender Zeichnung konstruiert, hat einen Grundriß von 8 m $\times$ 10 m und eine Höhe von 3 m. Alle Stäbe besitzen eine Querschnittsfläche von $A = 1000$ mm^2. Die unteren Eckknoten sind in allen Koordinaten fest, während die Seitenknoten sich in der Koordinate senkrecht zur Zeltkante entlang dem Boden bewegen können. Die äußere Kraft wirkt senkrecht in negativer z-Richtung auf die vier oberen Knoten des Zeltes gemaß $F_\lambda = \lambda \cdot 6 \times 10^5$ N. Durch die zwei Spiegelsymmetrien der Konstruktion, unter denen auch die angreifenden Kräfte invariant bleiben, ist ein noch reichhaltigeres Verzweigungsverhalten zu erwarten als bei dem Brusselator.

In der Tat ergibt sich in Abbildung 5.9, bei der die z-Verschiebung einer der oberen Knoten über den Parameter aufgetragen ist, bei Pfadverfolgung über einem Parameterbereich von $\lambda \in [-10, 20]$ eine Vielzahl von Verzweigungen, wobei die zusätzliche Einschränkung, daß die Verschiebungen in jeder Koordinate betragsmäßig nicht größer werden als 2 m, gemacht wurde. Anderenfalls würde das Verfahren u.a. auch noch die Lösungen finden, die durch Spiegelung der Lösungen an der xy-Ebene entstehen.

Bei Rechnungen ohne diese Verzweigungen ergibt sich nicht nur ein sehr viel mageres Diagramm, auch würde eine Stabilitätsanalyse an den Singularitäten mittels Bemerkung 2.2 zu einem falschen Ergebnis führen.

Nämlich wäre ohne Berechnung von Verzweigungen der Fehlschluß möglich, daß die Konstruktion einer Belastung von bis zu $\lambda \approx 15.28$, d.h. einer Kraft von $F_\lambda \approx 9170$ kN standhalten würde, da entlang dem aus der Ruhe-

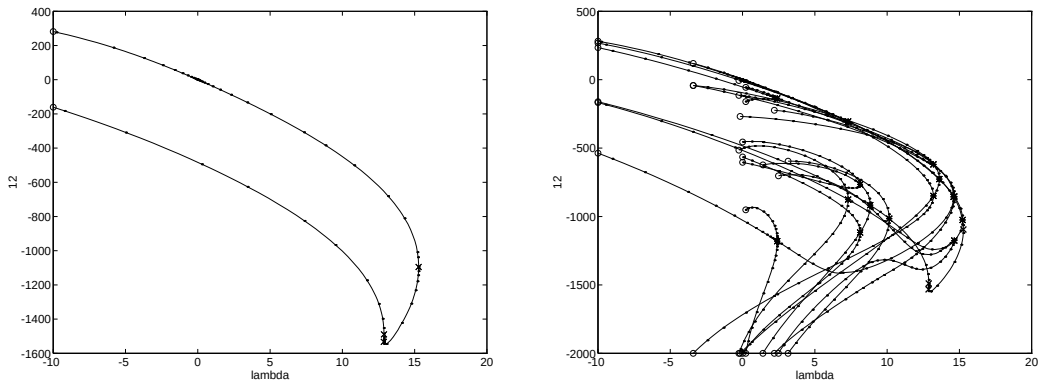


Abbildung 5.9: Berechnete Lösungen ohne und mit Verzweigungen

lage entspringenden Hauptpfad am Wendepunkt der erste erkennbare Stabilitätswechsel geschieht. Tatsächlich findet aber schon ein Stabilitätswechsel bei $\lambda \approx 13.62$ statt, d.h. bei einer Belastung von $F_\lambda \approx 8170$ kN, da sich dort die erste Verzweigung auf dem Hauptpfad befindet.

Da das nächstes Beispiel ein größeres Stabwerk mit denselben Symmetrieeigenschaften ist, wird an diesem übersichtlichen Beispiel der Zusammenhang zwischen Symmetrie und Verzweigungsverhalten dargestellt.

Die Konstruktion ist unter Spiegelungen an den horizontalen Ebenen durch die Längs- und Querachse des Zeltes symmetrisch; die Symmetriegruppe des Beispiels ist damit zu $\mathbb{Z}^2 \times \mathbb{Z}^2$ isomorph, indem z.B. $(1,0)$ mit der Spiegelung entlang der Längsachse und $(0,1)$ mit der Spiegelung entlang der Querachse identifiziert werden. Da die angreifenden Kräfte dieselbe Symmetrie besitzen, geht die Symmetrie auf die Gleichungen des Systems über.

Im Abbildung 5.10 sind die Lösungen des Hauptastes und der bei $\lambda \approx 13.62$ abzweigenden Äste bei $\lambda \approx 5.0$ dargestellt. In der obersten Reihe ist die Lösung auf dem Hauptast dargestellt; sie geht unter beiden Spiegelungen in sich selbst über, besitzt also die volle Symmetrie des Systems.

In der zweiten Reihe sind beide Lösungen der in der Heugabel-Verzweigung bei $\lambda \approx 13.62$ abzweigenden Äste aufgetragen. Beide Lösungen gehen bei einer Rotation um 180°, die $(1,1)$ entspricht, in sich selbst über und werden durch beide Spiegelungen aufeinander abgebildet.

