Diplomarbeit

Numerische Verfolgung von Gleichgewichtslagen dynamischer Systeme-Stabilitätsanalyse und Lösungsdiagramme mit praktischen Anwendungen

von
Yang Xie

Technische Universität Ilmenau, 2009

Inhaltsverzeichnis

1 Einleitung

Diese Arbeit ist die numerische Verfolgung von Gleichgewichtslagen dynamischer Systeme - Stabilitätsanalyse und Lösungsdiagramme mit praktischen Anwendungen. Die Thematik ist abgeleitet von der mehrjärigen Lehr- und Forschungstätigkeit zu Numerik zeitkontinuierlicher dynamischer Systeme. Dynamische Systeme wurden als mathematische Objekte eingeführt, die die zeitliche Entwicklung existierender Systeme in Natur-, Ingenieur-, und anderen Wissenschaften beschreiben. Zeitkontinuierliche Systeme werden dabei häufig durch gewönliche Differenzialgleichungen modelliert.

Zentraler Gegenstand sind Parameterabhängige Differentialgleichungssysteme mit reellem Parameterverktor $\lambda \in \mathbb{R}^m$

$$\dot{x} = f(x, \lambda) \quad \text{mit} \quad f : \mathbb{R}^n \times \mathbb{R}^m \longrightarrow \mathbb{R}^n.$$

Derartige Probleme treten z.B. bei zeitkontinuierlichen neuronalen Netzwerken (Neurodynamik), elektrischen Netzwerken (Energietechnik), chemischen Reaktionsproblemen (Reaktions-Chemie), Knochenumbau-Prozessen (Medizin) und der Stabilität von Flugkörpern (Aircraftstability) auf und zeigen eine große Vielfalt an stabilen und instabilen Gleichgewichtslösungen.

Gleichgewichtspunkte eines Systems sind die Nullstellen der Differentialgleichungen, mit denen das System beschrieben wurde. Damit sind sie die Stellen, bei denen keine sichtbaren Änderungen mehr erfolgen. Alle Änderungen laufen zwar nach wie vor ab, gleichen sich aber gegenseitig aus. Wenn man nur eine einzige Population betrachten würde, wäre das dynamische Gleichgewicht die Stelle, an der sich die Geburten und die Sterbefälle vollkommen ausgleichen. Obwohl nach wie vor Individuen sterben und geboren werden, ist auf der Ebene der Gesamtpopulation keine Änderung mehr erkennbar. Gleichgewichte dieser Art werden auch als Fließgleichgewichte bezeichnet. Betrachten wir ein Beispiel des logistisches Wachstums:

$$\dot{x}(t) = rx(t)(k - x(t))$$

So sind die Gleichgewichtspunkte: $x_1(t) = 0$ und $x_2(t) = k$.

- Lokale Stabilität

Hier werden nur Aussagen für eine Umgebung des Gleichgewichtpunktes gemacht. Eine lokale Stabilität liegt vor, wenn man sich ein wenig vom Gleichgewichtspunkt entfernt, wieder in diesen zurückkehrt. Anders ausgedrückt: Stört man die Ruhelage des Systems leicht, so wird die Störung ausgeglichen.

• Globale Stabilität

Ein Gleichgewichtspunkt ist global stabil, wenn das System immer in ihm mündet.

• Asymptotische Stabilität

Das System erreicht den Gleichgewichtspunkt. Es gibt auch Gleichgewichtspunkte, die von dem System sozusagen umkreist und nicht erreicht werden.

Die Untersuchung auf globale Stabilität ist häufig umständlich. Die lokale kann leichter untersucht werden.

In nächsten Kapitel berechnen wir die Gleichgewichtslösungen parameterabhängiger Systeme, die zugleich Fixpunkte des Flusses darstellen. Die dabei gewonnenen numerischen Techniken werden sich als grundlegend erweisen.

2 Parametrisierung von Lösungskurven und Fortsetzungsmethoden

Hier betrachtet man nur ein parameterabhängiges Vektorfeld $f \in \mathcal{C}^r(\mathbb{R}^{n+1})$ mit $r \geq 1$, und das autonome System gewöhnlicher Differenziagleichungen (DGLn),

$$\dot{x} = f(x, \lambda), f : D \times \mathbb{R}^m \to \mathbb{R}^n, D \subset \mathbb{R}^n, \text{offen}, \dot{x} \equiv \frac{dx}{dt} \tag{2.1}$$

wird so definiert. Dabei ist $\lambda \in \Lambda$ ein skalarer reeller Parameter aus einem vorgegebenen endlichen Intervall $\Lambda = [a, b]$.

2.1 Grundlagen

Punkte $x_0 \in D$ bezeichnet man als Fixpunkte der DGL oder des Flusses.

Definition 2.1 (Gleichgewichtspunkt, Attraktor, Repellor)

(i) Ein Punkt $x_0 \in D$ heißt Gleichgewichtspunkt, Fixpunkt oder kritischer Punkt der DGL (2.1) zu festem Parameterwert λ_0, wenn $f(x_0, \lambda_0) = 0$ ist.

(ii) Der Gleichgewichtspunkt x_0 ist ein Attraktor, falls eine Umgebung $\mathcal{U}_(x_0) \subset D$ von x_0 existiert, so dass für jede Lösung $x(t)$ von (2.1) mit $x(0) \in \mathcal{U}(x_0)$ stets $\lim_{t\to\infty} x(t) = x_0$ folgt. Gilt diese Eigenschaft für $t \to -\infty$, so ist x_0 ein Repellor.

x_0 ist ein Gleichgewichtspunkt zu festem Parameterwert λ_0 und $\varphi_t : D \to \mathbb{R}^n$ die Abbildung des Flusses der DGL(2.1),so ist offenbar $\varphi_t(x_0) = x_0, \forall t \in \mathbb{R}$. Damit ist x_0 auch ein Fixpunkt des Flusses, und die zugehörige konstante Lösung $x(t) = \varphi_t(x_0)$ wird als Gleichgewichtslösunng bzw. Ruhrlage des dynamischen Systems bezeichnet.

Wegen des Existenz- und Eindeutigkeitssatz für die DGL (2.1) kann ein Gleichgewichtspunkt in unendlicher Zeit erreicht werden. Deshalb wird man zu seiner Bestimmung eine Integration der DGL vermeiden, insbesondere wenn x_0 ein Attraktor (bzw.Repellor) ist. Wir gehen direkt von Definition 2.1(i) aus und bestimmen parameterabhängige Gleichgewichtspunkt als Lösungen des Gleichungssystems.

$$f(x,\lambda) = 0, f : D \times \Lambda \subset \mathbb{R}^{n+1} \to \mathbb{R}^n \tag{2.2}$$

mit skalarem Parameter

$$\lambda \in \Lambda = [a,b].$$

Beispiel 2.2

Der Meteorologe E.N.Lorenz untersuchte zur Wetterprognose ein vereinfachtes Modell von Grundgleichungen
$\dot{x}_1 = x_2 - x_1$
$\dot{x}_2 = \lambda x_1 - x_2 - x_1 x_3$
$\dot{x}_3 = x_1 x_2 - x_3$
mit dem positiven Kontrollparameter λ. Dann zeigt dieses System trotz scheinbar einfacher Struktur ein erstaunlich reichhaltiges Lösungsverhalten.

Die Variablen x und λ wird zu einer einzigen Variablen $y = (x,\lambda) \in \mathbb{R}^{n+1}$ zusammengefasst und behandelt das System (2.2) als unterbestimmtes Gleichungssystem

$$f(y) = 0, f : \mathbb{R}^{n+1} \to \mathbb{R}^n, f \subseteq \mathcal{C}^r(\mathbb{R}^{n+1}) \quad \text{mit} \quad r \geq 1 \quad \text{in} \quad \mathbb{R}^{n+1} \tag{2.3}$$

Die Lösungsmenge $\mathcal{M}$ besitzen alle $n \times (n+1)$-Matrix $f(y)$, die Jacobimatrix $Df(y)$ alle Lösung von $f(y)$ hat, d.h.:

$$\mathcal{M} = \{y \subseteq \mathbb{R}^{n+1} | f(y) = 0, \text{Rang}(Df(y)) = n\} \tag{2.4}$$

Definition 2.3 (Regulärer Punkt, Regulärer Wert)

(i) Sei ein Punkt $y \in \mathbb{R}^{n+1}$ und die glatte Abbildung $f : \mathbb{R}^{n+1} \to \mathbb{R}^n$. y heißt regulärer Punkt der Abbildung f, falls die Jacobi-Matrix $Df(y) \in \mathbb{R}^{(n+1)\times n}$ vollen Rang besitzt, d.h. $\text{Rang}(Df(y)) = n$. y heißt singulärer Punkt, falls y nicht regulär ist.

(ii) Ein Bildpunkt $\omega \in \mathbb{R}^n$ heißt regulärer Wert von f, falls alle Menge $f^{-1}(\omega)$ aus regulären Punkten von f besteht, d.h.:

$$f^{-1}(\omega) := \{y \in D \times \Lambda \subset \mathbb{R}^{n+1} | f(y) = \omega\}$$

ω heißt singulärer Wert, falls ω nicht regulär ist.
$0 \in \mathbb{R}^n$ ist ein reguläre Wert von f ist, die Jacobimatrix $Df(y)$ vollen Rang für alle Lösungen von $f(y) = 0$ besitzt. Mit den Lösungsmenge $\mathcal{M}$ beweisen wir die Menge alle regulären Lösungen von (2.3).

Beispiel 2.4
Sei $y = (x_1, x_2, \lambda) = (y_1, y_2, y_3)$

$$f(y) = \begin{pmatrix} y_2 - y_1 \\ y_1 + y_2 y_3 - y_1 y_3 \end{pmatrix}$$

Die Jacobi-Matrix lautet:

$$Df(y) = \begin{pmatrix} -1 & 1 & 0 \\ 1 & y_3 & -y_1 \end{pmatrix}$$

Der Lösungszweig $L_0 : y_0 = (0, 0, y_3)$ erfüllt die Bedingung $RangDf(y_0) = 2$, wenn $y_3 \equiv \lambda \neq -1$ ist. Und ist $0 \in \mathbb{R}^2$ kein regulärer Wert von f. Der Punkt $y_0 = (0, 0, -1)$ ist singulär.

2.2 Natürliche Parametrisierung

Wenn die Teilmatrix $D_x f(x_0, \lambda_0) \in \mathbb{R}^{n \times n}$ für einen Lösungspunkt $y_0 = (x_0, \lambda_0)$ regulär ist, kann man den natürliche Parameter λ zur lokalen Parametrisierung der Lösungsmenge $\mathcal{M}$ benutzen. Wegen des impliziten Funktionentheorem existiert eine lokale Funktion $x : [\lambda_0 - \varepsilon, \lambda_0 + \varepsilon] \to \mathbb{R}^n, \varepsilon > 0$ und $(x(\lambda), \lambda) \in \mathcal{M}$. Falls $D_x f(x, \lambda)$ regulär für alle $(x, \lambda) \in D \times \Lambda$, ist $x(\lambda)$ auf ganz Λ definiert.

Satz 2.5
f sei $\mathcal{C}^r$ - glatt mit $r \geq 2$ und regulärem $D_x f(x, \lambda)$ für $(x, \lambda) \in D \times \Lambda$. Dann existiert die implizit definierte Funktion $x : \Lambda \to D$ mit folgenden Eigenschaften:

(i) $f(x, \lambda) = 0$ haben genau eine Lösung $x = x(\lambda) \in D$, für jedes $\lambda \in \Lambda$.

(ii) Es gilt sie Abschätzung $\|x - x(\lambda)\| \leq M_0 \cdot \|f(x,\lambda)\|$, $\forall (x,\lambda) \in D \times \lambda$, $M_0 > 0$.

(iii) $x(\lambda)$ ist stetig differenzierbar in $\lambda \in \Lambda$ und genügt den Einbettungsgleichung:

$$x'(\lambda) = -[D_x f(x(\lambda),\lambda)]^{-1} D_\lambda f(x(\lambda),\lambda) \tag{2.5}$$

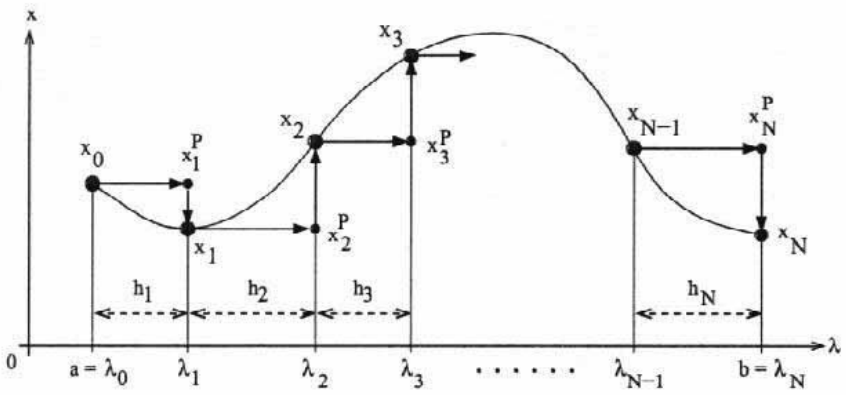


Abbildung 2.1 : Fortsetzung mit natürlicher Parametrisierung

Wegen der komplizierten rechten Seiten und der fehlenden Lösungsstabilität soll diese Einbettungsgleichg als Anfangswert Problem genommen werden.
Wir teilen wie in Abbildung (2.1) das Parameterintervall $\Lambda = [a,b]$ in N Teilintervalle mit den Teilpunkten $a = \lambda_0 < \lambda_1 < ..... < \lambda_N = b$ und bestimmen sukzessive die Punkte $(x(\lambda_j), \lambda_j)$ der Lösungskurve

$$\mathcal{L} = \{(x,\lambda) | (x,\lambda) \in D \times \Lambda \qquad \text{mit} \qquad f(x,\lambda) = 0\} \tag{2.6}$$

beginnend mit $(x(a), a) = (x_0, \lambda_0)$ in der Reihenfolge $(x_0,\lambda_0), (x_1,\lambda_1), (x_2,\lambda_2)$, $\ldots\ldots, (x_N,\lambda_N)$. Dabei abkürzen wir $x_j = x(\lambda_j)$.

Aufbau des Fortsetzungsschritt von λ_{j-1} bis λ_j wie folgt:
1. Gegeben ein Kurvenpunkt (x_{j-1}, λ_{j-1}).
2. Festlegung einer Fortsetzungsschrittweite $h_j > 0$ und des neuen Parameterwertes $\lambda_j = \lambda_{j-1} + h_j$.
3. Prädiktorschritt: Vorgabe eines Näherungswertes x^p für den neuen Lösungsvektor $x_j = x(\lambda_j)$.

4. Korrektorschritt: Iterative Verbesserung des Prädiktorwertes mittels eines Neutonähnlichen Verfahrens.

Für den Prädiktorschritt nutzen wir folgenden Informationen.
1. Basis-Prädiktor: den vorhergehenden Kurvenpunkt als Startwert für den folgenden Korrektorschritt:

$$x^P := x_{j-1} = x(\lambda_{j-1}) \tag{2.7}$$

2. Tangenten-Prädiktor (auch Euler-Prädiktor): Die Kurve wird durch ihre Tangente im vorhergehenden Kurvenpunkt ersetzt:

$$x(\lambda) = x(\lambda_{j-1}) + (\lambda - \lambda_{j-1}).x'(\lambda_{j-1}) = x_{j-1} + h_j.x'_{j-1} \tag{2.8}$$

Wobei x'_{j-1} mit der Einbettungsgleichung (2.5) genommen werden kann.
Setzen wir (2.5) in (2.8) an, bekommen wir eine neuene Prädiktorformel:

$$x^P := x_{j-1} - h_j.[D_x f(x_{j-1}, \lambda_{j-1})]^{-1} D_\lambda f(x_{j-1}, \lambda_{j-1})$$

3. Sekanten-Prädiktor: Man legt durch die 2 Konten (x_{j-2}, λ_{j-2}) und (x_{j-1}, λ_{j-1}) die Sekante, und extrapoliert zum Parameterwert λ_j :

$$x^P := x_{j-1} + \frac{\lambda_j - \lambda_{j-1}}{\lambda_{j-1} - \lambda_{j-2}}(x_{j-1} - x_{j-2}) \tag{2.9}$$

Es benötigt für x^P bereits an der Stelle λ_1 zwei Startwerte, die man sich mittels des Basis-Prädiktors verschaffen muss. Dieser Ansatz stellt einen präktikablen Zugang dar.

4. Polynomiale Extrapolation: Gegeben Kurvenpunkt (x_{j-1}, λ_{j-1}), $(x_{j-2}, \lambda_{j-2}), \ldots\ldots, (x_{j-k}, \lambda_{j-k})$.

Durch diesen Methode können wir sehr genaue Prädiktorformeln der allgemeinen Form gewinnen und den Sekantenprädiktor verbessern.

$$x^P := \alpha_{1j} x_{j-1} + \alpha_{2j} x_{j-2} + ... + \alpha_{kj} x_{j-k}, \qquad \alpha_{ij} \in \mathbb{R} \tag{2.10}$$

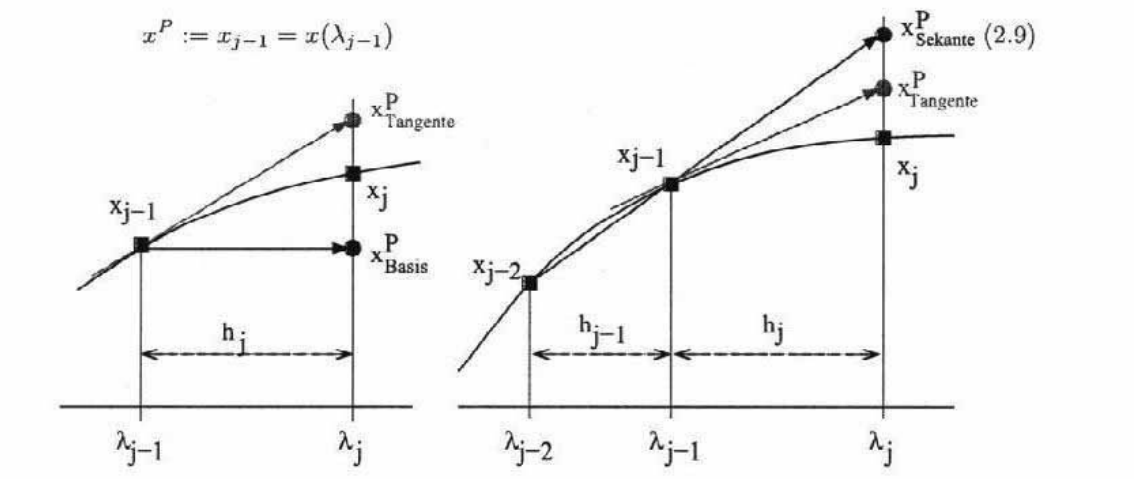


Abbildung 2.2 : Prädiktoren bei natürlicher Parametrisierung

Jetzt betrachten wir den Korrektorschritt. Wir benutzen häufig das Newton-Verfahren. Setzen wir $u_0 := x^p$ mit dem Prädiktorpunkt (x^p, λ_j) und bekommen die Formel der Näherungslösungen:

$$u_{k+1} := u_k - [D_x f(u_k, \lambda_j)]^{-1} f(u_k, \lambda_j), \qquad k = 0, 1, 2, \cdots, K-1$$

Durch die Umformung dieser Gleichung gewinnen wir eine neue Formel:

$$D_x f(u_k, \lambda_j)\Delta u_k = -f(u_k, \lambda_j) \quad \text{mit} \quad u_{k+1} := u_k + \Delta u_k, \quad k = 0, 1, \cdots, K-1. \tag{2.11}$$

Der ermittelte Punkt (x_j, λ_j) mit dem Wert $x_j := u_k$ liegt im Allgemein nicht auf der Kurve $\mathcal{L}$, da der Abbruch des Verfahren bereits nach endlich vielen Iterationen erfolgt. Wir benutzen oft Newton-ähnliche Verfahren mit konstanter Matrix $A_0 \sim D_x f(x^p, \lambda;)$, so dass nun die Form

$$A_0 \Delta u_k = -f(u_k, \lambda_j) \qquad \text{mit} \qquad u_{k+1} := u_k + \Delta u_k \tag{2.12}$$

entsteht.

Wobei

$$\{A_0\}_{ij} := \frac{1}{\Delta x_j}\{f_i(x_1^p, \cdots, x_j^p + \Delta x_j, \cdots, x_n^p, \lambda) - f_i(x^p, \lambda)\}, \qquad i, j = 1(1)n.$$

mit einer Differenzenapproximation der Matrixelemente.