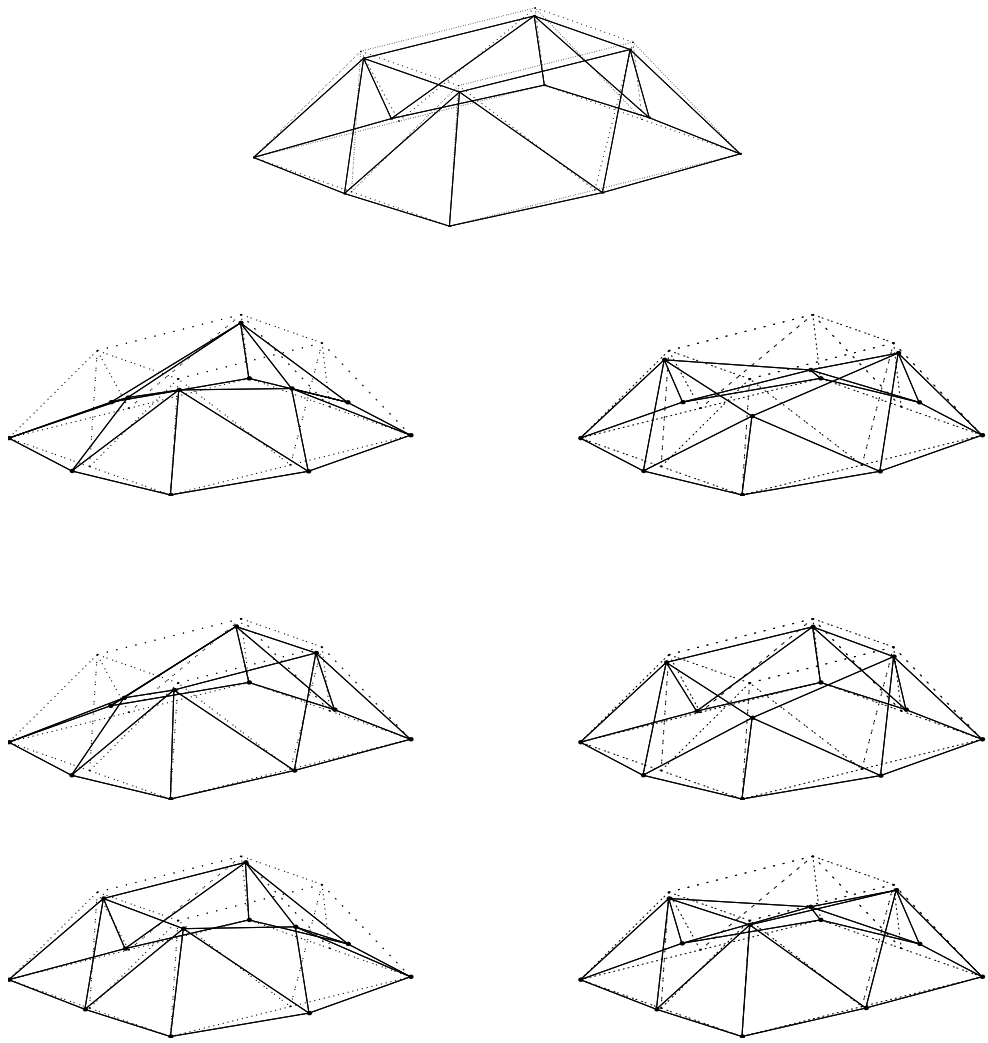


Abbildung 5.10: Lösungen von unterschiedlicher Symmetrie für $\lambda = 5.0$

Bei $\lambda \approx 13.20$ durchgehen beide Äste eine weitere Heugabel-Verzweigung, bei der auch die Rotationssymmetrie verlorengeht; die von jedem Ast abzweigenden Lösungen sind in den beiden Zeilen darunter abgebildet. Die abzweigenden Lösungen lassen sich durch Rotation ineinander überführen und werden durch Spiegelungen auf die abzweigenden Lösungen des jeweils anderen Astes abgebildet.

Ähnliches geschieht bei $\lambda \approx 14.60$, wo nur die Symmetrie $(1,0)$ erhalten bleibt, und bei $\lambda \approx 15.22$ kurz vor dem Wendepunkt, wo die Symmetrie $(0,1)$ erhalten bleibt. In beiden Fällen erleiden die abzweigenden Äste noch innerhalb des durchsuchten Parameterbereiches eine weitere symmetriebrechende Verzweigung, bei der völlig unsymmetrische Lösungen abzweigen.

Ein Hochspannungsmast

Als ein Beispiel einer Stabwerkkonstruktion von realistischer Dimension wird der Hochspannungsmast nach [18] gerechnet; er ist eine Modifikation des Beispiels aus [36]. Die Konstruktion ist in Figur 5.11 gezeichnet; die genauen Konstruktionsdaten sind allerdings in beiden Quellen nicht angegeben. Angegeben ist nur die Höhe des Mastes zu $47,24$ m und die Länge der unteren horizontalen Verstrebung zu 8 m. Mit der Annahme, daß sich die diagonalen Verstrebungen einer Konstruktionsebene im rechten Winkel schneiden, läßt sich die Höhe einer Konstruktionsebene berechnen, wenn der Steigungswinkel an der Ebene bekannt ist. Damit tritt eine Reduktion der Unbekannten in der Konstruktion auf, die, über den Daumen gepeilt, wie folgt gesetzt werden:

- Der Winkel an der Spitze ist $60°$.
- Die Abweichungen der Steigungswinkels von der Horizontalen α im unteren Teil und β im oberen Teil sind beide unbekannt; durch Festsetzen von $\beta = 0.6\alpha$ ist dann α implizit durch die Höhe gegeben.
- Die Winkel der Spitzen der Ausleger sind:

$$\begin{array}{rl} \text{für den mittleren Ausleger} & \gamma_1 = 10° \\ \text{für die anderen beiden Ausleger} & \gamma_2 = 16° \end{array}$$

Das nichtlineare Gleichungssystem, das den Abstand der Konstruktionsebenen zum Boden h_i und α als Variablen und die Berechnung der Höhen der Konstruktionsebenen $h_{i+1} - h_i = f(\alpha, h_i)$ als Gleichungen besitzt, wurde

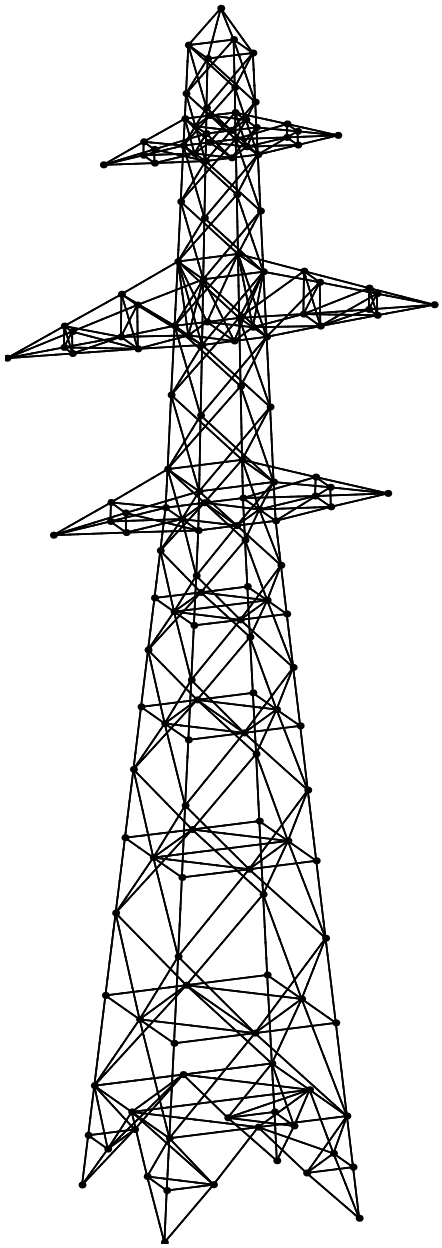


Abbildung 5.11: Konstruktionszeichnung des Hochspannungsmastes

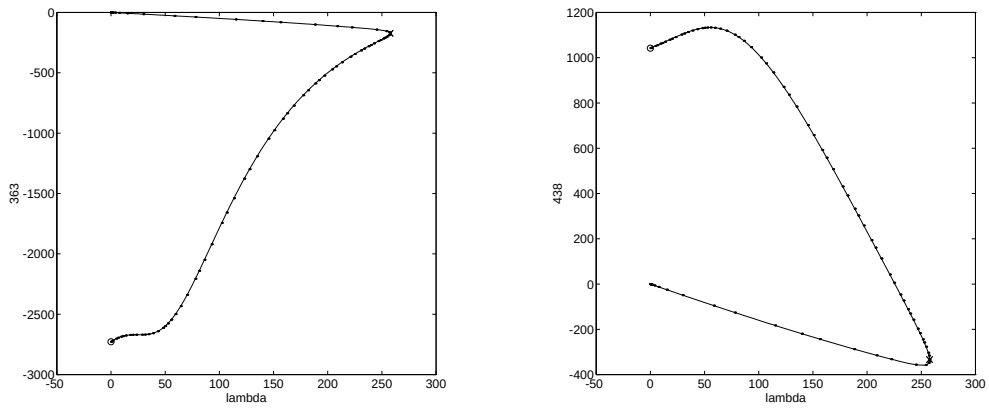


Abbildung 5.12: Verzweigungsdiagramm für den unsymmetrisch belasteten Mast

durch Einbettung (1.2) gelöst, wobei als Einbettungsparameter h_{top} gewählt wurde und von $h_{top} = (100 + 4 * \sin(60°))$ m, was $\alpha = 0$ entspricht bis $h_{top} = 47,24$ m, was $\alpha \doteq 5°$ ergab, verfolgt wurde. Das Programm `Alcon-S` benötigte dazu nur drei Schritte.

Die Konstruktion, so wie sie in [36] angegeben ist, ist als Stabkonstruktion nicht stabil, da einige als durch Knoten hindurchgehende Balken in der Stabkonstruktion an diesen Knoten ohne Krafteinwirkung knicken können. Gehrke hat in [18] diesen Effekt im Detail analysiert; seinem Vorschlag folgend wurden zur Stabilisierung in die ursprüngliche Konstruktion einige Stäbe eingefügt, wonach die Konstruktion aus 167 Knoten und 541 Stäben besteht.

Der Stäbe werden als unterschiedlich dick angenommen; die vertikalen tragenden Verstrebungen bis zum obersten Ausleger haben eine Querschnittsfläche von $\emptyset = 6400$ mm^2, die Stäbe der Ausleger inklusive der sie verbindenen horizontalen Verstrebungen im Mast und die diagonalen Stäbe $\emptyset = 1764$ mm^2 und die restlichen horizontalen Stäbe sowie die des Aufbaus $\emptyset = 529$ mm^2.

Durch das Gewicht der an den Spitzen der Ausleger angebrachten Stromkabel wird der Mast belastet. Zunächst wird angenommen, daß der Mast an einem Hang steht; dadurch greifen die Gewichtskräfte der Kabel in einer Weise an, die die Symmetrie der Konstruktion zerstört; die Kraft ist