Das Fortsetzungsverfahren erfordert den Nachweis der lokalen Konvergenz dieses Korrektorschrittes bei geeigetem Prädiktorwert. Damit ist die Endlichkeit der Anzahl von Fortsetzungsschritten über dem Parameterintervall $\Lambda = [a, b]$ zu begründen.

ALGORITHMUS 2.6 (Natürliche Parametrisierung)

Function $[N, \lambda, x] = \text{continuation } (f, Dfx, a, b, x_0, tol)$

S1 Setze $j := 0, \lambda_0 := a, x^P := x_0$
Wähle Schrittweite h sowie Iterationszahl k_{opt}.

S2 Startrechung
Bestimme $x_0 = x(\lambda_0)$ mit Verfahren (2.11) oder (2.12).

S3 Do while $\lambda_j \leq b$
1. Setze $j := j + 1$.
2. Setze $\lambda_j := \lambda_{j-1} + h$. Falls $\lambda_j > b$, so $\lambda_j := b$.
3. Prädiktor
Bestimme x^P mit Verfahren (2.7),(2.8) oder (1.9).
4. Korrektor
Bestimme $x_j = x(\lambda_j)$ mit Verfahren (2.11) oder (2.12) in k Schritten bis auf eine Genauigkeit tol.
5. Schrittweiten-Bestimmung
(a) Bestimme $\rho := \frac{k_{opt}}{k}$.
(b) Beschränke $\rho := \max\{\min(\rho, 2), \frac{1}{2}\}$.
(c) Setze $h := h * \rho$.
(d) Falls $h < \varepsilon_{Maschine}$, so STOP.

S4. Return $N := j, \lambda_j, x_j, j = 0(1)N$

2.3 Reguläre Lösungspfade

Satz 2.7 (Existenz einer glatten Parametrisierung)
Die Menge $D \subset \mathbb{R}^{n+1}$ sei nicht leer und offen. Die Funktion $f : D \to \mathbb{R}^n$ sei $2-mal$ stetig differenzierbar. Ferner sei $y_0 \in D$ eine reguläre Nullstelle von f, d.h. ein regulärer Punkt von f mit $f(y_0) = 0$.
Dann gibt es $s_0, \delta \in \mathbb{R}(\delta > 0)$ und eine glatte Kurve $\mathcal{L}$ mit Parameterdarstellung

$$y : J = (s_0 - \delta, s_0 + \delta) \to D, \tag{2.13}$$

so dass für alle $s \in J$ gilt:

(i) $y(s_0) = y_0$;

(ii) $f(y(s)) = 0$;

(iii) $Rang(f'(y(s))) = n$;

(iv) $y'(s) \neq 0$;

(v) $span(y'(s)) = kern(f'(y(s)))$;

(vi) $\begin{bmatrix} f'(y(s)) \\ y'(s)^T \end{bmatrix}$ ist regulär

$\mathcal{L}$ heißt Lösungspfad mit der Parametrisierung $y = y(s)$.

Folgerung 2.8
Setzen wir der Parametrisierung in die Gleichung ein und differenzieren nach s : $f'(y(s))\cdot y'(s) = 0$, d.h. der Vektor $y'(s)$ liegt im eindimensionalen Nullraum $N(f'(y(s)))$ und ist orthogonal zu alle n Zeilenvektoren dieser Matrix. Die erweiterte Jocobi Matrix ist

$$H'(y(s)) := \begin{pmatrix} f'(y(s)) \\ y'(s)^T \end{pmatrix} \quad \text{mit} \quad N(f'(y(s))) = \text{span}(y'(s)) \tag{2.14}$$

regulär auf dem Pfad $\mathcal{L}$ für $s \in J$.

Definition 2.9
Ein Lösungspfad $\mathcal{L} = \{y(s)|f(y(s)) = 0, \forall s \in J\}$ mit der Eigenschaft $Rang(f'(y(s))) = n$ für alle $s \in J$ wird als regulär bezeichnet.

Satz 2.10
Die Bedingung $Rang(f'(y)) = n$ ist genau dann erfüllt, wenn

(i) $D_x f(x, \lambda)$ regulär ist oder

(ii) $\dim N(D_x f(x, \lambda)) = 1$ und $D_\lambda f(x, \lambda) \notin R(D_x f(x, \lambda))$ gilt.

Beweis:(Notwendige Bedingung) Sei $f'(y) = A = (a_1, a_2, \cdots, a_n, a_{n+1})$. Wegen $RangA = n$ existieren n linear unabhängige Spalten a_i von A.
1.Fall: $D_x f(x, \lambda)$ ist linear unabhängig, dann heißt $D_x f(x, \lambda)$ regulär.
2.Fall: $D_x f(x, \lambda)$ ist linear abhaängig. Da $(a_1, \cdots, a_{n+1})$ n linear unabhängige Vektoren enthält, muß $i \in \{1, \cdots, n\}$ existieren, so daß

$$(a_1, \cdots, a_{i-1}, a_{i+1}, \cdots, a_n, a_{n+1})$$

linear unabhängig ist. Also folgt $\dim N(D_x f(x, \lambda)) = 1$ und $D_\lambda f(x, \lambda) = a_{n+1} \in R(D_x f(x, \lambda))$
(Hinreichende Bedingung)
1.Fall: Wenn $D_x f(x, \lambda)$ regulär ist, dann $RankA = n$.
2.Fall: Da $\dim N(D_x f(x, \lambda)) = 1$, so existieren $n-1$ linear unabhängige Spalten von $D_x f(x, \lambda) : \tilde{A} = (a_1, \cdots, a_{i-1}, a_{i+1}, \cdots, a_n)$. Da zudem $D_\lambda f(x, \lambda) \in R(D_x f(x, \lambda))$, so muß a_{n+1} linear unabhängig von $\tilde{A}$ sein, also ist $Rank(\tilde{A}, a_{n+1}) = Rank(f'(y)) = n$. □

Definition 2.11
Regulär Lösungen $y(s_0) = (x(s_0), \lambda(s_0)) \in \mathcal{L}$ erfüllen Satz 2.10 (i), wogegen eine Lösung in Falle (ii) als Umkehrpunkt (einfacher Grenzpunkt) bezeichnet wird.
Wegen des Voraussetzungens besteht jeder reguläre Lösungspfad nur aus regulären Punkten und einfachen Grenzpunkten.

Beispiel 2.12
$\dot{x} = x(x^3 - x - \lambda)$
$f(y) = y_1(y_1^3 - y_1 - y_2) = 0, \qquad y = (x, \lambda)$
$\mathcal{L} = \{(x, \lambda) | f(x, \lambda) = 0\} = \mathcal{L}_0 \cup \mathcal{L}_1$
Überprüfen wir jetzt das Satz 2.7:
(a) Betrachten $y^3 \in \mathcal{L}_1 : y^3 = (1, 0)$.
(i) $f \in \mathcal{C}^r(D \times \Lambda), r \leq 2, D = (0.9, 1.1), \Lambda = (-0.1, 0.1)$
(ii) $f(y^3) = 0$
(iii) $f'(y) = (4x^3 - 2x - \lambda, -x) \quad \text{mit} \quad f'(y^3) = (2, -1) \Longrightarrow Rang f'(y^3) = 1$
Da $Rang f_x(y^3) = Rang(2) = 1$, ist y^2 regulärer Kurvenpunkt. Sei $\lambda_0 = 0$, dann existiert $J = (-\delta, \delta), \delta > 0$ mit $y(0) = y^3, f(y(\alpha)) = 0, Rang f'(y(\alpha)) = 1, y'(\alpha) \neq 0$ für alle $\alpha \in J$ (lokale Parametrisierung von $\mathcal{L}$).
(b) Betrachten $y^1 = (x_1, \lambda_1) = (\frac{1}{3}\sqrt{3}, -\frac{2}{9}\sqrt{3})$.
(i) $f \in \mathcal{C}^2(D \times \Delta), D = (\frac{1}{3}\sqrt{3} - \epsilon, \frac{1}{3}\sqrt{3} + \epsilon), \Delta = (-\frac{2}{9}\sqrt{3} - \epsilon, -\frac{2}{9}\sqrt{3} + \epsilon), \epsilon > 0.$
(ii) $f(y^1) = 0$
(iii) $f'(y^1) = [0, -\frac{1}{3}\sqrt{3}] \Longrightarrow Rang f'(y^1) = 1$
Aber wegen $-\frac{1}{3}\sqrt{3} \neq \mathcal{C} \cdot 0 = 0, \mathcal{C} \in R$, dann kommt $Rang f_x(y^1) = 0$ und y^1 ist ein Umkehrpunkt. Lokale Parametrisierung $y(\alpha)$ liefert im Umgebung von y^1.
(c) Betrachte $y^0 = (x_0, \alpha_0) = (0, 0)$.
$f'(y^0) = (0, 0) \Longrightarrow Rang f'(y^0) = 0 < n$
$\Longrightarrow y^0$ sind kein regulärer und kein Umkehrpunkt, d.h. keine lokale Parametrisierung $y(\alpha)$ angebbar.

2.4 Bogenlängen-Parametrisierung

Obwohl jede Lösung $y = f(x, \lambda)$ regulär ist, versage der Fortsetzungsalgorithmus an der Stellen $y_0 = (x_0, \lambda_0)$, an denen $D_x f(x_0, \lambda_0)$ singulär wird.
Führen wir die Bogenlänge als Kurvenparameter ein, so genügt der Tangentenvektor in jedem Punkt der Beziehung

$$\|y'(s)\|_2 = 1, \forall s \in J \tag{2.15}$$

D.h. die Durchlaufgeschwindigkeit einer Kurve ist genau dann in jedem Punkt $s \in J$ gleich eins, wenn der Parameter s gleich der Bogenlänge ist. Der Tangentenvektor $y'(s)$ wird durch erweiterte System eindeutig bestimmt.

$$\begin{cases} f'(y(s)) \cdot y'(s) = 0 \\ y'(s)^T y'(s) - 1 = 0 \end{cases} \qquad \text{mit} \qquad \det \begin{pmatrix} f'(y(s)) \\ y'(s)^T \end{pmatrix} \geq 0 \tag{2.16}$$

Setzen wir für s_j die Lösung $y_j = y(s_j)$ ein, d.h. $y'(s_j)^T y'(s_j) - 1 = 0$. Linearisierung von (2.16) bzgl. $y'(s)$ bei y_j durch lokale Approximation der Bogenlänge liefert:

$$y'(s_j)^T \frac{y(s) - y(s_j)}{s - s_j} - 1 = 0$$

Wir multiplizieren die skalare Zusatzbedingung für y:

$$g(y) := y'(s_j)^T (y - y_j) - (s - s_j) = 0, \qquad g : \mathbb{R}^{n+1} \to \mathbb{R} \tag{2.17}$$

Das Ergebnis ist das erweiterte System für y mit der eingeführten skalarwertigen Funktion $g(y)$.

$$h(y) := \begin{pmatrix} f(y) \\ g(y) \end{pmatrix} = 0, \qquad h : \mathbb{R}^{n+1} \longrightarrow \mathbb{R}^{n+1} \tag{2.18}$$

Folgerung 2.13
Die Jakobi-Matrix dieses System $h'(y)$ ist regulär für $s \in J_0 = (s_j - \delta_0, s_j + \delta_0), \delta_0 > 0$:

$$h'(y(s)) := \begin{pmatrix} f'(y(s)) \\ g'(y(s)) \end{pmatrix} = \begin{pmatrix} f'(y(s)) \\ y'(s_j)^T \end{pmatrix} \tag{2.19}$$

Wegen der Stetigkeit von $f'(y)$ ist auch $h'(y(s))$ regulär.

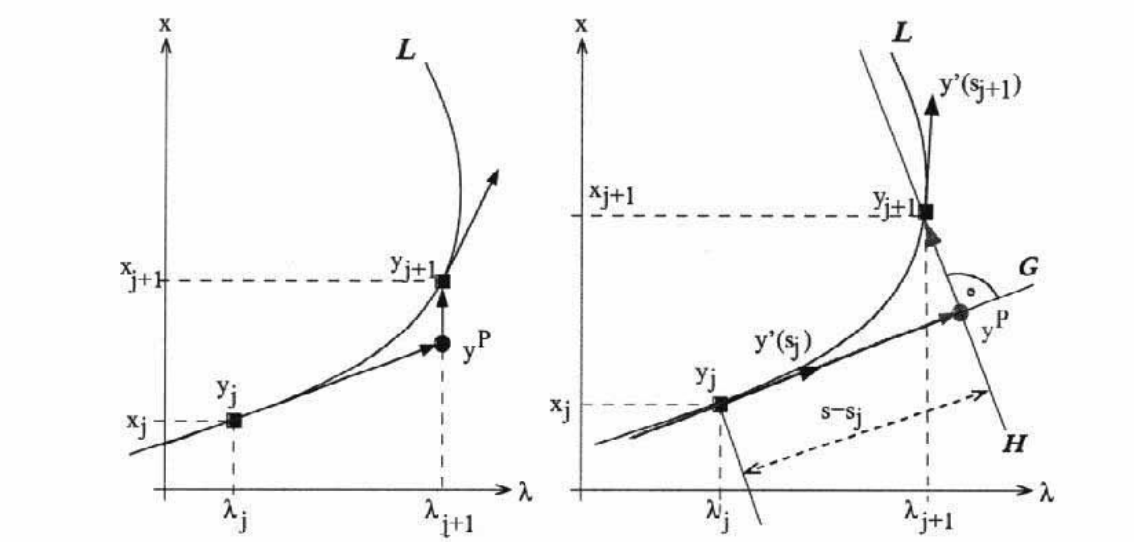


Abbildung 2.3 : Natürliche Parametrisierung (links) und Pseudo-Bogenlänge (rechts)

Jetzt lassen sich die Zusatzbedingung leichter interpretieren:
Wegen der Gleichung

$$y(\tilde{s}) = y_j + (s - s_j) \cdot y'(s_j) \tag{2.20}$$

den Abstand $s - s_j$ von y_j, da $\|y'(s_j)\|_2 = 1$ ist. Die Normal-Hyperebene H zu G durch y^P lautet: $y'(s_j)^T \cdot (y - y^P) = 0$.
Einsetzen des y^P in H:

$$y'(s_j)^T[y - y_j - (s - s_j)y'(s_j)] = y'(s_j)^T(y - y_j) - (s - s_j) = 0 = g(y)$$

d.h. $g(y)$ beschreibt die Normal Hyperebene H zu $y'(s_j)$, die den Abstand $s - s_j$ von y_j hat.

Bestimmung des normierten Tangentenvektors $y'(s_j)$:

Sei $A := f'(y(s_j)) \in R^{n\times(n+1)}$. Nach der Folgerung 2.8 ist $\dim N(A) = 1$ und $y'(s_j) \in N(A)$. Also ist ein Elment z des Nullraums von A zu ermitteln. Dies kann mittels LU-Zerlegung von A^T erreicht werden:

(1) LU-Zerlegung von $A^T \in R^{(n+1)\times n}$ mit Zeilenvertauschungen. Diese führt zu fol-

gender Darstellung:

$$PA^T = L \cdot \begin{pmatrix} U \\ \dots \\ 0^T \end{pmatrix} = \begin{pmatrix} 1 & & & \\ & 1 & & 0 \\ & & \ddots & \\ l_{ik} & & & 1 \end{pmatrix} \cdot \begin{pmatrix} u_{11} & & u_{ij} \\ 0 & \ddots & \\ & & u_{nn} \\ 0 & \cdots & 0 \end{pmatrix}$$

mit
$P \in R^{(n+1)\times(n+1)}$ – Permutationsmatrix
$L \in R^{(n+1)\times(n+1)}$ – untere Dreieck-Matrix
$U \in R^{n\times n}$ – obere Dreieck-Matrix.
Aus obiger Form folgt wegen $P \cdot P^T = I$

$$A^T = P^T L \cdot \begin{pmatrix} u \\ 0^T \end{pmatrix}$$

$$A = (U^T \vdots 0) \cdot L^T \cdot P$$

(2) Sei $e_{n+1} = (0, \dots, 0, 1)^T$ der $(n+1) - te$ Einheitsvektor. Setze

$$z := P^T (L^T)^{-1} e_{n+1}$$

Dann erhält man

$$A \cdot z = AP^T (L^T)^{-1} e_{n+1} = (U^T \vdots 0) \cdot \underbrace{L^T \underbrace{PP^T}_{I} (L^T)^{-1}_{e_{n+1}}}_{I}$$

$$A \cdot z = 0$$

zudem ist $z \neq 0$. (sonst müßte $e_{n+1} = 0$ sein.) Also ist $z \in N(A)$.
(3) Normierung von z liefert schließlich

$$y'(s_j) := \pm \frac{z}{\|z\|_2}$$

(4) Bestimmung des Vorzeichs: Um einen Durchlaufwechsel zu vermeiden, betrachtet man 2 aufeinanderfolgende Tangenten $y'(s_{j-1})$ und $y'(s_j)$:
Das Ziel ist, daß das Winkel φ zwischen den Tangentenrichtungen sein muß, wegen $\cos\varphi > 0$:

$$y'(s_{j-1})y'(s_j) = \|y'(s_{j-1})\| \cdot \|y'(s_j)\| \cdot \cos\varphi > 0$$

Man setze also

$$y'(s_j) := + \frac{z}{\|z\|_2}$$

und bilde

$$\tau := (y'(s_{j-1}), y'(s_j)) = y'(s_{j-1})^T y'(s_j)$$

Falls $\tau < 0$, so $y'(s_j) := -y'(s_j)$.

Wir fassen das Ergebnis zusammen und bekommen folgenden Algorithmus:

ALGORITHMUS 2.14 (Pesudo-Bogenlänge)

Function $[N, y] = \text{pseudoarclength}\ (f, Df, a, b, x, s\max, tol)$

S1 Setze $j := 0, \lambda_0 := a, x^P := x, s_0 := 0$
Wähle Schrittweite h sowie Iterationszahl k_{opt}.

S2 Startrechung
Bestimme $x_0 = x(\lambda_0)$ mit Verfahren (2.11) oder (2.12). Setze $y_0 = (x_0, \lambda_0)$.

S3. Do while ($a \leq \lambda_j \leq b$ and $s_j < s\max$)

1. Setze $j := j + 1$ und $s_j := s_{j-1} + h$.
2. Tangenten-Prädiktor
Bestimme y^P mittels (2.23).
3. Newton-Korrektor
Löse das erweiterte System (2.20) in R^{n+1} mit dem Newton-Verfahren oder Newton-ähnlichem Verfahren.
4. Schrittweiten-Anpassung
wie in Algorithmus (2.6)

S4 Return $N := j, y_j = (x_j, \lambda_j), j = 0(1)N$.
Dieser Algorithmus nutzt dafür den Tangentenprädiktor.

Satz 2.15
Mit diesen Voraussetzungen ist $y_j = y(s_j) \in \mathcal{L}$. Dann existiert ein $\delta > 0$, so dass für alle s mit $|s - s_j| < \delta$ das erweiterte System:

$$f(y) = 0$$
$$g(y) = y'(s_j)^T \cdot (y - y_j) - (s - s_j) = 0 \tag{2.21}$$

einen eindeutigen glatten Lösungszweig $y = y(s)$ besitzt. Die Jacobimatrix $h'(y(s))$ längs des Zweigs ist regulär.

Der Bogenlängen-Parameter wird in Abb.2.3 (rechts) anschaulich durch den Tangentenparameter s ersetzt, weshalb man von der Pseudo-Bogenlänge spricht. Bei der natürlichen Parametrisierung in Abb.2.3 (links) werden Umkehrpunkte leicht überwunden.