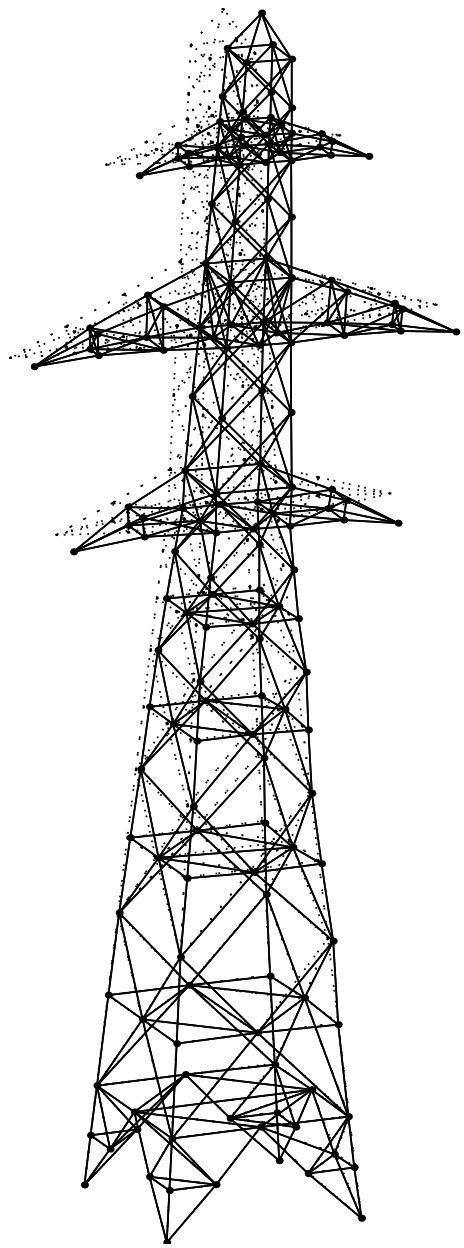


Abbildung 5.13: Deformation des Mastes am Wendepunkt

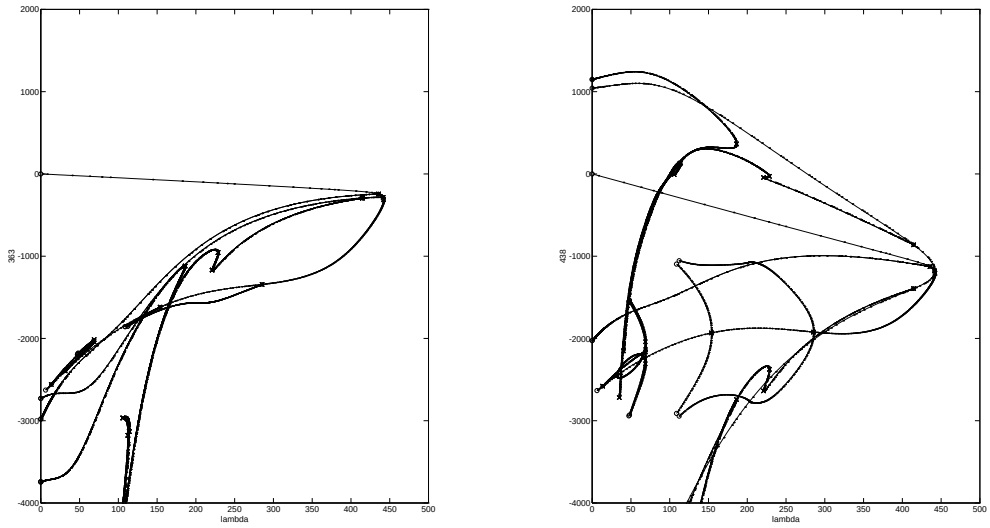


Abbildung 5.14: Verzweigungsdiagramm für den symmetrisch belasteten Mast

$\vec{F}_\lambda = \lambda \cdot (1000\ \mathrm{N}, 500\ \mathrm{N}, -10000\ \mathrm{N})$. Über die Spitze des Hochspannungsmastes wird das leichtere Erdungskabel geführt; dort greift eine Gewichtskraft von einem Zehntel der auf die Ausleger wirkenden Kraft an.

Das Lösungsdiagramm bei einer Pfadverfolgung über $\lambda \in [0, 400]$ ist in Figur 5.12 gegeben; dabei ist auf der linken Seite das Verhalten der z-Komponente der Spitze des Mastes und auf der rechten das Verhalten der z-Komponente der Spitze des linken Auslegers über λ aufgetragen.

Hinter dem Wendepunkt, der bei $\lambda \doteq 257.6$ auftritt, ist innerhalb des numerisch zugänglichen Lösungspfades kein weiterer Wendepunkt zu beobachten, so daß die Konstruktion bei Belastung jenseits des Wendepunktes vermutlich in sich zusammenfällt. Die Kondition des Problems verschlechtert sich auf dem instabilen Lösungszweig, so daß der Lösungspfad numerisch für Werte von $\lambda \ll 0$ in eine Ebene von Lösungen übergeht, auf der das Fortsetzungsverfahren hilflos umherirrt und regelrecht versumpft.

Durch die Art der angreifenden Kraft, die für den Mast eine Richtung, in die er knickt, auszeichnet, wird die Symmetrie der Konstruktion gestört. An keiner Stelle kann sich der Mast „entscheiden“, in welche Richtung er knickt.

Um ein Verzweigungsdiagramm zu bekommen, das auch wirklich Verzweigungen enthält, muß die Kraft symmetrisch angreifen.

Das folgende Verzweigungsdiagramm 5.14 ist für eine angreifende Kraft von $\vec{F}_\lambda = \lambda \cdot (0, 0, -10000 \text{ N})$ an den Auslegern und einem Zehntel von $\vec{F}_\lambda$ an der Spitze berechnet. Es ergibt sich ein ähnliches Verzweigungsverhalten wie für das Zelt; entlang der Pfade werden beide Symmetrien nacheinander gebrochen. Aufgrund der Unterschieds in der Konstruktion finden anders als beim Zelt nur zwei Brechungen der ersten Symmetrie vor dem Wendepunkt statt; insgesamt ergibt sich kein so reichhaltiges Verzweigungsverhalten, weil auch bei symmetrisch angreifenden Kräften sich die Konditionsverschlechterung bemerkbar macht.

Für die symmetrische Belastung ist in zwei Abbildungen dargestellt, wie sich der Mast bei einer Belastung von $\lambda = 400$ verhält, wobei zum einen die obere Lösung des symmetrischen Astes und zum anderen eine gänzlich unsymmetrische Lösung vom Nebenast des an der zweiten Verzweigung abzweigenden Astes angegeben sind.