2.5 Gauß-Newton-Fortsetzung

Aus der Bogenlänge-Parametrisierung erkennen wir, dass bei zu groß gewählter Fortsetzungsschrittweite h die Bogenlänge keinen Schrittpunkt y_j liefern muss. Und folgender Schritt von Bogenlängen-Parameter s_{j-1} zu s_j ist naheliegend:

1. Gegeben sei ein Kurvenpunkt $y_{j-1} = y(s_{j-1})$. Festlegung einer geeigneten Fortsetzungsschrittweite $h_j > 0$ und des neuen Parameterwertes $s_j = s_{j-1} + h_j$.
2. Vorgabe eines Prädiktorpunktes $v_0 = v = y^P$ für den neuen Kurvenpunkt y_j mit

dem Tangentenprädiktor

$$y^P := y_{j-1} + h_j \cdot y'(s_{j-1}), \qquad \|y'(s_{j-1})\|_2 = 1 \tag{2.22}$$

oder dem Sekantenprädiktor

$$y^P := y_{j-1} + \frac{h_j}{h_{j-1}} \cdot (y_{j-1} - y_{j-2}) \tag{2.23}$$

3. Bestimmung eines Kurvenpunktes $y_j \in \mathcal{L}$ mit minimalem Abstand von $v = y^P$, d.h. iterative Lösung der Aufgabe in Korrektorschritt

$$\|y_j - v\|_2 = \min_{f(y)=0} \|y - v\|_2 \tag{2.24}$$

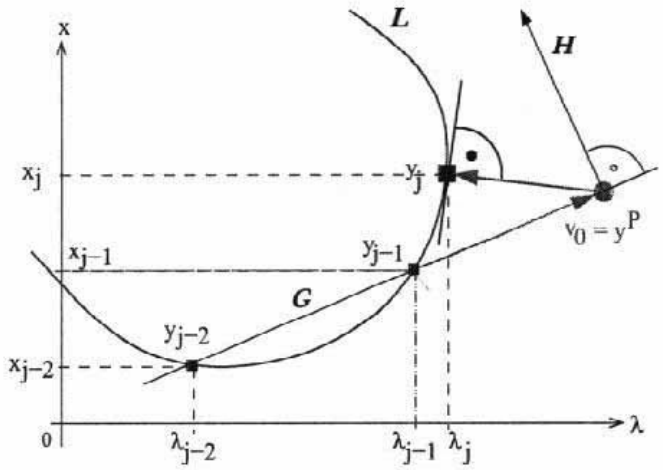


Abbildung 2.4 : Kurvenverfolgung mittels Projektion

Ein Minimalpunkt y_j durch die dargestellte Orthogonalitätsbedingung
$y'(s_j)^T \cdot (y_j - v) = 0$ mit den Tangentialvektor $y'(s_j)$ definiert, womit das bestimmende System für $y_j \in \mathbb{R}^{n+1}$ lautet:

$$\begin{aligned} &(i) \quad f(y) = 0 \\ &(ii) \quad y'(s_j)^T \cdot (y - v) = 0 \end{aligned} \tag{2.25}$$

Linearisierung dieses Systems am Nahrungspunkt $v \in \mathbb{R}^{n+1}$ mit zugehörigem Tangentenvektor $v'(s_j)$ liefert das lineare Gleichungssystem für den Wert v_{neu}:

$$f'(v) \cdot (v_{neu} - v) = -f(v)$$
$$v'(s_j)^T \cdot (v_{neu} - v) = 0 \tag{2.26}$$

Wenn man eine Lösung $x := v_{neu} - v$ des unterbestimmten linearen Gleichungssystem $Ax = b$ mit $A := f'(v), b := -f(v)$ findet, dann löst man die Minimierungsaufgabe mit

$$\|x^*\|_2 = \min_{Ax=b} \|x\|_2$$

Einsetzen $x^* = A^+ b$ mit A^+ - der Pesudoinversen von A, (m, n) - Matrix. Wegen $A \in \mathbb{R}^{n\times(n+1)}$ mit den Voraussetzungen des zeilenregulären Falles kann A^+ als Rechtsinverse direkt ausgerechnet werden.

$$A^+ = A^T(AA^T)^{-1} \qquad \text{wegen} \qquad AA^+ = AA^T(AA^T)^{-1} = I \tag{2.27}$$

Anwendung der Pesudoinvesen von $A := f'(v)$ auf (2.25) ergibt den Newtonschritt $v_{neu} = v - A^+ f(v)$ mit $A^+ = A^T(AA^T)^{-1}$ dessen iterative Wiederholung das Gauß-Neuton-Verfahren für unterbestimmte nichtlineare Gleichungssystem liefert:

$$v_{k+1} = v_k - A^T(AA^T)^{-1} f(v_k) \qquad \text{mit} \qquad A = f'(v_k), k = 0, 1, 2, \cdots \tag{2.28}$$

Wenn der Prädiktorwert des Startpunktes $v_0 = y^P$ hinreichend nahe am Lösungspfad $\mathcal{L}$ liegt, kann die quadratische Konvergenz dieses Verfahrens nachgewiesen werden.

ALGORITHMUS 2.16 (Gauß-Newton-Fortsetung)

Function $[N, y] = \text{gaussnewton}\ (f, Df, a, b, x, s\max, tol)$

S1 Setze $j := 0, \lambda_0 := a, x^P := x, s_0 := 0$
Wähle Schrittweite h sowie Iterationszahl k_{opt}.

S2 Startrechnung
Bestimme $x_0 = x(\lambda_0)$ mit Verfahren (2.11) oder (2.12). Setze $y_0 = (x_0, \lambda_0)$.

S3 Do while ($a \leq \lambda_j \leq b$ und $s_j < s\max$)
1. Setze $j := j + 1$ und $s_j := s_{j-1} + h$.
2. Sekannten-Prädiktor
Bestimme $v_0 := y^P$ mit Verfahren (2.22), falls $j > 1$, anderfalls mit Verfahren (2.21).
3. Gauß-Newton-Korrektor
(a) Setze $k := 0$.
(b) $A := Df(v_k) = f'(v_k)$
(c) Löse das reguläre lineare System

$$(AA^T) \cdot b = f(v_k), \qquad b \in \mathbb{R}^n$$

(d) $v_{k+1} := v_k - A^T b, k := k + 1$
(e) Falls $\|A^T b\|_2 > tol$, so gehe zu Schritt (b).
(f) Setze $y_j := v_k$.
4. Schrittweiten-Anpassung
wie in Algorithmus (2.6)

S4 Return $N := j, y_j = (x_j, \lambda_j), j = 0(1)N$

Konvergenz des Gauß-Newton-Verfahrens::

Der Startpunkt (Prädiktorwert) v_0 ist hinreichend nahe dem Lösungspfad $\mathcal{L}$, so kann die quadratische Konvergenz des unterbestimmten Gauß-Newton-Verfahren (2.28) nachgewiesen werden.

Satz 2.17
Für $f : D \subset \mathbb{R}^{n+1} \to \mathbb{R}^n$ seien die Voraussetzungen erfüllt. Dann existiert ein offene Umgebung $U(\mathcal{L}) \subset \mathcal{D}$ des Lösungspfads, mit der folgende Behauptungen gelten:

(i) Zu jedem $v \in U(\mathcal{L})$ existiert genau ein $y^* \in \mathcal{L}$ mit $\|y^* - v\|_2 = \min_{f(y)=0} \|y - v\|_2$. Die dadurch definiert Abbildung $s : U(\mathcal{L}) \to \mathcal{L}$ ist glatt.

(ii) Ist $v_0 \in U(\mathcal{L})$, so auch $v_1 \in U(\mathcal{L})$.

(iii) Zu jedem $v_0 \in U(\mathcal{L})$ konvergiert die Folge $(v_k), k = 0, 1, 2, \cdots$ der Iterierten gemäß (1.27) gegen $v_\infty \in \mathcal{L}$.

(iv) Für $v_0 \in U(\mathcal{L})$ gelten folgende Abschätzungen gleichmäßig für $v_k \in U(\mathcal{L})$:

$$\begin{aligned}
\|v_2 - v_1\| &\leq C_1\|v_1 - v_0\|^2 \\
\|v_\infty - v_1\| &\leq C_2\|v_\infty - v_0\|^2 \\
\|v_1 - S(v_0)\| &\leq C_3\|v_0 - S(v_0)\|^2 \\
\|v_\infty - S(v_0)\| &\leq C_4\|v_0 - S(v_0)\|^2
\end{aligned}$$

Zu praktischen Umsetzung des Verfahrens ist die vollständige Jacobi-Matrix $Df = f'(y)$ vorzugeben. Mit dem Sekanten-Prädiktor, der die Berechung der Tangente $y'(s_{j-1})$ vermeidet, liefert der Gauß-Newton-Korrektor diesen Algorithmus zusammen.

2.6 Schrittweitensteuerung

Die Wahl der Fortsetzungsschrittweite $h_j = \lambda_j - \lambda_{j-1}$ ist oft endscheidend für den Erfolg der Fortsetungsmethode. Bei zu großem Wert kann der Prädiktorpunkt außerhalb des Einzugsbereiches der zu bestimmenden Lösung liegen. Dies führt meist zur Divergenz des Korrektors, mitunter auch zur Konvergenz gegen eine andere unerwünschte Lösung. Durchgehend kleine Schrittweiten h_j erhöhen andererseits die Zahl N der Fortsetzungsschritte und so den Gesamtaufwand über alle Maßen.
Eine einfache Strategie soll λ - Schrittweiten $h = h_j > 0$ wählen, die die Konvergenz des Korrektors garantiert und zugleich die Anzahl der Rechenoperation minimiert. Wir lassen sich der Rechenaufwand $A(h)$ durch die Anzahl

$$A(h) := k(h)/h$$

mit $k(h)$ - Korrektorschritten ansetzen. Mit einen optimalen Schritt $h_{opt} = \rho h$ wird der minimale Aufwand

$$A(h_{opt}) := k(h_{opt})/h_{opt}$$

erreicht, dann liefert Division:

$$\frac{A(h)}{A(h_{opt})} = \frac{k \cdot h_{opt}}{h \cdot k_{opt}} = \frac{k}{k_{opt}} \cdot \rho \geq 1 \qquad \text{mit} \qquad k := k(h), k_{opt} := k(h_{opt})$$

D.h. $\rho \geq k_{opt}/k$ sinnvoll. Geben wir eine optimale Anzahl von Korrektoriterationen k_{opt} vor, setzen wir also $\rho := k_{opt}/k$. Eine zusätzliche Begrenzung $\frac{1}{2} \leq \rho \leq 2$ kann die Sicherheit der Wahl von $h_{opt} = \rho \cdot h$ erhöhen. Und der Algorithmus (2.16) für das Prädiktor-Korrektor-Verfahren zur numerischen Lösungsfortsetzung passt die aktuelle Fortsetzungsschrittweite h_j in jedem Schritt an, so dass eine optimal Zahl k_{opt} von Korrektorschritten ausgeführt wird. Für das Newton-Verfahren hat sich $k_{opt} = 5$ oder $k_{opt} = 6$ bewährt.

3 Stabilitäts- und Bifurkationsanalyse

3.1 Stabilität von Gleichgewichtslagen

3.1.1 Stabilität des Gleichgewichts ($n = 1$)

Das System sei mit einem Freiheitsgrad, verallgemeinerte Koordinate φ, und das Potential $U = U(\varphi)$.
Die Gleichgewichtslage ist sicher dann stabil, wenn für beliebige virtuelle Verschiebungen $\delta_\varphi \neq 0$: $U(\varphi + \delta_\varphi) > U(\varphi)$ gilt.
Taylor-Entwicklung des Potentials $U = U(\varphi)$:

$$U(\varphi + \delta_\varphi) = U(\varphi) + \frac{1}{1!} \underbrace{\frac{dU}{d\varphi}}_{=0} \delta_\varphi + \frac{1}{2!}\frac{d^2U}{d\varphi^2}(\delta_\varphi)^2 + \frac{1}{3!}\frac{d^3U}{d\varphi^3}(\delta_\varphi)^3 + \cdots$$

Hinreichende Bedingung für Stabilität: $\frac{d^2U}{d\varphi^2} > 0$

3.1.2 Stabilität des Gleichgewichts ($n \geq 2$)

Das System sei mit n Freiheitsgraden, verallgemeinerte Koordinaten $q_1, q_2, \cdots, q_n$, und das Potential $U = U(q_1, q_2, \cdots, q_n)$.
Gleichgewichtslage ist sicher dann stabil, wenn für beliebige virtuelle Verschiebungen $\delta_{q_k} \neq 0$:

$$U(q_1 + \delta_{q_1}, q_2 + \delta_{q_2}, \cdots, q_n + \delta_{q_n}) > U(q_1, q_2, \cdots, q_n)$$

gilt.

Taylor-Entwicklung des Potentials $U = U(q_1, q_2, \cdots, q_n)$:

$$U(q_1 + \delta_{q_1}, q_2 + \delta_{q_2}, \cdots, q_n + \delta_{q_n}) = U(q_1, q_2, \cdots, q_n) + \tfrac{1}{1!} \sum_{i=1}^{n} \underbrace{\frac{\partial U}{\partial q_i}}_{=0} \delta_{q_i} +$$

$$\tfrac{1}{2!} \sum_{i=1}^{n} \sum_{j=1}^{n} \tfrac{\partial^2 U}{\partial q_i \partial q_j} \delta q_i \delta q_j + \cdots$$

Hinreichende Bedingung für Stabilität:

$$(\frac{\partial^2 U}{\partial q_i \partial q_j}) \qquad \text{positiv definit}$$

3.2 Stabilität von Ruhelagen

Wir setzen für das dynamische System

$$\dot{x} = f(x, \lambda), f : D \times \Lambda \to R^n \tag{3.1}$$

hinreichende glatt und das Vektorfeld f voraus:

$$f \in C^r(D \times \Lambda), r \geq 2, D \subset R^n, \Lambda \subset R, \qquad \text{offen} \tag{3.2}$$

Eine Trennung von Phasenraum D und Parameterraum Λ soll vorgenommen werden, d.h. es soll eine Parametrisierung der Ruhelagen von $f(x, \lambda)$ nach λ erfolgen.

In solchen Abschnitt werden singuläre Punkte stehen, die keine Umkehrpunkte sind. Und in diesem Kontext ist die Stabilität von Ruhelagen wesentlich.

3.2.1 Linearisierung des Vektorfeldes f:

Sei x_0 Ruhelage zu $\lambda \in \Lambda, \lambda$ fest, d.h. $f(x_0, \lambda) = 0$.
Betrachten das Verhalten von Orbits in einer kleinen Umgebung von x_0:

$$\mathcal{S}_\delta(x_0) = x \| x - x_0 \| < \delta.$$

Sei $x = x(t)$ eine Lösung der DGL (3.1). Für die Differenz $\xi(t) := x(t) - x_0$ gilt:

$$\begin{aligned} \dot{\xi}(t) = \dot{x} - \dot{x}_0 = f(x, \lambda) - f(x_0, \lambda) = f_x(x_0, \lambda)\xi + \frac{1}{2} f_{xx}(x_0, \lambda)(\xi, \xi) \\ + o(\|\xi\|^3) - f(x_0, \lambda) = f_x(x_0, \lambda)\xi + o(\|\xi\|^2) \\ \Longrightarrow \dot{\xi}(t) = A \cdot \xi + o(\|\xi\|^2) \qquad \text{mit} \qquad A := f_x(x_0, \lambda) \end{aligned} \tag{3.3}$$

Falls $\|\xi\| < \delta \ll 1$, so approximiert $A \cdot \xi$ die rechten Seiten gut.
Die Fehlergleichung lautet:

$$\dot{\xi} = A \cdot \xi, \qquad \text{mit} \qquad A := f_x(x_0, \lambda) = D_x f(x_0, \lambda) \tag{3.4}$$

Definition 3.1:
(i) Das System (3.4) heißt Linearisierung der DGL (3.1) bei x_0. Die lineare Funktion $A\xi = f_x(x_0, \lambda) \cdot \xi$ heißt der lineare Teil von f an der Stelle x_0.

(ii) x_0 heißt hyperbolische Ruhelage, falls für alle Eigenwert $\mu_i \in \sigma(A)$ gilt: $\mathrm{Re}\mu_i \neq 0, i = 1(1)n$.

Bemerkung 3.2:
Kann $x_0 = 0$ angenommen werden, so lautet (3.4): $\dot{x} = A \cdot x$.

3.2.2 Invariante Unterräume

Seien $\mu_1, \mu_2, \cdots, \mu_n \in \sigma(A)$, die n Eigenwerte von $A := f_x(x_0, \lambda)$, wobei k reelle und $m - k$ komplexe Eigenwerte existieren, wie folgt angeordent seien:

$$\begin{cases} \mu_1 = & \alpha_1 \\ \mu_2 = & \alpha_2 \\ \vdots & \vdots \\ \mu_k = & \alpha_k \end{cases} \quad k \quad \text{reelle Eigenwerte}$$

$$\begin{cases} \mu_{k+1} = & \alpha_{k+1} + i\beta_{k+1} \\ \mu_{k+2} = & \alpha_{k+2} + i\beta_{k+2} \\ \vdots & \vdots \\ \mu_m = & \alpha_m + i\beta_m \end{cases} \quad 2 \cdot (m-k) \quad \text{komplexe Eigenwerte} \quad \mu_i \quad \text{und} \quad \overline{\mu_i}, \beta_i \neq 0$$

muss $2(m-k) + k = 2m - k = n$ sein.

Zugehörige verallgemeinste Eigenvektoren sind:

$$\begin{cases} w_1 = & u_1 \\ w_2 = & u_2 \\ \vdots & \vdots \\ w_k = & u_k \end{cases} \quad 2n \quad \mu_i, \quad i \leq k$$

$$\begin{cases} w_{k+1} = & u_{k+1} + iv_{k+1} \\ w_{k+2} = & u_{k+2} + iv_{k+2} \\ \vdots & \vdots \\ w_m = & u_m + iv_m \end{cases} \qquad 2n \quad \mu_i, \quad i > k$$

Folgerung 3.3:
Die Menge der $2m - k = n$ Vektoren $B = \{\mu_1, \mu_2, \cdots, \mu_k, \mu_{k+1}, v_{k+1}, \mu_{k+2}, v_{k+2}, \cdots, \mu_m, v_m\}$ bildet eine Basis des R^n.

Definition 3.4:
Die Untervektorräume E^s, E^u, E^c mit

$$\begin{aligned} E^s &= span\{\mu_i, v_j | Re\mu_i < 0\} \\ E^u &= span\{\mu_i, v_j | Re\mu_i > 0\} \\ E^c &= span\{\mu_i, v_j | Re\mu = 0\} \end{aligned} \tag{3.5}$$

heißen stabiler, instabiler und zentraler Unterraum des linearen Systems (3.4) mit den Dimensionen $n_s = \dim E^s, n_u = \dim E^u, n_c = \dim E^c, n_s + n_u + n_c = n$.

Satz 3.5:
(i) $E^s \oplus E^u \oplus E^c = R^n$

(ii) E^s, E^u, E^c sind invariant unter dem Fluß φ_t des linearen System $\dot{x} = Ax$.

Folgerungen 3.6:
(i) x_0 hyperbolischer Ruhepunkt $\Longleftrightarrow n_c = 0$ und $n_s + n_u = n$.

(ii) x_0 Senke (Quelle) $\Longleftrightarrow n_u = n_c = 0$ und $n_s = n$. (oder $n_s = n_c = 0$ und $n_u = n$)

(iii) x_0 Sattel $\Longleftrightarrow n_c = 0, n_s > 0, n_u > 0$.

Bemerkung 3.7:
Für E^s und E^u gelten folgende Eigenschaften:
(i) Ist $x_0 \in E^s \backslash \{0\}$, so $\lim_{t \to +\infty} \varphi_t(x_0) = 0$ und $\lim_{t \to -\infty} \|\varphi_t(x_0)\| = \infty$.

(ii) Ist $x_0 \in E^u \backslash \{0\}$, so $\lim_{t\to-\infty} \varphi_t(x_0) = 0$ und $\lim_{t\to+\infty} \|\varphi_t(x_0)\| = \infty$.

Beispiel 3.8:
Sei $\dot{x} = x(x^3 - x - \lambda)$ Betrachte Ruhelage $x_0 = 0$ für $\lambda \in [-1, 1]$.
$f(x, \lambda) = x(x^3 - x - \lambda)$
$\Longrightarrow f_x(x, \lambda) = 4x^3 - 2x - \lambda$
$\Longrightarrow f_x(x_0, \lambda) = -\lambda$
Damit $\mu = \mu(\lambda) = -\lambda$ einziger Eigenwert von $f_x(x_0, \lambda)$.
1. Fall: $\lambda = 0 \Longrightarrow Re\mu = 0$
$\Longrightarrow n_c = 1, n_u = n_s = 0$
$\Longrightarrow x_0$ nicht hyperbolisch
2. Fall: $\lambda < 0 \Longrightarrow Re\mu > 0$
$\Longrightarrow n_u = 1, n_s = n_c = 0$
$\Longrightarrow x_0$ Quelle
3. Fall: $\lambda > 0 \Longrightarrow Re\mu < 0$
$\Longrightarrow n_s = 1, n_u = n_c = 0$
$\Longrightarrow x_0$ Senke
Folglich gilt $\mu(\lambda)$ bei $\lambda = 0$ durch die imaginäre Achse; wegen $\mu'(\lambda) = -1 \neq 0$ verlauft diese Bewegung des Eigenwertes gleichförmig.

3.2.3 Ljapunov-Stabilität

Definition (3.9):
Eine Ruhelage x^* einer autonomen DGL heißt stabil (im Sinne von Ljaounov), wenn es zu jedem $\varepsilon > 0$ ein $\delta = \delta(\varepsilon) > 0$ gibt, mit folgenden Eigenschaft: Für jeden Anfangswert x_0 mit $\|x_0 - x^*\| < \delta$ existiert die Lösung $x(t; x_0)$ für alle $t \geq 0$ und genügt der Abschätzung

$$\|x(t; x_0) - x^*\| < \varepsilon \qquad \text{für alle} \qquad t \geq 0 \tag{3.6}$$

Anderfalles heißt die Ruhelage x^* instabil.

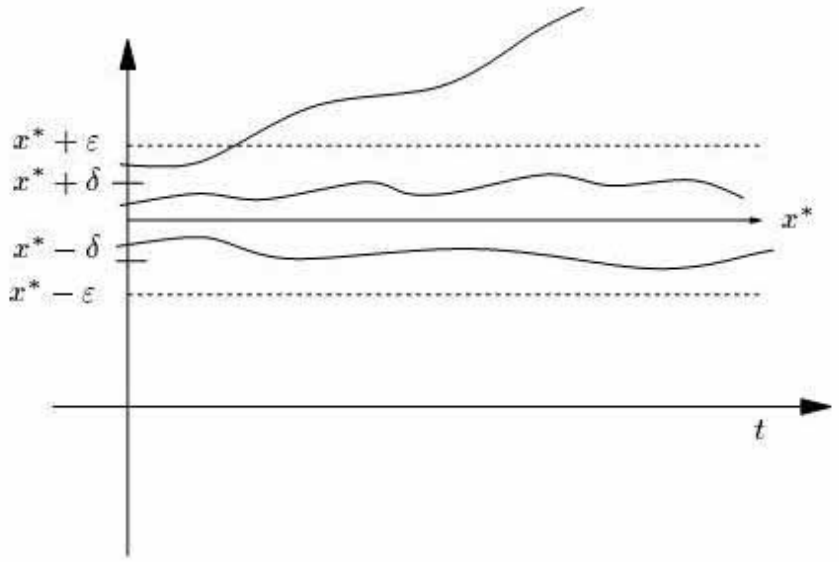


Abbildung 3.1

Beispiel (3.10):
Betrachte die lineare Skalare DGL $\frac{dx}{dt} = x$ mit die Lösung $x(t; x_0) = x_0 e^{\alpha t}$. Falls $\alpha \neq 0$ ist die Ruhelage $x^* \equiv 0$ eindeutig und zusätzlich:
stabil: falls $\alpha < 0$. Nimm $\delta = \varepsilon$!
instabil: falls $\alpha > 0$, weil für jedes $\varepsilon > 0$ gilt für

$$x_{0,n} = \frac{1}{n} \Longrightarrow \|x(t, x_0, n)\| \geq 0, \forall t \geq t_n = \ln(n\varepsilon)$$

Bemerkung (3.11):
Von der stetigen Abhängigkeit bzg. des Anfangswertes haben wir immer: $\exists \delta = \delta(\varepsilon, T) > 0$ für jedes $\varepsilon > 0$ und $0 < T < \infty$, so dass

$$\|x_0 - x^*\| < \delta \Longrightarrow \|x(t; x_0) - x(t; x^*)\| < \varepsilon, \forall 0 \leq t \leq T \tag{3.7}$$

Im obigen Beispiel genügt die Ruhelage x^* einer stärkeren Eigenschaft für $\alpha < 0$: $x(t, x_0) \to x^*$ für $t \to \infty$, d.h. einer Attraktivitätseigenschaft.

Definition (3.12):
Eine Ruhelage x^* heiße attraktiv, wenn es ein $\delta_0 > 0$ gibt mit folgender Eigeschaft: Für jeden Anfangswert x_0 mit $\|x_0 - x^*\| < \delta_0$ existiert die Lösung $x(t; x_0)$ für alle

$t \geq 0$ und es gilt

$$\lim_{t \to \infty} \|x(t; x_0) - x^*\| = 0 \tag{3.8}$$

Beispiel (3.13):
Die nichtlineare DGL $\frac{dx}{dt} = x - x^2$ hat Ruhelagen 0 und 1.

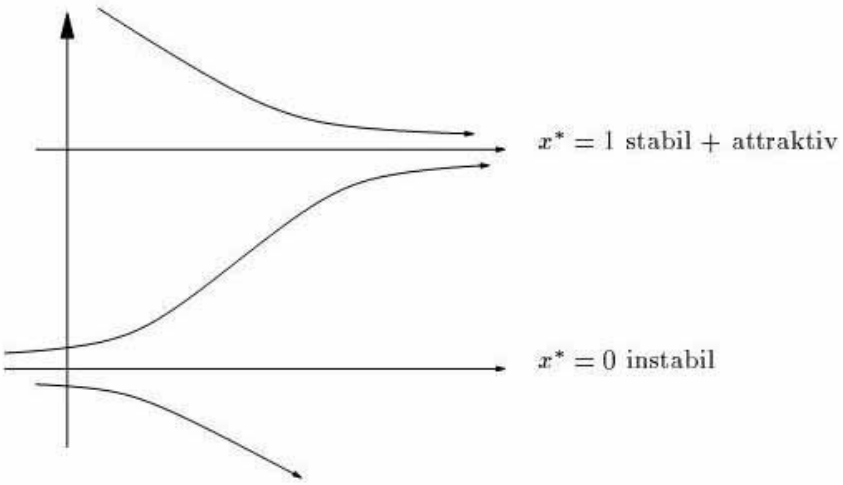
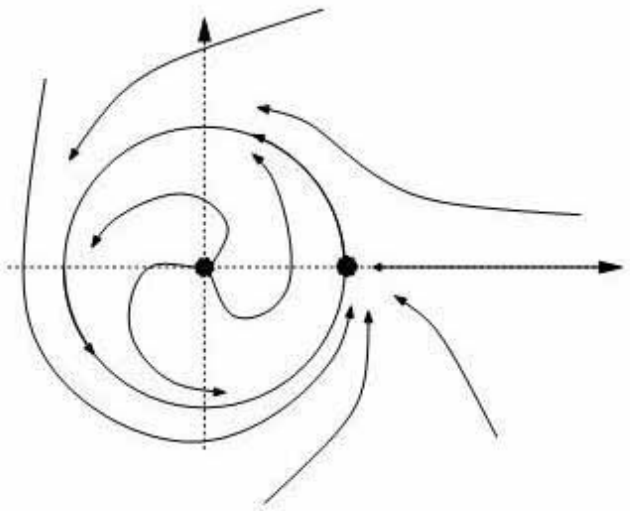


Abbildung 3.2

Definition (3.14):
Eine Ruhelage x^* heißt asymptotisch stabil, wenn sie stabil und attraktiv ist.

Beispiel (3.15):
Betrachte die folgende 2-dimensionale DGL in Polarkoordinaten $\frac{dr}{dt} = r - r^2$, $\frac{d\theta}{dt} = \sin^2(\frac{1}{2}\theta)$.
Die Ruhelagen sind $(x^*, y^*) = (0, 0)$ und $(x^*, y^*) = (1, 0)$. Der Einheitskreis $r = 1$ ist auch eine invariante Kurve.



Die Ruhelage $(x^*, y^*) = (1, 0)$ ist attraktiv aber nicht stabil.

Abbildung 3.3

Satz 3.16:
(Ljapunov, Spektrum und Stabilität)
Gegeben sei eine autonome Differentialgleichung mit einer Ruhelage x_0 . Die Abbildung f sei in einer Umgebung von x_0 stetig differenzierbar und $A = Df(x_0)$ sei die Linearisierung (Ableitung) von f im Punkt x_0. Dann gilt:

1. Haben alle Eigenwerte von A negative Realteile, so ist x_0 asymptotisch stabil.

2. Gibt es einen Eigenwert von A dessen Realteil positiv ist, so ist x_0 instabil.

Beweis: Wir beweisen zunächst die Stabilitätsaussage. O.B.d.A dürfen wir davon ausgehen, dass $x_0 = 0$ ist. Zur Erleichterung der Schreibweise wollen wir dies auch tun. Da $A = Df(0)$ ist, kann man die Abbildung f in der Form $f(x) = Ax + g(x)$ schreiben, wobei g bei 0 stetig differenzierbar ist und $Dg(0) = 0$ ist. Also hat die zu betrachtende Gleichung die Form

$$\dot{x} = Ax + g(x) \tag{3.9}$$

Wir müssen die Stabilität und die Attraktivität von $x_0 = 0$ zeigen. Die Stabilität kann man auch so formulieren: Zu jedem $\varepsilon > 0$ existiert ein $\delta > 0$, so dass

$$|\upsilon_0| < \delta \Longrightarrow |x(t, \upsilon_0)| < \varepsilon \qquad \forall t \geq 0.$$

Sei also $\varepsilon > 0$ gegeben. Sei $\upsilon_0 \in U$. Die Lösung $x(t) = x(t, \upsilon_0)$ von (3.9) genügt der Intergralgleichung

$$x(t) = E(A, t)\upsilon_0 + \int_0^t E(A, t-s)g(x(s))ds,$$

da natürlich nach Einsetzen der Lösung $x(t, \upsilon_0)$ in die Gleichung (3.9) daraus mit $h(t) = g(x(t, \upsilon_0))$ die äquivalente Gleichung

$$\dot{x} = Ax + h(t)$$

entsteht. Sei $\mu = \max\{Re(\lambda)|\lambda \in \sigma(A)\}$. Dann ist $\mu < 0$ und die Norm von $E(A, t)$ kann für jedes $\alpha \in (\mu, 0)$ abgeschätzt werden durch

$$\|E(A, t)\| \leq M_\alpha e^{\alpha t},$$

wobei M_α die reelle Konstante ist, und deren Existenz gewährleistet. Wir finden auch eine Positive Konstante $L > 0$ mit

$$M_\alpha L < \frac{1}{2}\min\{-\alpha, 1\}.$$

Es gibt ein $\eta > 0$, $\eta < \varepsilon$, so dass auf $B_\eta(0)$ die Funktion g die Lipschitz Konstante L hat. Sei $\delta = \frac{1}{2}\frac{\eta}{M_\alpha}$. Nun hat man für $|\upsilon_0| < \delta$ die Abschätzung

$$|x(t, \upsilon_0)| \leq \|E(A, t)\||\upsilon_0| + \int_0^t \|E(A, t-s)\||f(x(s, \upsilon_0))|ds. \tag{3.10}$$

Man kann es einsetzen, solange $|x(s, \upsilon_0)| < \eta$ ist

$$\begin{aligned} |x(t, x_0)| &\leq M_\alpha e^{\alpha t}\delta + M_\alpha L\eta \int_0^t e^{\alpha(t-s)}ds \\ &< \frac{1}{2}\eta + M_\alpha L\eta \frac{1}{|\alpha|}(1 - e^{\alpha t}) \\ &< \frac{1}{2}\eta + M_\alpha L\eta \frac{1}{|\alpha|} \end{aligned}$$

$$< \eta < \varepsilon.$$

Damit hat man die Stabilität, denn angenommen es gäbe eine Zahl $t^* > 0$ mit $|x(t^*, \upsilon_0)| = \eta$, dann kann man o.B.d.A. davon ausgehen, dass t^* minimal gewählt wurde. Dann ist für $t \in (0, t^*)|x(t, \upsilon_0)| < \eta$. Unsere Abschätzung garantiert, dass auch noch für $t = t^*$ die Lösung $x(t, \upsilon_0)$ in $B_\eta(0)$ enthalten ist. Damit ergibt sich ein Widerspruch.

Um die asymptotische Stabilität zu zeigen, nutzen wir die Formel (3.10) und schätzen etwas anders ab:

$$|x(t, \upsilon_0)| \leq M_\alpha \alpha e^{\alpha t}\delta + M_\alpha L \int_0^t e^{\alpha(t-s)}|x(s, \upsilon_0)|ds. \tag{3.11}$$

Multiplikation mit $e^{-\alpha t}$ liefert für die Funktion $\upsilon(t) = e^{-\alpha t}|x(t, \upsilon_0)|$ die Abschätzung

$$\upsilon(t) \leq M_\alpha \delta + M_\alpha L \int_0^t \upsilon(s)ds.$$

Hier verwenden wir das Lemma von Gronwall und erhalten

$$\upsilon(t) \leq M_\alpha \delta e^{M_\alpha L t}.$$

Damit ergibt sich für $x(t, x_0)$ die Abschätzung

$$|x(t, x_0)| \leq M_\alpha \delta e^{(M_\alpha L + \alpha)t}.$$

Da $M_\alpha L + \alpha < 0$ ist, konvergiert dies gegen Null. Damit ist die asymptotisch Stabilität gezeigt.

Wir nehmen nun an, dass die Linearisierung einen Eigenwert mit positivem Realteil besitzt. Wir zeigen, dass es eine nichttriviale Lösung $x(t)$ gibt mit $x(t) \longrightarrow \omega_0$ für $t \longrightarrow -\infty$. Daraus folgt dann die Behauptung. Zunächst wollen wir dies als eigenes Lemma formulieren. Sei $x(t)$ eine Lösung mit $x(t) \longrightarrow \omega_0 = 0$ für $t \longrightarrow -\infty$. Sei $\varepsilon = |x(0)| > 0$. (Es ist keine Beschränkung der Allgemeinheit anzunehmen, dass 0 im maximalen Existenzintervall enthalten ist.) Angenommen, es existiere ein $\delta > 0$, so dass aus $|x_0| < \delta$ folgen würde, dass für alle $t \geq 0$ gilt $|x(t, x_0)| < \varepsilon$, so könnte

man schließen, dass $|x(t)| > \delta, \forall t \leq 0$. Dies ist ein Widerspruch. Wie angekündigt wollen wir die Existenz von $x(t)$ getrennt zeigen. $\square$

3.2.4 Stabilitätskriterium für Ruhelagen x_0

Sei $A = f_x(x_0, \lambda), x_0$ Ruhelage, λ fest und $\sigma(A) = \{\mu_1, \mu_2, \cdots, \mu_n\}$ die charakteriste Exponenten:

(i) Gilt $\mathrm{Re}\mu_i < 0, i = 1, \cdots, n$, so ist x_0 asymptotisch stabil.

(ii) Existiert ein $i \in \{1, \cdots, n\}$ mit $\mathrm{Re}\mu_i > 0$, so ist x_0 Ljapunov-instabil.

Folgerung 3.17:
(i) Ist die Spektralabszisse der Matrix A

$$\upsilon(A) := \max_{\mu_i \in \sigma(A)} \mathrm{Re}\mu_i \tag{3.12}$$

negativ, so ist x_0 asymptotisch stabil.

(ii) Jede Senke x_0 ist asymptotisch stabil; jeder Sattel und jede Quelle ist Ljapunov-instabil.

(iii) Über nicht-hyperbolische Ruhelagen kann nicht entschieden werden. (Im linearen Fallen $\dot{x} = Ax$ kann die Ljapunov-Stabilität garantiert werden, falls alle μ_i mit $\mathrm{Re}\mu_i = 0$ einfach sind.)

Beispiel 3.18
Harmonische Gleichung: $\dot{x_1} = x_2, \dot{x_2} = -x_1$
$f(x, \lambda) = \begin{pmatrix} x_2 \\ x_1 \end{pmatrix} \Longrightarrow f_x(x, \lambda) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$
$\Longrightarrow \mu_{1,2} = \pm i \Longrightarrow x_0 = 0$ ist nicht hyperbolisch.
$\Longrightarrow$ das Kriterium ist nicht anwendbar (aber wegen der Linearität und Einfachheit von μ_1, μ_2).

3.2.5 Stabilitätswechsel bei parameterabhängigen System

Folgerungen 3.19:
(i) Ist x_0 keine reguläre Lösung von $f(x, \lambda_0) = 0$, so $f_x(x_0, \lambda_0)$ singulär. Dann muss für minderstens einen Eigenwert $\mu_1(\lambda_0)$ der Jacobi-Matrix $A := f_x(x_0, \lambda_0) : \mu_1(\lambda_0) = 0$ gelten. Im Allgemeinen tritt bei $\lambda = \lambda_0$ ein Stabilitätswechsel (stabil $\Longleftrightarrow$ instabil) ein.

(ii) Stabilitätsverluste (stabil $\longrightarrow$ instabil) werden durch $Re\mu_i(\lambda^*) = 0$ für ein $i \in \{1, \cdots, n\}$ singularisiert Bedingung ist notwendig für den Stabilitätsverlust bei $\lambda = \lambda^*$, jedoch nicht hinreichend.

Beispiel 3.20:
$\dot{x} = x(x^3 - x - \lambda), \lambda \in [-1, 1]$
Betrachte wir den Lösungszweig:

$$\mathcal{L}_0 = \{(x_0, \lambda) | x_0 = 0, \lambda \in [-1, 1]\}$$

Weil $f_x(x, \lambda) = 4x^3 - 2x - \lambda^2$ ist, dann $f_x(x_0, \lambda) = -\lambda^2, \quad \forall \lambda \in \Lambda$. Der Eigenwert lautet:

$$\mu_1(\lambda) = -\lambda^2 \leq 0, \forall \lambda \in \Lambda$$

$x_0 = 0$ ist asymptotisch stabil für alle $\lambda \neq 0$.
Bei $\lambda = \lambda^* = 0$ ist zwar $\mathrm{Re}\mu_1(\lambda^*) = 0$, aber es findet kein Stabilitätswechsel statt. Und hier ist
$\mu_1'(\lambda) = -2\lambda \longrightarrow \mu_1'(\lambda^*) = 0$

3.2.6 Bedeutung des Stabilitätsverhaltens

Da in praktischen Anwendungen asymptotisch stabile Lösungen x_0 der Gleichung $x_0 = f(x, \lambda), x(0) = x_0$ sinnvoll sind (wegen unvermeidlicher Störungen in $x(0)$), ist die Klärung des Stabilitätsverhaltens sehr bedeutsam.

3.3 Bifurkationsanalyse

3.3.1 Definition der Bifurkation

Eine Bifurkation oder Verzweigung ist eine qualitative Zustandsänderung in nichtlinearen Systemen unter Einfluss eines Parameters λ.
Nichtlineare Systeme, deren Verhalten von einem Parameter abhängt, können bei einer Änderung des Parameters ihr Verhalten plötzlich ändern. Zum Beispiel kann ein System, das zuvor einem Grenzwert zustrebte, nun zwischen zwei Werten hin und her springen, also zwei Häufungspunkte aufweisen. Dies nennt man eine Bifurkation. Bestimmte Systeme können unter finiter Änderung des Parameter λ unendlich viele Bifurkationen erfahren, und damit eine unendliche Menge an Grenzwerten aufweisen. Das Verhalten solcher Systeme wandelt sich unter Änderung des Parameters λ somit zu deterministisch chaotischem Verhalten. Bifurkationen lassen sich in Bifurkationsdiagrammen graphisch darstellen.

Definition 3.21:
Eine Bifurkation eines dynamischen Systems ist eine Änderung der Anzahl an (stabilen und instabilen) Grenzwerten (Gleichgewichtszuständen) unter dem Einfluss eines Parameters λ. Man unterscheidet lokale Bifurkationen, die nahe einzelner Orbits des dynamischen Systems ablaufen, und globale Bifurkation, die sofort einen großen Teil des Phasenraumes betreffen.

Beispiel 3.22:
Ein typisches Beispiel einer Bifurkation ist das Knicken eines Stabes unter Druckspannung.
Man stelle sich einen im Boden eingespannten senkrecht stehenden masselosen Stab mit einem Gewicht λ an der Spitze vor. Die Winkelabweichung des Stabes aus der Senkrechten entspricht dabei der Variablen x.
Ist das Gewicht genügend klein, so ist $x = 0$ eine stabile Gleichgewichtslage des Systems, d.h. für kleine Abweichungen richtet sich der Stab selbständig wieder in die Senkrechte ($x = 0$) aus. Wird das Gewicht λ kontinuierlich gesteigert, so wird bei einem bestimmten Gewicht (der Knickspannung) die senkrechte Gleichgewichtslage instabil. Gleichzeitig entstehen (für ein ebenes System) zwei neue (stabile) Gleichgewichtslagen. (Indem der Stab nach links oder rechts abknickt.) Der Übergang des Systems von einer (stabilen) zu drei (zwei stabile eine instabile) Gleichgewichtslagen ist die Bifurkation, die in diesem Fall eine Heugabel-Bifurkation ist.