Als Fazit läßt sich behaupten, daß die Berechnung von Verzweigungen auch bei Beispielen hoher Dimension für die Einsicht in die Lösungsstruktur eine wesentliche Bereicherung darstellt, insbesondere bei der Berechnung von Problemen mit (einfachen) Symmetrieen.

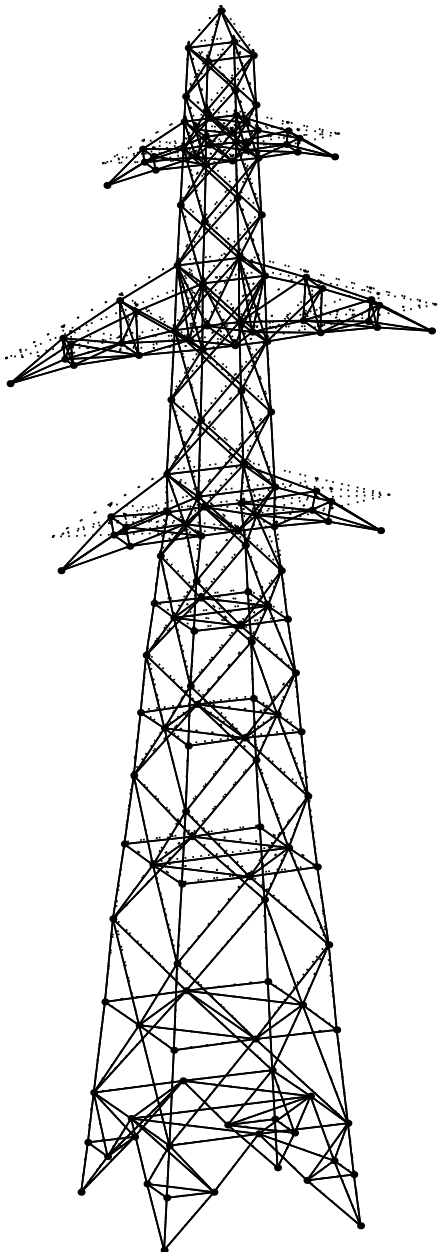


Abbildung 5.15: Symmetrische Deformation des Mastes

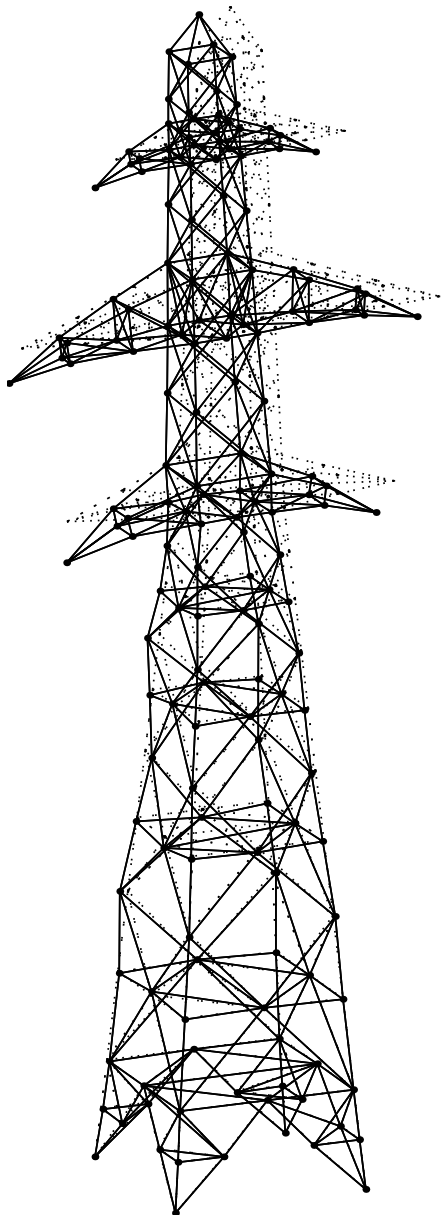


Abbildung 5.16: Unsymmetrische Deformation des Mastes

A. Die tangentiale Determinante einer $(n, n+1)$-Matrix

Hier wird der noch ausstehende Beweis des Satzes 2.1 aus Abschnitt 2.4 zur Definition der Determinante einer $(n, n+1)$-Matrix nachgetragen, der dort weggelassen wurde, um die Kontinuität der Textes nicht durch eher technische Überlegungen zu unterbrechen.

Zunächst wird noch einmal der Satz angeführt:

Satz 2.1: *Sei für eine beliebige $(n, n+1)$ Matrix A mit (normiertem) Tangentialvektor t die Determinante als*

$$det(A) := (-1)^i \frac{det(A^i)}{t_i} \qquad \textit{für } i \textit{ bel. mit } t_i \neq 0 \tag{2.9}$$

definiert.

Dann gilt

a) Die Definition ist von i unabhängig.

b) $rank(A) = n \iff det(A) \neq 0$.

c) Für jede (n, n) Matrix B gilt

$$det(BA) = det(B)\, det(A)$$

d) Für jede $(n+1, n+1)$-Matrix B gilt

$$\|B\big|_{B^{-1}(\mathcal{N}(A))}\| det(AB) = det(A)\, det(B)$$

Zur Schreibweise wird noch einmal daran erinnert, daß für eine Matrix A mit A_i die i-te Spalte und mit mit A^i die Matrix bezeichnet wird, die durch Streichen der i-ten Spalte entsteht. Im Beweis des Satzes wird eine ähnliche Schreibweise für Vektoren eingeführt; ist t ein n-Vektor, für den t_i die i-te Komponente bezeichnet, dann bezeichnet t^i den $n-1$-Vektor, der durch Streichen der i-ten Komponente des Vektors t entsteht. Damit läßt sich der Beweis formulieren:

Beweis: Zum Teil a) des Beweises, der die Wohldefiniertheit der Determinante (bei eindeutig orientierter Tangente) gewährleistet, führt die Beobachtung, daß die Eigenschaft der Tangente $At = 0$ auch als

$$A^i t^i = -A_i t_i$$

geschrieben werden kann, zu einem (n, n)-Gleichungssystem, welches sich bei vollem Rang mit Hilfe der Cramerschen Regel lösen läßt. Diese besagt, daß die j-te Komponente von t^i $(j \neq i)$ durch

$$t_j = \frac{\det(A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_{j-1}, -t_i A_i, A_{j+1}, \ldots, A_{n+1})}{\det(A^i)} \tag{A.1}$$

gegeben ist. Wird nun zur Berechnung der oberen Determinante der Faktor $-t_i$ herausgezogen, dann werden $|j - i - 1|$ Vertauschungen von benachbarten Spalten benötigt, um die an der j-ten Stelle stehende Spalte A_i zwischen die Spalten A_{i-1} und A_{i+1} zu bewegen; nach diesen Vertauschungen ist aber die Matrix A^j entstanden. Unter Berücksichtigung der durch die Vertauschungen entstandenen Vorzeichenwechsel ergibt sich aus (A.1)

$$t_j = (-1)^{i-j} t_i \frac{\det(A^j)}{\det(A^i)}$$

woraus sich durch Einsetzen von j anstelle von i in Gleichung (2.9) für $t_j \neq 0$ dasselbe Ergebnis für $\det(A)$ ergibt.

Falls $t_i \neq 0$, aber $\operatorname{rank}(A^i) < n$, dann folgt aus Teil b), daß $\operatorname{rank}(A) < n$ und daher $\det_t(A) = 0$ unabhängig von i.

Der Beweis von Teil b) benötigt dabei die Unabhängigkeit von i nicht. Ist $\operatorname{rank}(A) < n$, dann auch $\operatorname{rank}(A^i) < n$, also $\det(A^i) = 0$ für jedes i, un somit $\det_t(A) = 0$. Falls aber $\det(A^i) = 0$ für ein i mit $t^i \neq 0$, dann ist existiert wegen $\operatorname{rank}(A^i) < n$ ein weiterer, von t unabhängiger Kernvektor $\bar{t}$, der durch Einfügen einer Null an der i-ten Stelle in die Lösung von $A^i \bar{t}^i = 0$ entsteht. Damit ist $\operatorname{rank}(A) < n$.

Teil c) folgt aus der Beobachtung, daß $BA_i = (BA)_i$ und daher auch $BA^i = (BA)^i$, des weiteren ist ein Tangentialvektor t von A natürlich auch ein Tangentialvektor von BA. Demzufolge ist

$$\det(BA) = (-1)^i \frac{\det(BA^i)}{t_i} = (-1)^i \frac{\det(B)\det(A^i)}{t_i} = \det(B)\det(A)$$

Teil d) ist etwas umständlicher. Daher wird von vornherein der rangdefekte Fall $\det(AB) = 0$ ausgeschlossen. Denn dann ist wegen $\operatorname{rank}(AB) < n$ entweder $\operatorname{rank}(A) < n$ oder $\operatorname{rank}(B) < n+1$; in beiden Fällen bleibt nichts zu beweisen.

Sei also der ominöse Faktor $\|B|_{B^{-1}(\mathcal{N}(A))}\|$ betrachtet; er trägt der Tatsache Rechnung, daß B sich auf $\mathcal{N}(A)$ beliebig verändern kann, ohne das Produkt AB zu ändern. Da $\operatorname{rank}(A) < n$ ausgeschlossen, spannt jeder Tangentialvektor t von A den Tangentialraum $\mathcal{N}(A)$ auf; ist $\mathcal{N}(A) \subseteq \mathcal{R}(B)$, dann existiert ein $\hat{t}$ mit $B\hat{t} = t$; anderenfalls ist B eingeschränkt auf $B^{-1}(\mathcal{N}(A)) = \{0\}$, der Nulloperator und daher $\|B|_{B^{-1}(\mathcal{N}(A))}\| = 0$. In diesem Fall ist aber $\dim(\mathcal{R}(B)) < n+1$, also auch $\det(B) = 0$, wodurch die zu beweisende Gleichung wieder aus $0 = 0$ besteht. Ist hingegen $\operatorname{rank}(B) = n+1$ und daher $\mathcal{N}(A) \subseteq \mathcal{R}(B)$, dann ist das Urbild von $\mathcal{N}(A)$ auch eindimensional und wird von $\hat{t}$ aufgespannt; daher errechnet sich

$$\left\|B\big|_{B^{-1}(\mathcal{N}(A))}\right\| = \max_{x \in \mathcal{R}(\hat{t})} \frac{\|Bx\|}{\|x\|} = \frac{\|t\|}{\|\hat{t}\|} = \frac{1}{\|\hat{t}\|} \tag{A.2}$$

Der Beweis der Gleichung $\det(AB) = \det(A)\det(B)$ unter Ausschluß der rangdefekten Fälle läuft nun über zwei Tricks. Der eine besteht darin, ähnlich wie im Beweis von Teil a) die Cramersche Regel anzuwenden, aber diesmal auf die Gleichung $B\hat{t} = t$, was nach einigen Spaltenvertauschungen zu der Gleichung

$$\hat{t}_j = (-1)^j \frac{\det(\ t\ B^j)}{\det(B)} \tag{A.3}$$

führt. Der andere Trick besteht darin, die Multiplikativität der Determinante für quadratische Matrizen auf das Produkt

$$\begin{pmatrix} t^T \\ A \end{pmatrix} \begin{pmatrix} t & B^j \end{pmatrix} = \begin{pmatrix} 1 & t^T B^j \\ 0 & AB^j \end{pmatrix}$$

anzuwenden, was die Gleichung

$$\det \begin{pmatrix} t^T \\ A \end{pmatrix} \det \begin{pmatrix} t & B^j \end{pmatrix} = \det\left(AB^j\right) \tag{A.4}$$

produziert. Die erste Determinante auf der linken Seite berechnet sich durch Entwicklung nach der ersten Zeile zu

$$\det \begin{pmatrix} t^T \\ A \end{pmatrix} = \sum_{i=1}^{n+1} (-1)^i t_i \det(A^i) = \underbrace{\|t\|^2}_{=1} \det(A) \tag{A.5}$$

die zweite ergibt sich aus Gleichung (A.3) zu $\det(t\ B^j) = (-1)^j \hat{t}_j \det(B)$.