3.3.2 Typen von Bifurkation

(a) Sattel-Knoten-Bifurkation
Die Sattel-Knoten-Bifurkation ist ein bestimmter Typ einer Bifurkation eines nichtlinearen Systems. Die Sattel-Knoten-Bifurkation wird auch Turning-point-Bifurkation genannt.
Die Normalform der Sattel-Knoten-Bifurkation ist:

$$\frac{dx}{dt} = \lambda - x^2 \tag{3.13}$$

wobei λ der Parameter ist.
Die Sattel-Knoten-Bifurkation hat folgende Gleichgewichtspunkte (man beachte, dass diese erst für $\lambda > 0$ auftreten):

$$\begin{aligned} x_1^* &= \sqrt{\lambda} \\ x_2^* &= -\sqrt{\lambda} \end{aligned} \tag{3.14}$$

Die Anzahl an Gleichgewichtspunkten verhält sich also für eine Änderung des Parameters λ folgendermaßen:

$$\begin{aligned} \lambda < 0 &: \text{GGW} = \{\} \\ \lambda = 0 &: \text{GGW} = \{0\} \\ \lambda > 0 &: \text{GGW} = \{x_1^*, x_2^*\} \end{aligned} \tag{3.15}$$

Die Anzahl an Gleichgewichtspunkten wechselt also von 0 zu 1 zu 2.

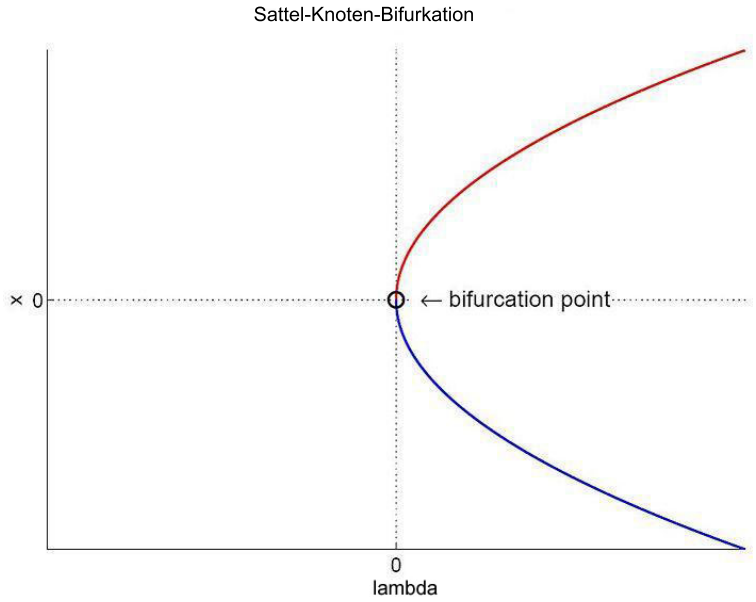


Abbildung 3.4: Bifurkationsdiagramm einer Sattel-Knode-Bifurkation

(b) Heugabel-Bifurkation
Die Normalform der Heugabel-Bifurkation ist:

$$\frac{dx}{dt} = \lambda \cdot x + \alpha \cdot x^3, \qquad \text{mit} \quad \alpha = \pm 1 \text{und} \lambda \text{der Parameter ist.} \tag{3.16}$$

Für $= 1$ erhält man die subkritische Heugabel-Bifurkation und für $= -1$ erhält man die superkritische Heugabel-Bifurkation.

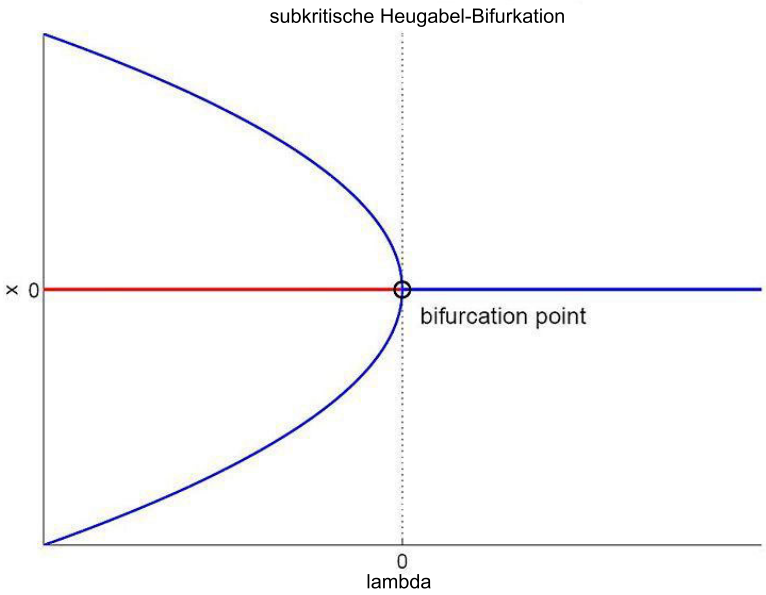


Abbildung 3.5: Bifurkationsdiagramm einer subkritischen Heugabel-Bifurkation

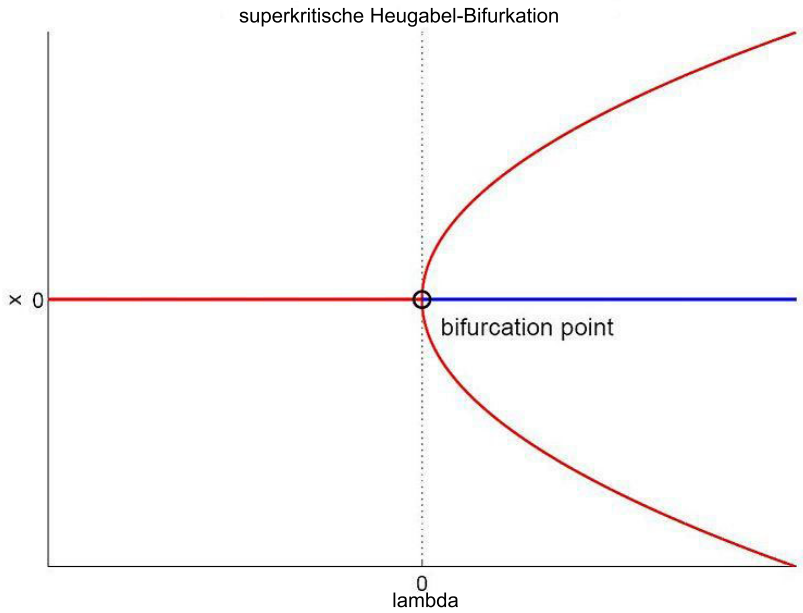


Abbildung 3.6: Bifurkationsdiagramm einer superkritischen Heugabel-Bifurkation

Die Heugabel-Bifurkation hat folgende Gleichgewichtspunkte:

$$x_1^* = 0$$
$$x_2^* = \pm\sqrt{-\frac{\lambda}{\alpha}} \tag{3.17}$$

(c) Transkritische Bifurkation
Die Normalform der Transkritischen Bifurkation ist:

$$\frac{dx}{dt} = \lambda \cdot x - x^2, \qquad \text{mit} \quad \lambda\text{der Parameter ist.} \tag{3.18}$$

Die Transkritische Bifurkation hat folgende Gleichgewichtspunkte:

$$x_1^* = 0$$
$$x_2^* = \lambda \tag{3.19}$$

Die Anzahl an Gleichgewichtspunkten verhält sich also für eine Änderung des Parameter λ folgendermaßen:

$$\begin{aligned} \lambda < 0 : \text{GGW} &= \{\lambda, 0\} \\ \lambda = 0 : \text{GGW} &= \{0\} \\ \lambda > 0 : \text{GGW} &= \{0, \lambda\} \end{aligned} \tag{3.20}$$

Die Anzahl an Gleichgewichtspunkten wechselt also von 2 zu 1 zu 2.

Diskretes System:
Für ein diskretes System wird:

$$x_{t+1} = x_t + \lambda \cdot x - x_t^2 \tag{3.21}$$

Die Lage der Fixpunkte bleibt gegenüber dem kontinuierlichen System unverändert.

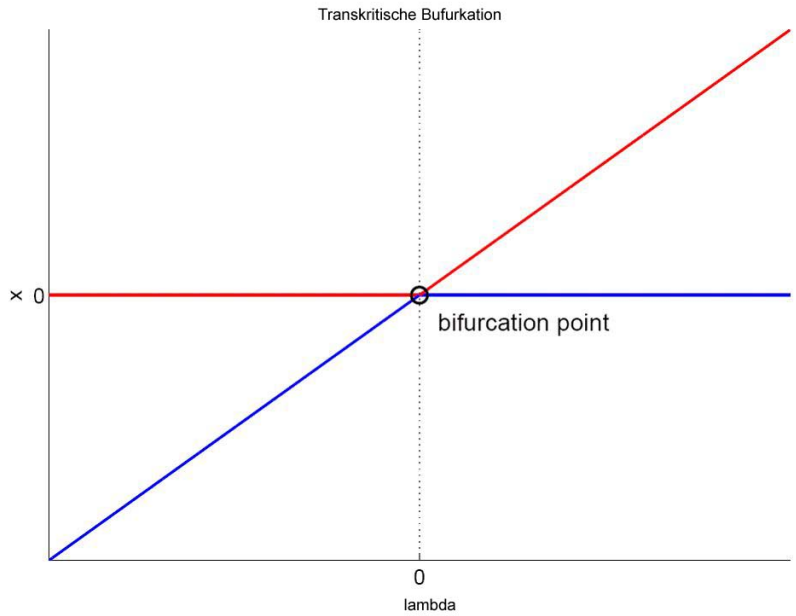


Abbildung 3.7: Bifurkationsdiagramm einer Transkritische Bifurkation

(d) Hopf-Bifurkation
Eine Hopf-Bifurkation oder Hopf-Andronov-Bifurkation ist ein Typ einer lokalen Bifurkation in nichtlinearen Systemen.
Hopf-Bifurkationen zeichnen sich dadurch aus, dass bei der Variation eines Parameters ein Gleichgewicht seine Stabilität verliert und in einen Grenzzyklus übergeht. Je nachdem, ob dieser Grenzzyklus stabil oder instabil ist, spricht man von einer superkritischen oder subkritischen Hopf-Bifurkation.
Bei einer Hopf-Bifurkation überquert an einem Gleichgewichtspunkt (Fixpunkt) des

Systems ein Paar komplex konjugierter Eigenwerte der aus der Linearisierung des Systems resultierenden Jacobimatrix die imaginäre Achse der komplexen Ebene, am Bifurkationspunkt sind die konjugierten Eigenwerte also rein imaginär.
Die Kodimension der Hopf-Bifurkation ist ebenso wie bei der Sattel-Knoten-Bifurkation, der Heugabel-Bifurkation und der Transkritischen Bifurkation gleich eins, die anderen Typen von Bifurkationen der Kodimension 1 zeichnen sich hingegen durch einen Eigenwert $\mu = 0$ der Jacobimatrix am Fixpunkt aus.

4 Numerische Anwendungen

4.1 Beispiel 1: Eindimensionale Gleichung mit Parameter λ

Betrachten wir das eindimensionales Beispiel $f(x, \lambda) = 0$ mit
$f(x, \lambda) := 40\mu x^5 - 200\mu x^4 + (342\mu + 8)x^3 - (226\mu + 24)x^2 + 0.5(91\mu + 37)x - \lambda$,
wobei $\lambda \geq 0$ der Systemparameter ist.

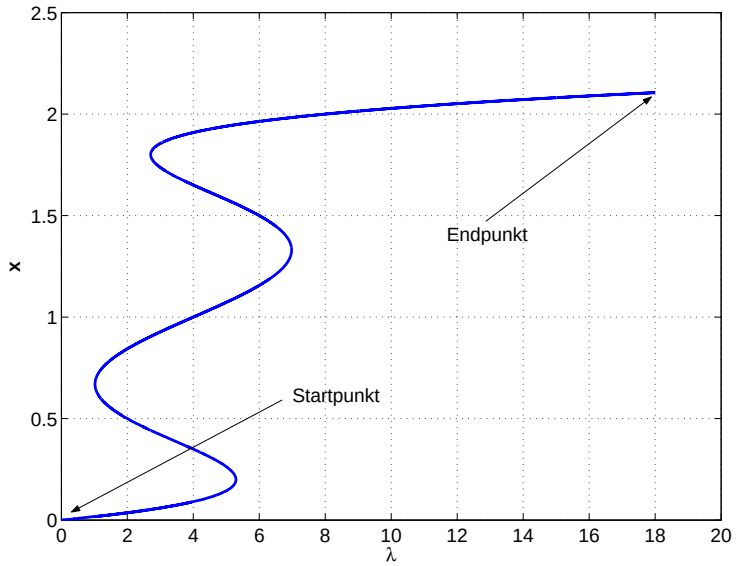


Abbildung 4.1: Exakte Lösungskurve mit $\mu = 1$ (Bsp.1)

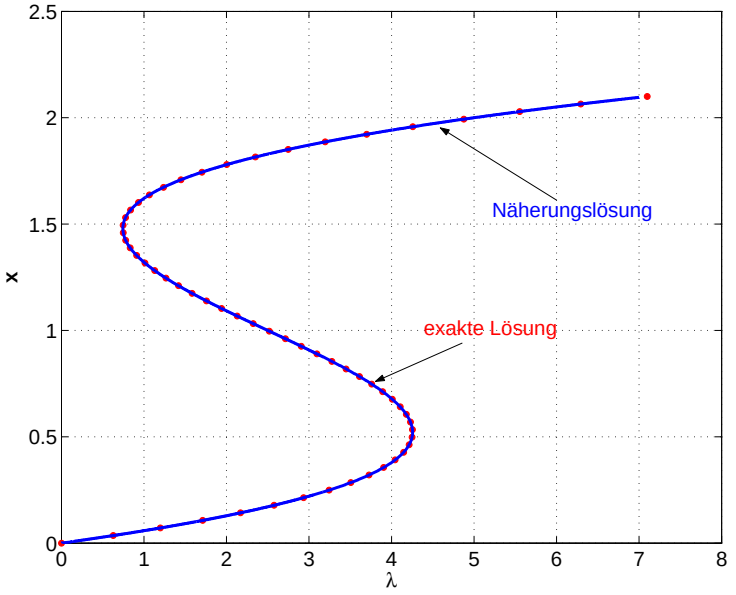


Abbildung 4.2: Exakte Lösungskurve mit $\mu = 0$ (Bsp.1)

In Abb.(4.1) und (4.2) können wir die Lösung für verschiedne Parameter μ fin-

den, z.B. bei $\lambda = 1$, für festen Parameter $\mu = 0$ treten drei Lösungen und für $\mu = 1$ fünf Lösungen auf. Jede Lösung $y = (x, \lambda)$ von $f(y) = 0$ ist regulär, weil die Jacobi-Matrix $D_\lambda f(x, \lambda) = -1$ ist. Für festen Parameter $\mu = 1$ erhalten wir mit Startwerten $(x_0, \lambda_0) = (0, 0)$ und maximaler Fortsetzungsschrittwerte $h_{max} = 0,005$ in Abb.(4.1) die Lösung mit Genauigkeit $tol = 10^{-3}$ über alle 4 Umkehrpunkt hinweg. Bei $\mu = 0$ arbeitet die Fortsetzung zuverlässig und liefert in Abb. (4.2) die Näherungslösung.

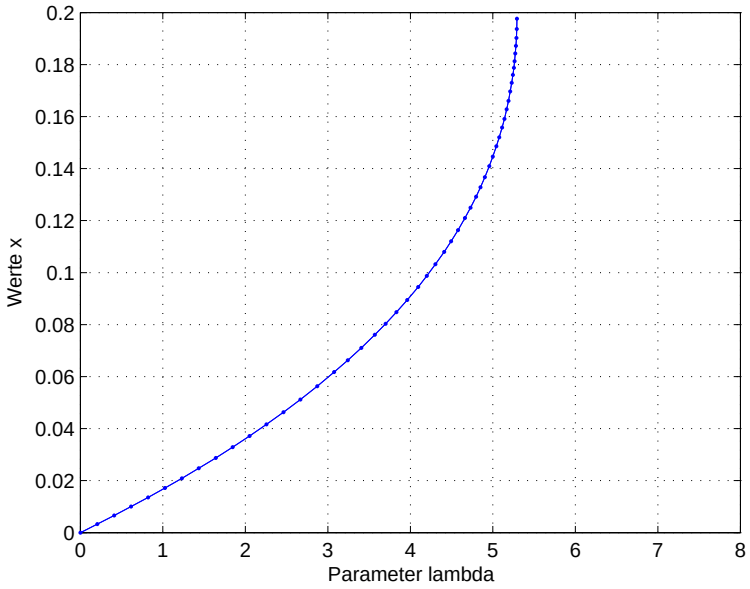


Abbildung 4.3: Berechnete Lösung mit $\mu = 1$ (Bsp.1)

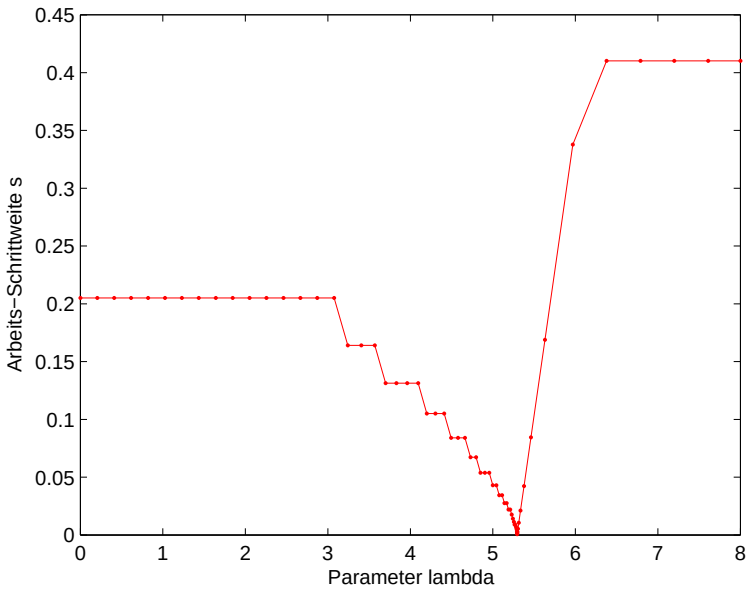


Abbildung 4.4: Arbeits-S chrittweite h (Bsp.1)

Die natürliche Parameterisierung nach Algorithmus 2.6 liefert mit Startwerten $(x_0, \lambda_0) = (0, 0)$ die Lösung zuverlässig bis zum ersten Umkehrpunkt (Abb. 4.3). Dann bricht

das Verfahren mit extremer Schrittweitenreduktion zusammen. (dargesstellt in Abb. 4.4)

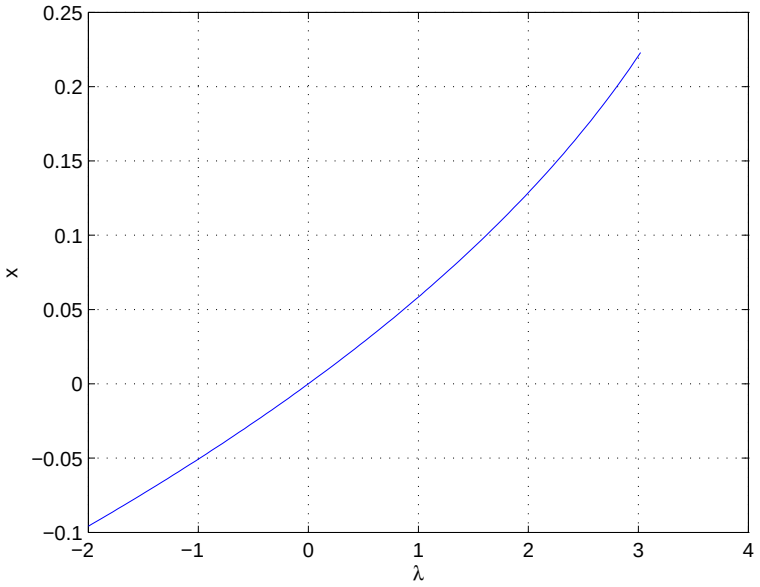


Abbildung 4.5: Stabilitätverhalten: Instabilität (Bsp.1)

4.2 Beispiel 2: Gleichung mit Parameter λ

Betrachten wir das folgendes Beispiel in Abb.(4.6) mit:
$f(x, \lambda) = x^3 - x - \lambda, \quad \text{mit} \quad \lambda \geq 0.$

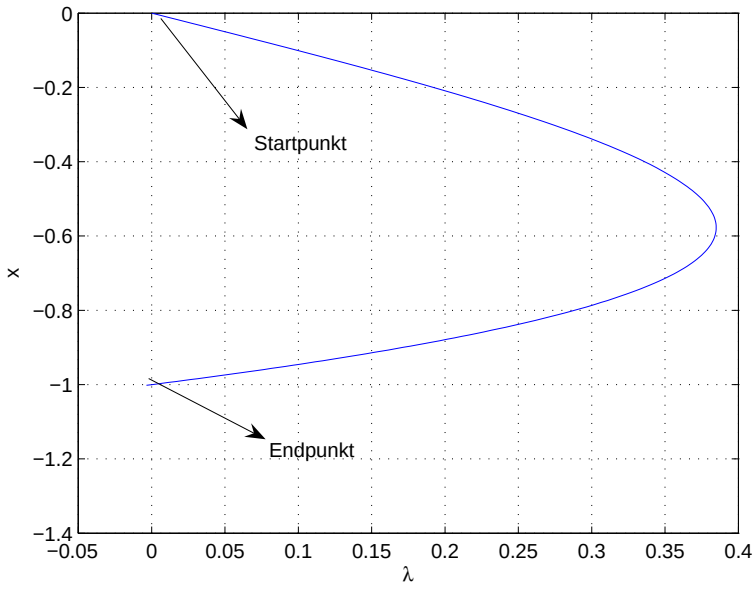


Abbildung 4.6: Exakte Lösungskurve (Bsp.2)

Wir treten für $\lambda = 0$ drei Lösungen auf, weil das Jacobi-Matrix $D_\lambda f(x, \lambda) = -1$ ist, dann ist jede Lösung $y = (x, \lambda)$ von $f(y) = 0$ regulär. Und die natürliche Parametriserung liefert die Lösung zulässig bis zum ersten Umkehrpunkt (in Abb.4.7). Dann bricht das Verfahren mit extremer Schrittweitenreduktion zusammen ,dargestellt wie in Abb.(4.8).

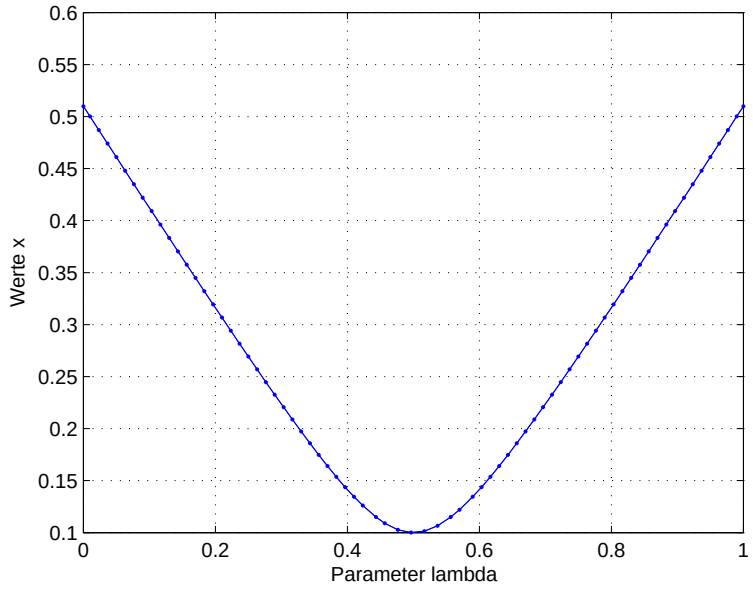


Abbildung 4.7: Berechnete Lösung (Bsp.2)

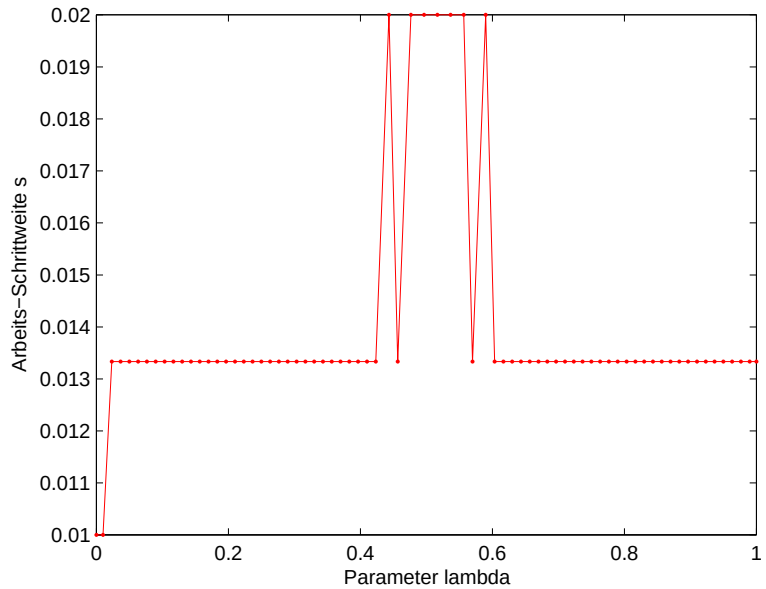


Abbildung 4.8: Arbeits-Schrittweite h (Bsp.2)

Erhalten wir mit Startwerten $(x_0, \lambda_0) = (0, 0)$ und maximaler Fortsetzungsschrittweite $h_{max} = 0,05$ die Lösung mit Genauigkeit $tol = 10^{-4}$ über die Umkehrpunkt.

Zum Schluß ist die Stabilität des Beispiels 2 instabil.

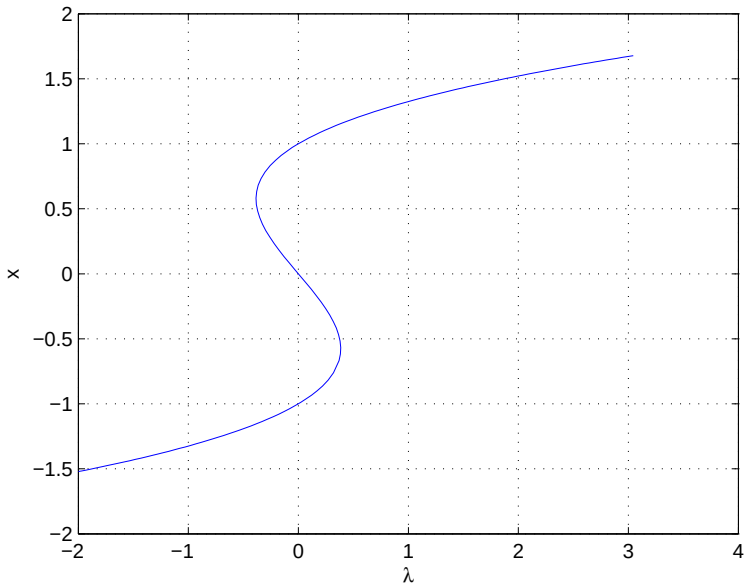


Abbildung 4.9: Stabilitätsverhalten: Instabilität (Bsp.2)

4.3 Beispiel 3: Das Stabilitätsproblem vom Flugzeug

Dieses Problem kann in folgender Form geschrieben werden:

$$-3.933x_1 + 0.107x_2 + 0.126x_3 - 9.99x_5 - 45.83\lambda - 0.727x_2x_3 + 8.39x_3x_4 - 684.4x_4x_5 + 63.5x_4\lambda = 0$$
$$-0.987x_2 - 22.95x_4 - 28.37u + 0.949x_1x_3 + 0.173x_1x_5 = 0$$
$$0.002x_1 - 0.235x_3 + 5.67x_5 - 0.921\lambda - 0.713x_1x_2 - 1.578x_1x_4 + 1.132x_4\lambda = 0$$
$$x_2 - x_4 - 0.168u - x_1x_5 = 0$$
$$-x_3 - 0.196x_5 - 0.0071\lambda + x_1x_4 = 0$$

Hier sind x_1, x_2, x_3die Rolle-Rate, Pitch-Rate und Gierrate. x_4 ist die inkrementelle Anstellwinkel. Und x_5 ist der Gleit-Winkel. Die Variable u ist die Kontrolle für den Lift und λ ist die für das Querruder. Die Verformung des Ruders ist auf Null gesetzt. Für $u = 0$ ist das Problem symmetrisch: $F(x, \lambda) = F(x, -\lambda)$. Für $u = -0.008$ ist die gestörte Symmetrie noch sichtbar.

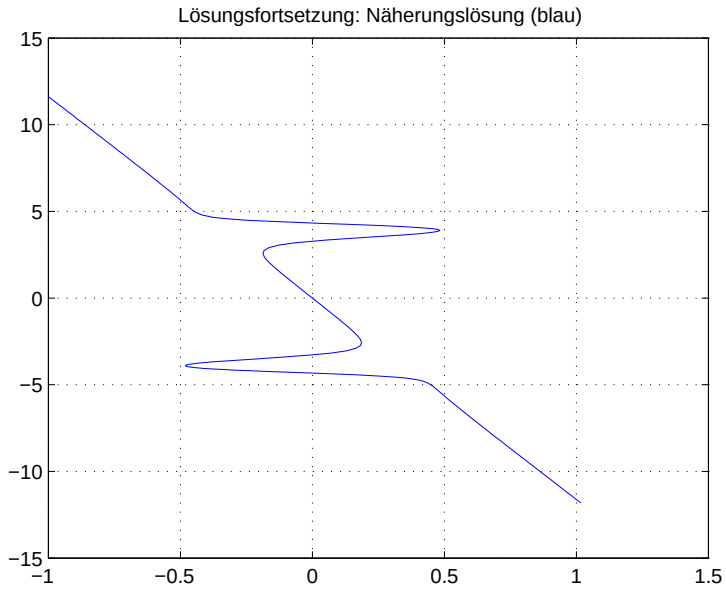


Abbildung 4.10: Nährugslösung : x_1 (Bsp.3)

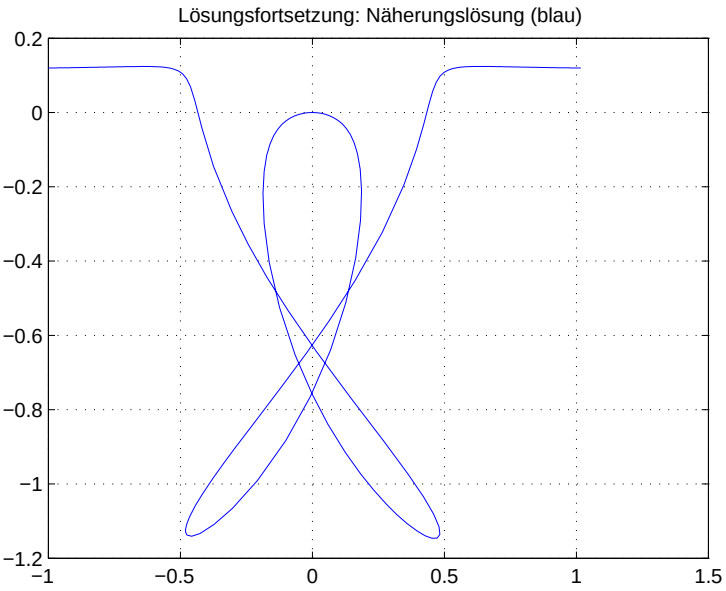


Abbildung 4.11: Nährugslösung : x_2 (Bsp.3)

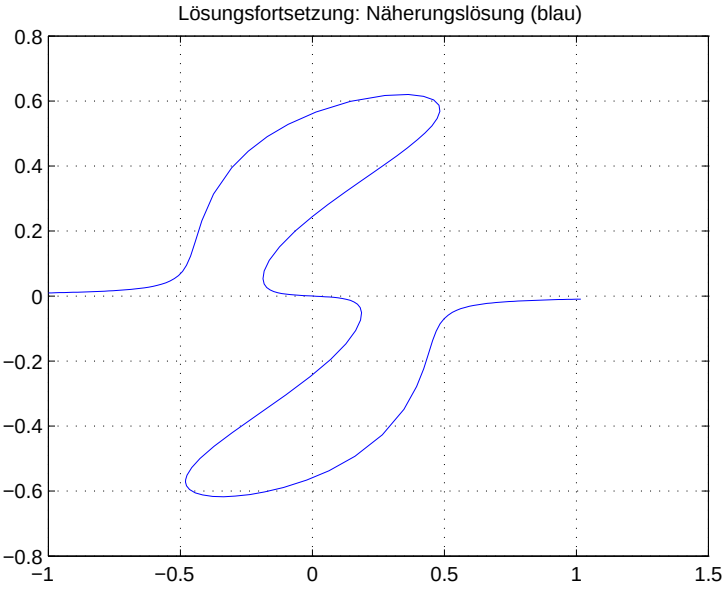


Abbildung 4.12: Nährugslösung : x_3 (Bsp.3)

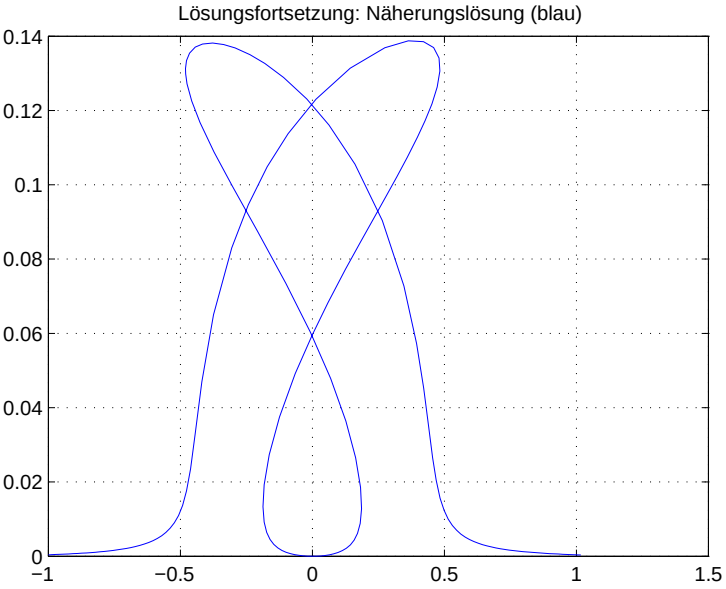


Abbildung 4.13: Nährugslösung : x_4 (Bsp.3)

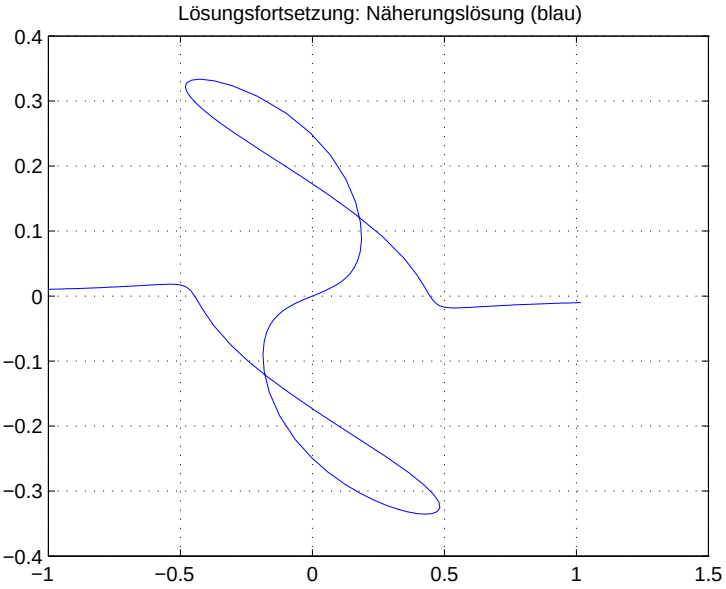


Abbildung 4.14: Nährugslösung : x_5 (Bsp.3)

In den Abb.4.10 - Abb.4.14 können wir der Charakter über $x_i, \ldots, x_5$ mit $\lambda \in [-1, 1.5]$ anschauen. Alle Variablen sind symmetrisch. Bei x_4 können wir einfach sehen, daß dieser Charakter auch im Stabilitätswechsel benutzt wird. (Abb.4.15)

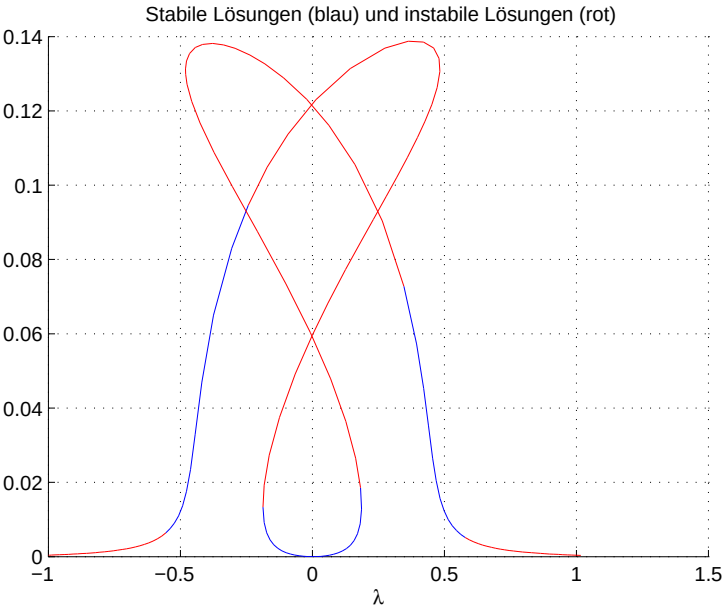


Abbildung 4.15: Stabilitätsverhalten: x_4 (Bsp.3)

4.4 Beispiel 4: Zweiboxen-Brüsselator

Der Brüsselator ist ein chemisches reagierendes System, das bei Änderung von Parametern durch eine Hopf-Bifurkation selbsterregte Schwingungen erzeugen kann. Wir wollen jetzt zwei homogene Teilsysteme ("Boxen“), in denen die Reaktionen mit den gleichen Parametern ablaufen, durch lineare Diffusion der beiden Stoffe X und Y miteinander koppeln.
Das mathematische Modell besteht dann in folgendem Gleichungssystem:

$M = R^4_+ : \dot{x_1} = f(x_1, y_1) + d(x_2 - x_1), f(x, y) = A - (B + 1)x + x^2 y,$
$\dot{y_1} = g(x_1, y_1) + D(y_2 - y_1), g(x, y) = B - x^2 y,$
$\dot{x_2} = f(x_2, y_2) - d(x_2 - x_1),$
$\dot{y_2} = g(x_2, y_2) - D(y_2 - y_1), d \geq 0, D \geq 0.$

in dem x_1 und y_1 bzw. x_2 und y_2 die Konzentrationen der Stoffe X und Y in den Boxen 1 bzw. 2 bezeichenen und d bzw. D die Diffusionskonstanten beider Stoffe sind. Die Fixpunkte $(x_1^0, y_1^0, x_2^0, y_2^0)$ erfüllen dann dia algebraischen Gleichungen

$$A - (B + 1)x_1 + x_1^2 y_1 + d(x_2 - x_1) = 0$$

$$Bx_1 - x_1^2 y_1 + D(y_2 - y_1) = 0$$

$$A - (B + 1)x_2 + x_2^2 y_2 + d(x_2 - x_1) = 0$$

$$Bx_2 - x_2^2 y_2 + D(y_2 - y_1) = 0 \tag{4.1}$$

Wir stellen zunächst fest, daß der Fixpinkt $x = A, y = B$ des homognen Brüsselators auch eine homogene stationäre Lösung des Zweiboxenmodells (4.1) liefert:

$$x_1 = x_2 = A, y_1 = y_2 = B/A \tag{4.2}$$

Weiterhin erkennen wir an der Symmetrie von (4.1), daß mit (x_1, x_2, y_1, y_2) auch das Spiegelbild (x_1, x_2, y_1, y_2) ein Fixpinkt ist. Diese inhomogenen Lösungen $x_1 \neq x_2, y_1 \neq y_2$ können wir explizit berechnen. Aus der Summe aller vier Gleichung in (4.1) folgt

$$x_1 + x_2 = 2A, \tag{4.3}$$

was wir nach x_2 auflösen. Anschließend lösen wir die Summe der beiden ersten Gleichungen nach y_2 und danach die erste Gleichung nach y_1 auf und erhalten

$$x_2 = 2A - x_1, y_1 = [(B + 2d + 1)x_1 - (2d + 1)A]/x_1^2$$

$$y_2 = y_1 + (x_1 - A)(2d + 1)/D$$

Wegen (4.3) substituieren wir $x_1 = A + u$, das $x_2 = A + u$ zur Folge hat, so daß mit u auch $-u$ Lösung der entstehenden Gleichung ist. Nach Abspalten eines Faktor u entsteht deshalb die biquadratische Gleichung

$u^4 + 2[(b + 1)D - A^2]u^2 + A^2[2(1 - b)D + A^2] = 0$
$b := \frac{B}{2d+1}$

die explizit gelöst werden kann.
Wir verzichten auf eine vollständige Diskussion und stellen hier nur fest, daß es bei geeigneten Parameterwerten zwei bzw. vier inhomogene stationäre Zustände gibt, in denen beide Boxen unterschiedliche Konzentrationswerte der Stoffe X bzw. Y aufweisen.