Wird bei der Matrix der ersten Determinante durch $n-1$ Vertauschungen von benachbarten Zeilen der Tangentialvektor nach unten getauscht, ergibt sich gerade die alternative Definition der Determinante aus Bemerkung 2.8; aus Gleichung (A.5) ist die in dieser Bemerkung gemachte Behauptung ersichtlich, daß beide Definitionen bis auf den Faktor $(-1)^{n-1}$ gleich sind, der durch die Zeilenvertauschungen entsteht.

Damit berechnet sich die Determinante $\det(AB)$ zu

$$\begin{aligned} \det(AB) &:= (-1)^j \frac{\det(AB^j)}{\hat{t}_j / \|\hat{t}\|} = (-1)^j \frac{\det(A)\,(-1)^j \hat{t}_j \det(B)}{\hat{t}_j / \|\hat{t}\|} \\ &= \|\hat{t}\| \det(A) \det(B) \end{aligned}$$

womit unter Berücksichtigung von Gleichung (A.2) alles gezeigt ist. □

Symbole

$\mathbb{R}$	reelle Zahlen
$\mathbb{Z}$	ganze Zahlen
$\mathbb{N}$	natürliche Zahlen
$\lvert x \rvert$	Absolutbetrag einer Zahl
$[\omega]$	Schätzwert für die Größe ω

<u>Für Vektoren des Hilbertraums $\mathbb{R}^n$:</u>

$\langle \cdot, \cdot \rangle$	kanonisches Skalarprodukt, $\langle x, y \rangle = \sum x_i y_i$
$\langle \cdot, \cdot \rangle_D$	durch die reguläre Matrix D induziertes Skalarprodukt, $\langle x, y \rangle_D := \langle Dx, Dy \rangle$
x^*	Adjungierter Vektor zu x; das Skalarprodukt läßt sich mit Hilfe von x^* als $\langle x, y \rangle = x^* y$ schreiben.
$\lVert x \rVert$	kanonische Norm, $\lVert x \rVert := \sqrt{x^* x}$

<u>Spezielle Operatoren</u>

Id_n	Identische Abbildung des $\mathbb{R}^n$, $Id\, x = x$
$\mathrm{diag}(d_1, \ldots, d_n)$	Diagonalmatrix $d_{ii} = d_i$; $d_{ij} = 0$ für $i \neq j$

<u>Für Teilräume U eines Hilbertraums V</u>

$\dim(U)$	Dimension von U
$U^\perp$	Orthogonales Komplement zu U, $U^\perp := \{x \in V \mid \forall y \in U \; y^* x = 0\}$
$\mathcal{P}_U$	Orthogonaler Projektor auf U, $\mathcal{P}_U x = x \;\; \forall x \in U$; $\mathcal{P}_U x = 0 \;\; \forall x \in U^\perp$
${\mathcal{P}_U}^\perp$	Projektor auf das orthogonale Komplement von U, ${\mathcal{P}_U}^\perp := \mathcal{P}_{U^\perp} = Id - \mathcal{P}_U$

Für lineare Operatoren $A : V \to W$:

A^T	Transponierte eines in Matrixform vorliegenden Operators, $(a_{ji})^T = (a_{ij})$
A^*	Adjungierter Operator, $\langle A^* x, y \rangle = \langle x, Ay \rangle$
$\mathcal{N}(A)$	Kern von A, $\mathcal{N}(A) := \{x \in V \mid Ax = 0\}$
$\mathcal{R}(A)$	Bild von A, $\mathcal{R}(A) := \{y \in W \mid \exists x \; Ax = y\}$
$\|A\|$	Operatornorm von A, $\|A\| := \max\limits_{\|x\|=1} \|Ax\|$
$\operatorname{rank}(A)$	Rang von A, $\operatorname{rank}(A) = \dim(\mathcal{R}(A))$
A^{-1}	Inverse des (regulären) Operators A
A^+	Pseudoinverse des Operators A
$\mathcal{P}_A$	Orthogonaler Projektor auf das Bild von A, $\mathcal{P}_A := \mathcal{P}_{\mathcal{R}(A)}$

Literaturverzeichnis

[1] D.G. Aronson; E.J. Doedel; H.G. Othmer *An Analytical and Numerical Study of the Bifurcations in a System of Linearly-coupled Oscillators* Physica 25(D):20-104, 1987

[2] R.E. Bank; T.F. Chan *PLTMGC: A Multi-Grid Continuation Program for Parametrized Nonlinear Elliptic Systems* SIAM J.Sci.Stat. Comput. 7:540-559, 1986

[3] E. Barragy; G.F. Carey *Bifurcation Detection Using the Lanczos Method and Imbedded Subspaces* IMPACT 3:76-92, 1991

[4] Å. Björk; I.S. Duff *A Direct Method for the Solution of Sparse Linear Leasts Squares Problems* Lin.Alg.Appl. 34:43-67, 1980

[5] G.C. Broyden *A Class of Methods for Solving Nonlinear Simultaneous Equations* Math.Comput. 19:577-593, 1965