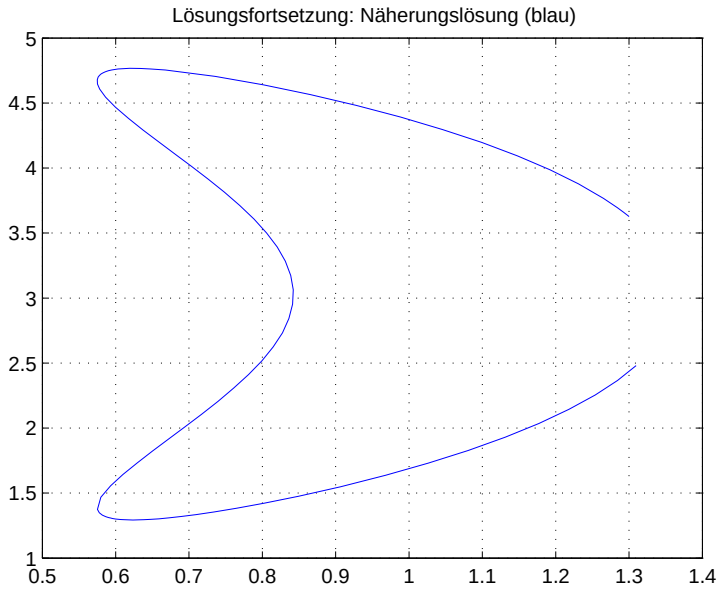


Abbildung 4.16: Nährungslösung : x_1 (Bsp.4)

Die Stabilität der inhomogenen Zustände ermitteln wir nur durch Bifurkationsanalyse und betrachten dazu in Abb.4.13 die Abhängigkeit der x_1-Werte aller Fixpunkte vom Kopplungsparameter d für den Fall $D = 2d$. Wir erkennen, daß bei $d = 0.6$ die beiden inhomogenen Zweige durch je eine Tangentenbifurktion entstehen.

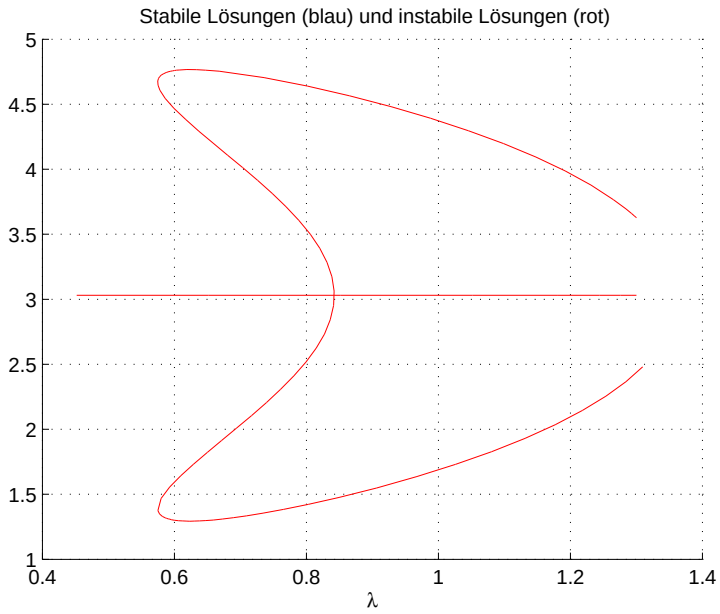


Abbildung 4.17: Stabilitätsverhalten : x_1 (Bsp.4)

4.5 Beispiel 5:Chemisches Reaktionsproblem

Dieses Modell aus [∗] hat die Gleichung:

$$\lambda(1 - x_3)\exp(10x_1/(1 + 0.01x_1)) - x_3 = 0$$
$$22\lambda(1 - x_3)\exp(10x_1/(1 + 0.01x_1)) - 30x_1 = 0$$
$$x_3 - x_4 + \lambda(1 - x_4)\exp(10x_2/(1 + 0.01x_2)) = 0$$
$$10x_1 - 30x_2 + 22\lambda(1 - x_4)\exp(10x_2/(1 + 0.01x_2)) = 0$$

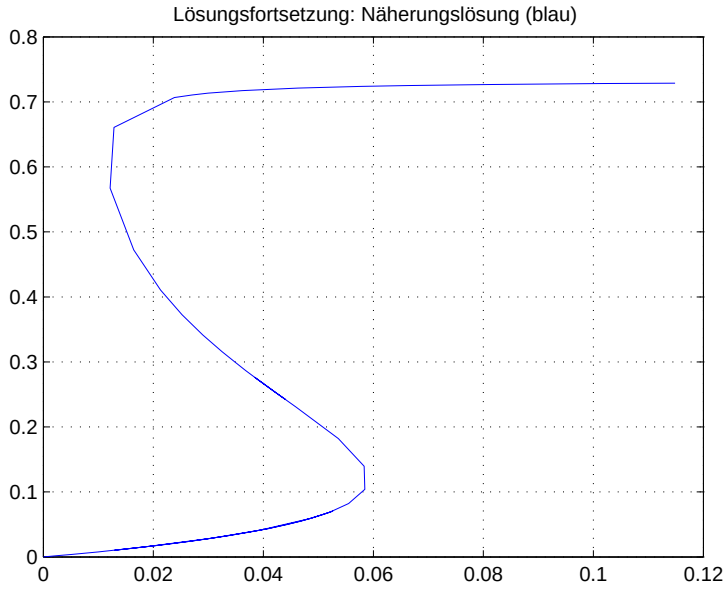


Abbildung 4.19: Nährungslösung : x_1 (Bsp.5)

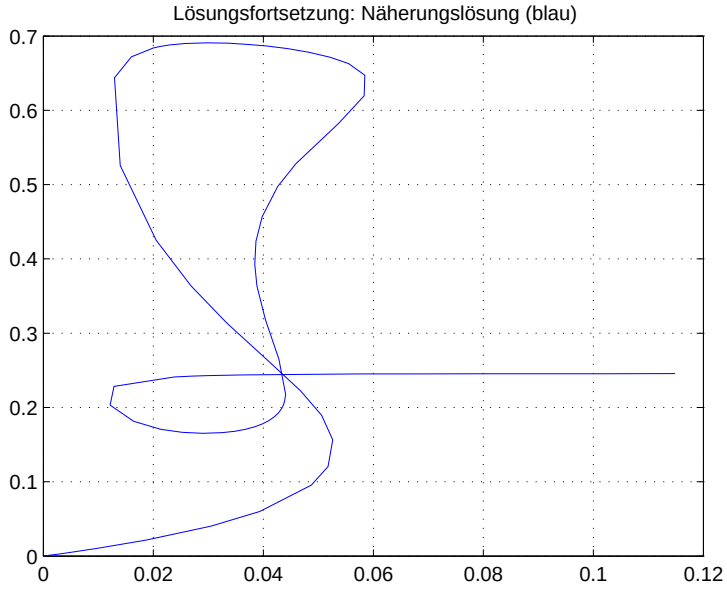


Abbildung 4.20: Nährungslösung : x_2 (Bsp.5)

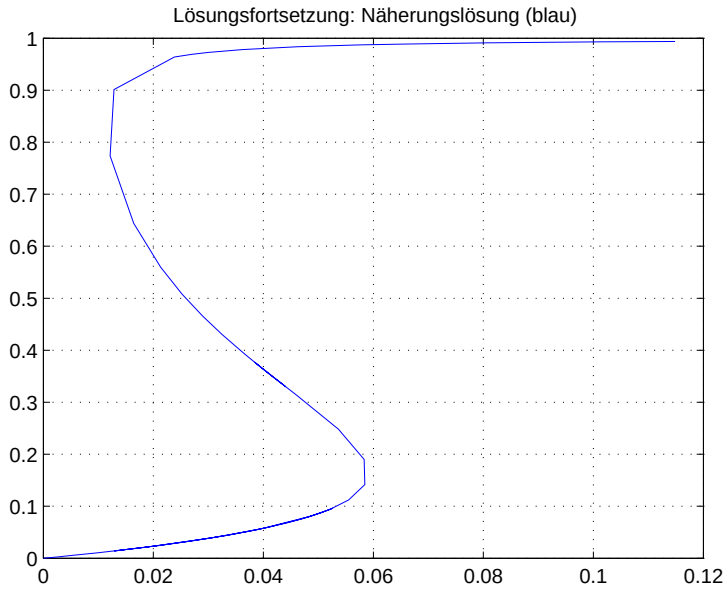


Abbildung 4.21: Nährungslösung : x_3 (Bsp.5)

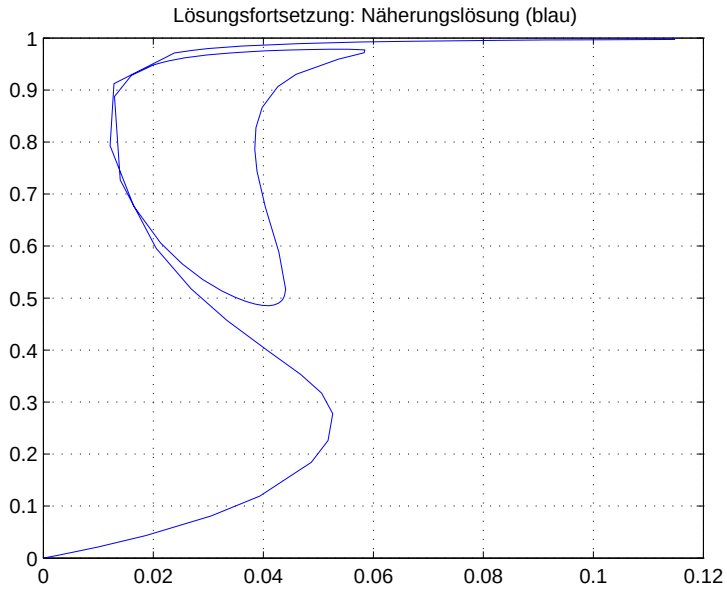


Abbildung 4.22: Nährungslösung : x_4 (Bsp.5)

In Abb.(4.19) - Abb.(4.22) können wir die Lösungen x_1, x_2, x_3 und x_4 finden. Alle Lösungen sind für $\lambda \in [0, 0.12]$ dargestellt. x_2 ist ein gekreuzt Punkt mit einer Projektion. Die Stabilitätsanalyse zeigt uns folgende Abbildung:

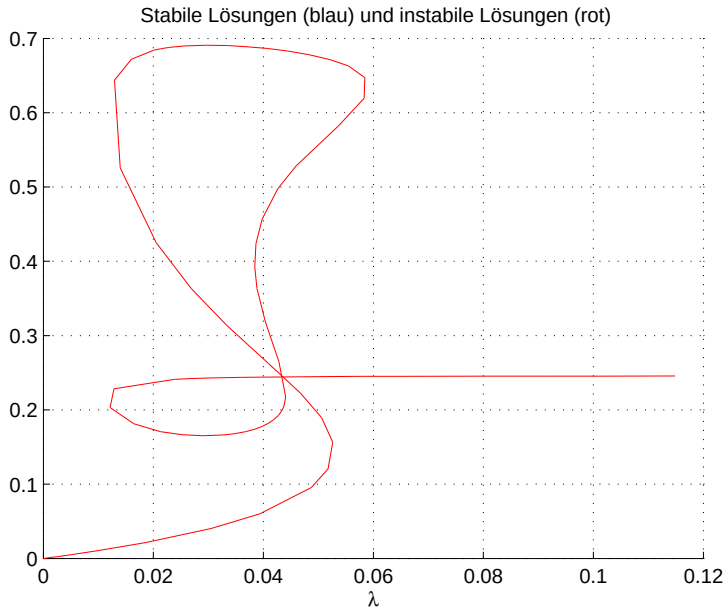


Abbildung 4.23: Stabilität : x_2 (Bsp.5)

4.6 Beispiel 6: Jäger und Beute - Modell

Als System 1. Ordnung ergibt sich

$$\begin{cases} u_1' = 3u_1(1-u_1) - u_1u_2 - \lambda(1-e^{-5u_1}), \\ u_2' = -u_1 + 3u_1u_2. \end{cases}$$

Hier kann man sich u_1 als Fisch und u_2 als Haifisch denken, während durch die Bezeichung $\lambda(1-e^{-5u_1})$ die Fischerei mit Fischerei-Quote λ dargestellt wird.
Wenn die stationären Lösungenen $\lambda = 0$ sind:

$$\begin{cases} 3u_1(1-u_1) - u_1u_2 = 0 \\ -u_1 + 3u_1u_2 = 0 \end{cases}$$

$$\Rightarrow (u_1, u_2) = (0,0), (1,0), (\frac{1}{3}, 2)$$

mit der Jacobi-Matrix

$$J = \begin{pmatrix} 3 - 6u_1 - u_2 - 5\lambda e^{5u_1} - u_1 \\ 3u_2 - 1 + 3u_1 \end{pmatrix} = J(u_1, u_2; \lambda)$$

$$J(0,0;0) = \begin{pmatrix} 3 & 0 \\ 0 & -1 \end{pmatrix}, \quad \text{Eigenwerte} \quad 3, -1 \quad \text{(instabil)}$$

$$J(1,0;0) = \begin{pmatrix} -3 & -1 \\ 0 & 2 \end{pmatrix}, \quad \text{Eigenwerte} \quad -3, 2 \quad \text{(instabil)}$$

$$J(\frac{1}{3},2;0) = \begin{pmatrix} -1 & -\frac{1}{3} \\ 6 & 0 \end{pmatrix}, \quad \text{Eigenwerte} \begin{cases} (-1-\mu)(-\mu)+2=0 \\ \mu^2+\mu+2=0 \\ \mu_\pm = \frac{-1\pm\sqrt{-7}}{2} \\ Re(\mu_\pm) < 0 \end{cases} \quad \text{(stabil)}$$

Alle drei Jacobi-Matrix bei $\lambda = 0$ sind nicht singulär. Damit haben alle drei Punkte von der IFT stationäre Fortsetzungen für (klein) $\lambda \neq 0$.

In diesen Problem können wir alle folgenden Lösungen finden.

I :

$$(u_1, u_2) = (0, 0).$$

II :

$$u_2 = 0, \quad \lambda = \frac{3u_1(1-u_1)}{1-e^{-5u_1}}.$$

$$\lim_{u_1 \to 0} \lambda = \lim_{u_1 \to 0} \frac{3(1-2u_1)}{5e^{-5u_1}} = \frac{3}{5}.$$

III :

$$u_1 = \frac{1}{3}, \quad \frac{2}{3} - \frac{1}{3}u_2 - \lambda(1-e^{-\frac{5}{3}}) = 0 \Rightarrow u_2 = 2 - 3\lambda(1-e^{-\frac{5}{3}}).$$

Dieser Lösungszweig teilt sich auf zwei Zweigpunkte und einer ist $(u_1, u_2, \lambda) = (0, 0, \frac{3}{5})$.

Stabilität der Branche I :

$$J((0,0);\lambda) = \begin{pmatrix} 3-5\lambda & 0 \\ 0 & -1 \end{pmatrix}, \quad \text{Eigenwerte} \quad 3-5\lambda, -1 \quad \text{(instabil)}$$

Damit ist die triviale Lösung instabil für $\lambda < \frac{3}{5}$ und stabil für $\lambda > \frac{3}{5}$. wie Abb. (4.24).
Stabilität der Branche II:
Diese Branche hat keine stabilen positiven Lösungen.
Stabilität der Branche III:
Bei $\lambda_H \approx 0.67$ (die Lösung 8 in der Abb.(4.24) kreuzen die komplexen Eigenwerte der imaginären Achse. Dieser Kreuzung ist die Hopf - Bifurkation. Darüber gibt es eine periodische Lösung λ_H, deren Period T als λ erhöht. Siehe die Figur (4.24) für einige vertretende Orbits. Die Period wird unendlich bei $\lambda = \lambda_\infty \approx 0.07$ sein. Dieser abschließender Orbit ist im Wirklichkeit ein Zyklus.

Aus der Abb. (4.24) können wir die Lösung für die langsame Erhöhung λ ableiten:

−Branch I ist folgend bisλ_H.
−Periodische Lösung der erhöhenden Period ist bis$\lambda = \lambda_\infty$.
−Zusammenbruch der trivialen Lösung (Branch I).

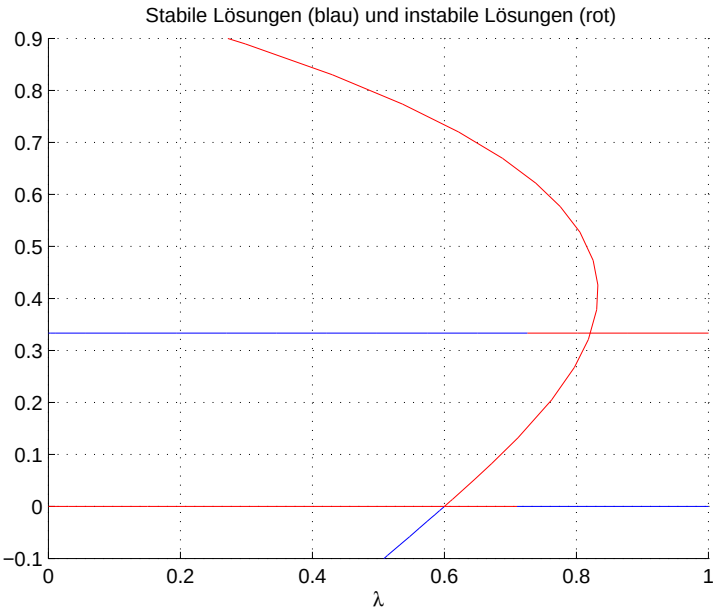


Abbildung 4.24: Bifurkationsdiagramm des Jäger und Beute Modells (Bsp.6)

Dieser periodische Lösungszweig hat also gezeigt, für stationäre Lösung der vertikalen Achse ist u_1 einfach, während für periodische Lösungen $\max u_1$ gezeichnet ist. Feste (oder gedachte) Linien bezeichnen stabile (oder instabile) Lösungen. Offene Plätze sind die Zweigspunkte. Der feste Platz ist die Hopf-Bifurkation.

5 Fazit

Die Aufgabe der Diplomarbeit lautete, parameterabhängige Differentialgleichungssysteme mit Parameter $\lambda \in R$ über die Stabilität zu analysieren.

Ein stabiles Gleichgewicht ist das Ergebnis eines Regelkreises (z.B. beim Aufhängen eines Gegenstandes das abnehmende Pendeln durch Schwerkraft und Reibungseffekte). Führen Abweichungen vom Gleichgewicht zu noch großer Abweichung vom Arbeitspunkt, liegt ein labiles Gleichgewicht vor (etwa beim Balancieren einer Stange). Im vorliegenden Beitrag wird ein numerisches Lösungskonzept für die Stabilitätsanalyse vorgestellt. Ausgehend von Ljapunov angegebenen Stabilitätsdefinitionen werden diese einfach handhabbar sein. Die für je ein Beispiel ermittelten Lösungen werden durch charakteristische Eigenschaften einer Nachbarbewegung berechnet, mittels bewährten Zeitintegrationsverfahren, verifiziert.

Durch die Einstellung der Parameter des Programms, kann man selbst wählen, wie genau und wie effektiv die Kurven bestimmt werden sollen. An verschiedenen Beispielen wurden die Einstellmöglichkeiten der Parameter getestet, um eine Stabilität im Programm festzulegen.

Kapitel 2 setzt sich mit Fragestellungen rund um die Lösungen und deren numerischer Bestimmung auseinander. Dieses Prinzip nennt man Prädiktor-Korrektor-Methode. Die Idee einer effizierteren Lösung besteht in der Verwendung eines Prädiktors für die Ableitungen. Die ermittelte Schätzung für die Ableitung im zu bestinnenden Lösungspunkt dient dann als Iterationsmatrix des Korrektors. Dadurch muß in jedem Fortsetzungsschritt nur eine Ableitung, die im Lösungspunkt, ermittelt werden. Kapital 3 thematisiert die Stabilitäts- und Bifurkationsanalyse und deren Anpassung auf die vorliegende Problemklasse. Kapitel 4 stellt die Anwendungsbeispiele in Form des Matlab Programm dar und bewertet die vorgestellten Verfahren.

6 Anhang

Bestimmung aller Eigenwerte $\mu_1, \cdots, \mu_n$ reeller Matrizen:

Um eine vollständige Stabilitätsanalyse durchführen zu können, ist das vollständige Eigenwertproblem für die Jacobimatrix $A = f_x(x_0, \lambda)$ zu lösen, d.h. es ist das Gesamtspektrum $\sigma(A)$ zu ermitteln. Man unterscheidet

(1) Direktes Verfahren: Bestimmung des charakteristischen Polynoms $P_n(\mu) = \det(A - \mu I)$ (bzw. des Minimalpolynoms) sind ausschließende Ermittlung aller Nullstellen.