[6] T.F. Chan *On the Existence and Computation of LU-Factorizations with Small Pivots* Math.Comput. 42:535-547, 1984

[7] T.F. Chan; D.C. Resasco *Generalized deflated block elimination* SIAM J.Numer.Anal. 23:913-924, 1986

[8] P. Deuflhard *Newton Techniques for Highly Nonlinear Problems – Theory and Algorithms* (to be published)

[9] P. Deuflhard; B. Fiedler; P. Kunkel *Efficient Numerical Pathfollowing beyond Critical Points* Technical Report 278, Universität Heidelberg, 1984

[10] P. Deuflhard; B. Fiedler; P. Kunkel *Efficient Numerical Pathfollowing beyond Critical Points* SIAM J.Numer.Anal. 18:949-987, 1987

[11] P. Deuflhard; G.Heindl *Affine Invariant Convergence Theorems for Newton's Method and Extensions to Related Methods* SIAM J.Numer.Anal. 16:1-10, 1979

[12] P. Deuflhard, A. Hohmann *Numerische Mathematik – eine algorithmisch orientierte Einführung, 2.Auflage* de Gruyter Berlin:New York, 1993

[13] P. Deuflhard; W. Sautter *On Rank-Deficient Pseudoinverses* Lin.Alg.Appl. 29:91-111, 1980

[14] L. Dieci; J. Lorenz; R.D. Russel *Numerical Calculation of Invariant Tori* SIAM J.Sci.Stat.Comput. 12:607-647, 1991

[15] I.S. Duff `MA28` - *A Set of FORTRAN Subroutines for Sparse Unsymmetric Linear Equations* Harwell Report A.E.R.E.-R.8730, 1980

[16] A.M. Erisman; J.K. Reid *Monitoring the Stability of the Triangular Factorization of a Sparse Matrix* Numer.Math. 22:183-186, 1974

[17] K. Gatermann; A. Hohmann *Symbolic Exploitation of Symmetry in Numerical Pathfollowing* IMPACT 2:330-365, 1991

[18] E. Gehrke *Ein Homotopieverfahren für Gleichungssysteme großer Dimension* Universität Heidelberg (Diplomarbeit), 1989

[19] G.H. Golub: Ch.F. van Loan *Matrix Computations (2nd ed.)* The John Hopkins University Press, 1989

[20] M. Golubitsky; D. Schaeffer *Singularities and Groups in Bifurcation Theory, Vol. 1* Series: Applied Mathematical Sciences, Springer Verlag, 1985

[21] W. Govaerts: J.D. Pryce *Block Elimination with One Refinement Solves Bordered Linear System Accurately* BIT 30:490-507, 1990

[22] M.R. Hestenes *Inversion of Matrices by Biorthogonalization and Related Results* SIAM J.Appl.Math. 6:51-90, 1958

[23] H. Heuser *Lehrbuch der Analysis* Teubner, Stuttgart, 1980

[24] N.J. Higham; D.J. Higham *Large Growth Factors in Gaussian Elimination With Pivoting* SIAM J.Matrix Anal.Appl. 10:155-164, 1989

[25] A. Hohmann; C. Wulff *Modular Design of Extrapolation Codes* Technical Report TR 92-5, Konrad-Zuse-Zentrum Berlin, 1992

[26] H.B. Keller *The Bordering Algorithm and Path Following near Singular Points of Higher Nullity* SIAM J.Sci.Stat.Comput. 4:573-582, 1983

[27] B.N. Lundberg; A.B. Poore *Variable Order Adams-Bashforth Predictors with an Error-Stepsize Control for Continuation Methods* SIAM J.Sci.Stat.Comput. 12:695-723, 1991

[28] E.H. Moore *On the Reciprocal of the General Algebraic Matrix (Abstract)* Bull.Amer.Math.Soc. 26:394-395, 1920

[29] G. Moore *The Numerical Treatment of Non-Trivial Bifurcation Points* Numer.Funct.Anal.Optimiz. 2:441-472, 1980

[30] C.C. Paige *An Error Analysis of a Method for Solving Matrix Equations* Math.Comput. 27:355-359, 1973

[31] R. Penrose *A Generalized Inverse for Matrices* Cambridge Phil.Soc. 51:406–413, 1955

[32] G. Peters; J.H. Wilkinson *The Least Squares Problem and Pseudo-inverses* Comput.J. 13:309-316, 1970

[33] I.Prigorin, R.Lefever *Symmetry Breaking Instabilities in Dissipative Systems II* J.Chem.Phys. 48:1695-1700, 1968

[34] W. Sautter *Fehleranalyse für die Gauß-Elimination zur Berechnung der Lösung minimaler Länge* Numer.Math. 30:165-184, 1978

[35] H.G. Schuster *Deterministic Chaos* VCH, Weinheim, 1989

[36] H.R. Schwarz *Methode der finiten Elemente* Teubner, Stuttgart, 1984

[37] R.D. Skeel *Scaling for Numerical Stability in Gaussian Elimination* J.Assoc.Comput.Mach. 26:494-526, 1979

[38] G.W. Stewart *On the Implicit Deflation of Nearly Singular Systems of Linear Equations* SIAM J.Sci.Stat.Comput. 2:136-140, 1981

[39] E.F. van de Velde, J. Lorenz *Adaptive Data Distribution for Concurrent Continuation* Numer.Math. 62:269-294, 1992

[40] C. Wulff *Numerische Pfadverfolgung von periodischen Lösungen mit Symmetrie* FU Berlin (Diplomarbeit), 1993

[41] Werner *Praktische Mathematik I* Springer, Heidelberg 1970

[42] G. Zielke *Lineare Gleichungssysteme und verallgemeinerte Inversen: Grundlagen und numerische Verfahren* Wiss.Z.Univ.Halle XXXX.1:45-59, 1991