(2) Iteratives Verfahren: Unendliche Folge von Ähnlichkeitstransformationen $B = U^{-1}AU$, die A in (Fort-) Dreiecksgestalt überführt und Herauslesen der Eigenwerte.

Während die direkten Verfahren für kleine Probleme ($n \leq 10$) anwendbar sind, finden für beliebige Probleme die iterativen Verfahren Verwendung. Das effizienteste Verfahren zur Lösung des vollständigen Eigenwertproblems ist (mit geeigneten Modifikationen) der QR-Algorithmus.

• Idee des QR-Algorithmus:
– Ziel: Überführung der Matrix A durch eine Folge orthogonaler Ähnlichkeitstransformationen $Q_1, Q_2, \cdots$ in eine obere Block-Dreiecksform B (z.B. für $n = 6$):

$$B = \begin{bmatrix} \boxed{b_{11}} & b_{12} & b_{13} & \cdots\cdots & b_{16} \\ 0 & b_{22} & b_{23} & \cdots\cdots & b_{26} \\ 0 & b_{32} & b_{33} & \cdots\cdots & b_{36} \\ & & 0 & \boxed{b_{44}} \quad \cdots & b_{46} \\ & & & 0 \quad \boxed{b_{55}} & b_{56} \\ & & & 0 \quad \boxed{b_{66}} & \end{bmatrix}, \tag{6.1}$$

$$\text{d.h.} \qquad b_{i+1,i} \cdot b_{i+2,i+1} = 0, \qquad i = 1(1)n-2$$

– Eigenwerte: Aus $D = det(B - \mu I) = 0$ folgt durch sukzessive Entwicklung nach der jeweils 1. Spalte:

$$
\begin{array}{lll}
(1) & b_{11} - \mu = 0 & \Longrightarrow \quad \mu_1 = b_{11} \\
(2) & b_{66} - \mu = 0 & \Longrightarrow \quad \mu_6 = b_{66} \\
(3) & b_{55} - \mu = 0 & \Longrightarrow \quad \mu_5 = b_{55} \\
(4) & b_{44} - \mu = 0 & \Longrightarrow \quad \mu_4 = b_{44} \\
(5) & \begin{vmatrix} b_{22} - \mu & b_{23} \\ b_{32} & b_{33} - \mu \end{vmatrix} = 0 & \Longrightarrow \quad \mu_{2,3} = \alpha \pm i\beta
\end{array}
$$

– Grundidee einer orthogonalen Ähnlichkeitstransformation:
(i) Zerlegung von A in das Produkt einer orthogonalen Matrix Q

$$Q^T Q = I, \qquad \text{also} \qquad Q^{-1} = Q^T \tag{6.2}$$

und einer oberen Dreiecksmatrix R, also

$$A = Q \cdot R \qquad \text{(QR- Zerlegung)} \tag{6.3}$$

(ii) Anschließende Multiplikation in umgekehrter Reihenfolge:

$$A' = R \cdot Q \tag{6.4}$$

liefert wegen $R = Q^{-1}A = Q^T A$

$$A' = R \cdot Q = Q^T A Q, Q^T = Q^{-1} \tag{6.5}$$

folglich geht A' aus A durch eine orthogonale Ähnlichkeitstransformation hervor. Das Spektrum $\sigma(A) = \sigma(A')$ ist invariant.

• QR-Algorithmus (6.1):
S1: Setze $A_1 := A, k := 1$

S2: (QR-Zerlegung) $A_k = Q_k R_k$

S3: (Ähnlichkeitstransformation) $A_{k+1} = R_k Q_k$

S4: (Abbruchtort) $k := k + 1$. Falls in A_k gilt $a_{i+1,i} \cdot a_{i+2,i+1} = 0$,

$i = 1(1)n-2$, so STOP; sonst gehe zu S2.

• **Bemerkungen (6.2):**
(1) Bestimmung von Eigenvektoren:
wegen

$$\begin{aligned} A_{k+1} &= Q_k^T A_k Q_k \\ &= Q_k^T (Q_{k-1}^T A_{k-1} Q_{k-1}) Q_k \\ &= (Q_{k-1}Q_k)^T A_{k-1} (Q_{k-1}Q_k) \\ &\vdots \end{aligned}$$

folgt $A_k = (Q_1 Q_2 \cdots Q_k)^T (Q_1 Q_2 \cdots Q_k)$.
sei $U_k := Q_1 Q_2 \cdots Q_k$, so gilt damit

$$A_k = U_k^T A U_k, k = 1, 2, \cdots \tag{6.6}$$

Falls $\lim_{k \longrightarrow \infty} A_k = \Lambda = diag(\mu_1, \cdots, \mu_n)$, so konvergieren die U_k gegen die Modalmatrix

$$U = (v_1, v_2, \cdots, v_n)$$

der normierten Eigenvektoren. Denn

$$\begin{aligned} \Lambda &= U^T A U \\ AU &= U \cdot \Lambda \qquad \text{bzw.} \\ Av_i &= \mu_i \cdot v_i, i = 1(1)n. \end{aligned} \tag{6.7}$$

Für einen Spezialfall gilt folgender:

• **Satz (6.3):**
Falls alle μ-paarweise verschieden sind und A ein vollständiges ONS von Eigenvektoren v_i besitzt, so gilt für die Folgen A_k und U_k gemäß (6.6)

$$(i) \quad \lim_{k \longrightarrow \infty} U_k = V = (v_1, v_2, \cdots, v_n) \tag{6.8}$$

$$(ii) \quad \lim_{k \longrightarrow \infty} A_k = diag(\mu_1, \mu_2, \cdots, \mu_n), \quad \text{mit} \quad |\mu_1| \geq |\mu_2| \geq \cdots |\mu_n|. \tag{6.9}$$

(2) QR-Zerlegung einer Matrix
Diese Grundaufgabe erfolgt durch schnelle Rotationen in $O(n^2)$ Operationen.

(3) Transformation und obere Hessenberg-Form:
Überführt man durch die sog. Hessenberg-Transformation in $O(n^2)$-Schritten die Ausgangsmatrix A in die obere Hessenbergform

$$A' = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots\cdots\cdots & a_{1,n-1} & a_{1n} \\ -1 & a_{22} & a_{23} & \cdots\cdots\cdots & a_{2,n-1} & a_{2n} \\ 0 & -1 & a_{33} & \cdots\cdots\cdots & a_{3,n-1} & a_{3n} \\ 0 & 0 & -1 & \cdots\cdots\cdots & a_{4,n-1} & a_{4n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots\cdots\cdots & -1 & a_{nn} \end{bmatrix}, \tag{6.10}$$

so bleibt diese bei allen orthogonalen Ähnlichkeitstransformationen des QR-Algorithmus erhalten.

(4) Verbesserung der Grundform (6.9)
Durch Spektralverschiebungen der Form

$$\mathcal{B}_k = A_k - \sigma_k \cdot I \tag{6.11}$$

mit geeigneten Eigenwertnäherungen σ_k erhält man eine wesentliche Konvergenzbeschleunigung. Für die Behandlung konjungiert komplexer Eigenwerte vermeidet man das Rechnen mit komplexen Zahlen, indem ein Doppelschritt

$$\mathcal{B}_{k+2} = Q_{k+1}^T Q_k^T \mathcal{B}_k Q_k Q_{k+1} \tag{6.12}$$

mit einer komplexen und der damit konjungiert komplexen Verschiebung ausgeführt wird.

A Programmtext

A.1 XCONT

```
function [lambda,s,charmult,bif] =   xcont(f,I,s0,opt)
% Pseudo-arclength continuation for the computation of equilibrium
% solutions of systems of differential equations dx/dt=f(x,lambda)
%
% inputs:
% f   right hand side of diffential equation [function handle]
% I   Continuation intervall for parameter lambda
      [ m-dimensional row vector, m =2|3]
%     m=2: initial parameter I(1).
%     cont direction:I=[min parameter value, max parameter value]
                      -> positive lambda-direction.
%     I=[max parameter value, min parameter value]
                      -> negative lambda-direction.
%     m=3: initial parameter I(2).
%     cont direction:I=[min parameter value, nitial parameter value,
          max parameter value] -> positive lambda-direction.
%     I=[max parameter value, initial parameter value,
        min parameter value] -> negative lambda-direction.
% s0  Initial approximation of the solution for the initial parameter
      [ n-dimensional column vector ]
% opt  Options (optional) [ structure ]
%               View contset for further details.
%
% outputs:
% lambda   Parameter values of the computed solutions [row vector]
% s        Values of the computed solutions [(n,length(lambda))-matrix]
% charmult Eigenvalues of the solutions(optional)[(n,length(lambda))-matrix]
% bif      Computed bifurcation points with details (optional)[structure]
%

% Defaults
if nargin<4
    error('Not enough input arguments!')
end
```

```
if (nargin == 3)
    opt=xcontset;
end

if length(I)==3 & I(1)<=I(2) & I(2)<=I(3)
    lambda(1)   =   I(2);
    lambdamin   =   I(1);
    lambdamax   =   I(3);
elseif length(I)==2 & I(1)<=I(2)
    lambdamin   =   I(1);
    lambdamax   =   I(2);
    lambda(1)   =   I(1);
elseif length(I)==3 & I(1)>=I(2) & I(2)>=I(3)
    lambda(1)   =   I(2);
    lambdamin   =   I(3);
    lambdamax   =   I(1);
elseif length(I)==2 & I(1)>=I(2)
    lambdamin   =   I(2);
    lambdamax   =   I(1);
    lambda(1)   =   I(1);
else
    error('Second input argument is not an interval!')
end

cor         =   opt.Cor;
maxit       =   opt.MaxNewSteps;
tol         =  opt.SolTol;
maxstepsize =  opt.MaxStepSize;
minstepsize =  opt.MinStepSize;
tancon      =   1-opt.TanCon;
al_end      =  opt.MaxArcLength;

% initial values and constants
format long e
h           =   sqrt(eps);    % discretisation step size (standard)
n           =   length(s0);   % ode dimension
n1          =   n+1;          % ode dimension + 1
```

```
j           =   1;            % continuation counter
jj          =   1;            % counter continuation steps with old
                                iteration matrix(cor='nl2')
s(:,1)      =   s0;           %
H           =   zeros(n1,1);  % H(y)= H(s,lambda)=(f(s,lambda))
                              %                   (tvp' (s-s0))
DH          =   zeros(n1,n1); % DH(y) = (df/ds  df/d lambda)
                              %         (tvp'  0  ),
                              % tvp-tangent vector in predictor point
al(1)       =   0;            % arclength parameter
ha(1)       =   abs(I(2)-I(1))/100;          %initial step size
lb          =   1;            % counter bifurcations
tvp         =   zeros(n1,1);  % tangent vector in predictor point
d           =   ones(n1,1);   % newton correction vector
kk          =   0;            % counter newton method
mu          =   1;            % factor newton method
charmult    =   [];
bif         =
struct('lam',[],'s',[],'char',[],'tv',[],'type',''); pre =0;

% Initialisation status bar
wb = waitbar(0,'cont is calculating solutions');tic;
fprintf('\ncont is calculating solutions for\n') display(f);
fprintf('with following options:\n\n') display(xcontset(opt));

% correction of s0 with damped newton method
[s(:,1),DH(1:n,1:n),suc]=xshoot_pepar(f,s(:,1),lambda(1),1e-6);

if suc==1
    fprintf('\nInitial solution found.\n')
else
    fprintf('\nNo initial solution found.\n')
    lambda=[];
    s=[];
    tv=[];
    return
end
```

```
% computation of tangent vector in [s0; lambda0]
% dx/d(lambda)-Approximation
x           =   feval(f,s(:,j),lambda(j)); x_lambda = x;

hlambda     =   h*(1+abs(lambda(j))); x  =
feval(f,s(:,j),lambda(j)+hlambda);

DH(1:n,:)   =   [DH(1:n,1:n)  (x-x_lambda)/hlambda];

tv(:,1)          =   null(DH(1:n,1:n1));
if (I(2)-I(1))*tv(n1,1)<0
% secures continuaton in pos.(neg.)lambda-direction
    tv(:,1)=-tv(:,1);
end;

% stability analysis  ->  \"{U}berarbeiten!!!!!
if nargout>=3
    charmult(:,j)=eig(DH(1:n,1:n));

    % bifurcaton detection
    if nargout==4
    re      =2*max(sign(imag(charmult(:,j))),0)-1;
    au(:,j)=[real(prod(charmult(:,j)-1))      % fold
             real(prod(charmult(:,j)+1))      % flip
             prod(abs(charmult(:,j))-re)                % torus
             real(prod(charmult(:,j)-1))*sign(tv(n1,j)) % pitchfork
             sum(re)
             sign(tv(n1,j))          ];
    end
end   % stability analysis

% continuation
while (al(j)<abs(al_end)) & (lambda(j)<=lambdamax) &
(lambda(j)>=lambdamin)
    j       =   j+1;
    al(j)   =   al(j-1)+ha(j-1);

    % refresh status bar
```

```
waitbar(al(j)/abs(al_end))

% predictor
if j>3 & pre==0
    if nargout<3    % quadratic Hermite predictor
        a12=[s(:,j-1)-s(:,j-2);lambda(j-1)-lambda(j-2)]./ha(j-2);
        a23=[s(:,j-2)-s(:,j-3);lambda(j-2)-lambda(j-3)]./ha(j-3);
        a123=(a12-a23)/(al(j-1)-al(j-3));
        s(:,j=s(:,j-3)+a23(1:n).*(al(j)-al(j-3))+a123(1:n).
                        *(al(j)-al(j-3)).*(al(j)-al(j-2));
        lambda(j)=lambda(j-3)+a23(n1).*(al(j)-al(j-3))+a123(n1).
                         *(al(j)-al(j-3)).*(al(j)-al(j-2));
    else            % cubical Hermite predictor
        a12= [s(:,j-1)-s(:,j-2);lambda(j-1)-lambda(j-2)]./ha(j-2);
        a122=(a12-tv(:,j-2))./ha(j-2);
        a112=(tv(:,j-1)-a12)./ha(j-2);
        a1122=(a112-a122)./ha(j-2);
        s(:,j)=s(:,j-2) +   (al(j)-al(j-2)) .*  tv(1:n,j-2)+
              (al(j)-al(j-2))^2   .*  a122(1:n,1) +
              al(j)-al(j-2))^2*(ha(j-1))  .*  a1122(1:n,1);
        lambda(j)=lambda(j-2) +   (al(j)-al(j-2)) .*  tv(n1,j-2)+
                  (al(j)-al(j-2))^2   .*  a122(n1,1) +
                  (al(j)-al(j-2))^2*(ha(j-1))  .*  a1122(n1,1);
    end
else % tangent/secant predictor
    s(:,j)      =   s(:,j-1)    +   ha(j-1)  .*   tv(1:n,j-1);
    lambda(j)   =   lambda(j-1) +   ha(j-1)  .*   tv(n1,j-1);
    DH(n1,:)    =   tv(:,j-1)';
end
praed = [s(:,j); lambda(j)];
% corrector
if (cor=='nl1') | (jj>5) |  abs(tv(n1,j-1))<1/(2*(sqrt(n1)))
                    % correction with new iteration matrix
    jj  =   1;
    DH(1:n,:)   =   xnumdiff(f,lambda(j),s(:,j))
    tvp         =   (DH\[zeros(n,1);1]);
    tvp         =   tvp/norm(tvp);
else  % correction with old iteration matrix
```

```
    tvp =   tv(:,j-1);
    jj  =   jj+1;
end

% check tangent constraint a priori (only if jj=1)
if j>2&(tv(:,j-1)'*tvp)<tancon % new prediction with half stepsize
    j   =   j-1;
    ha(j)  =   0.5*ha(j);

    if ha(j)<minstepsize   % break if stepsize too small
       fprintf('\nStepwidth to small for continuation.')
       break
    end
else % correction
    % QR-decomposition of iteration matrix
    [Q,R]   =   qr(DH);
    d       =   ones(n1,1);
    kk      =   0;
    mu      =   1;

    % damped newton-like method
    while (norm(d,inf)>tol*(1+norm([s(:,j);lambda(j)])))&(kk<=maxit)
        mu =min(2*mu,1);
        if kk==0
            x       =   feval(f,s(:,j),lambda(j));
            H       =   [x ; 0];
            d       =   R\(Q'*H);
            ytest   =   [s(:,j);lambda(j)]-mu*d(:,1);
        else
            d       =   dtest;
            ytest   =   [s(:,j);lambda(j)]-mu*d(:,1);
        end

        x       =   feval(f,ytest(1:n),ytest(n1));
        Htest   =   [x; tvp'*(ytest-praed)];
        dtest   =   R\(Q'*Htest);

        while norm(dtest)>norm(d)
```

```
        mu=.5*mu;
        if mu<1e-6
             kk=maxit;
             break
        end
        ytest=[s(:,j);lambda(j)]-mu*d(:,1);
        x           =   feval(f,ytest(1:n),ytest(n1));
        Htest       =   [x ; tvp'*(ytest-praed) ];
        dtest       =   R\(Q'*Htest);
    end

    s(:,j)=ytest(1:n);
    lambda(j)=ytest(n1);
    if cor=='new'
    [Q,R]=qr([xnumdiff(f,lambda(j),s(:,j));tv(:,j-1)']);
    end

    kk=kk+1;

end % while newton

% approximation of new tangent vector
if nargout>2 % tangent
    DH(1:n,:)   =   xnumdiff(f,lambda(j),s(:,j))
    tv(:,j)     =   (DH\[zeros(n,1);1]);
    tv(:,j)     =   tv(:,j)/norm(tv(:,j));
    jj          =   1;
else % secant
    tv(:,j)     = [s(:,j)-s(:,j-1);lambda(j)-lambda(j-1)]/norm
                    ([s(:,j)-s(:,j-1);lambda(j)-lambda(j-1)]);
end

DH(n1,:)    =   tv(:,j)';

% check tangent constraint
if(tv(:,j-1)'*tv(:,j))<tancon|kk>maxit
% new prediction with half step size
```

```
        j   =   j-1;
        ha(j)  =   0.5*ha(j)
        jj  =   1;

        if ha(j)<minstepsize   % break if stepsize too small
            fprintf('\nStepwidth to small for continuation.')
            break
        end

    else
        % correct stepsize and arclength parameter
        ha(j-1)=norm([s(:,j)-s(:,j-1);lambda(j)-lambda(j-1)]);
        al(j)=al(j-1)+ha(j-1);

        % stability analysis  -->  Überarbeiten!!!!!
        if nargout>=3

            charmult(:,j)=eig(DH(1:n,1:n));

            % bifurcation detection
            if nargout==4
            re      =   2*max(sign(imag(charmult(:,j))),0)-1;
            tau(:,j)=[prod(charmult(:,j)-1)          % fold
                      prod(charmult(:,j)+1)          % flip
                      prod(abs(charmult(:,j))-re)    % torus
                      prod(charmult(:,j)-1)*sign(tv(n1,j))%pitchfork
                      sum(re)
                      sign(tv(n1,j))                    ];

                % bifurcationintervall found
                lbalt=lb;
                if tau(1,j)*tau(1,j-1)<0 & tau(6,j)*tau(6,j-1)<0
                    fprintf('\nfold bifurcation       ')
                    [bif(lb)]=bifdet(f,T,lambda(j-1:j),s(:,j-1:j),
                    tau(1,j-1:j),tv(:,j-1:j),'fold',opt);
                    lb=lb+1;
                elseif tau(2,j)*tau(2,j-1)<0
                    fprintf('\nflip bifurcation       ')
```

```
    [bif(lb)]=bifdet(f,T,lambda(j-1:j),s(:,j-1:j),
    tau(2,j-1:j),tv(:,j-1:j),'flip',opt);
    lb=lb+1;
elseif tau(3,j)*tau(3,j-1)<0 & tau(5,j)==tau(5,j-1)
    fprintf('\ntorus bifurcation      ')
    [bif(lb)]=bifdet(f,T,lambda(j-1:j),s(:,j-1:j),
    tau(3,j-1:j),tv(:,j-1:j),'torus',opt);
    lb=lb+1;
elseif tau(1,j)*tau(1,j-1)>0 & tau(6,j)*tau(6,j-1)<0;
    fprintf('\npitchfork bifurcation  ')
    [bif(lb)]=bifdet(f,T,lambda(j-1:j),s(:,j-1:j),
    tau(6,j-1:j),tv(:,j-1:j),'pitchfork1',opt);
    lb=lb+1;
elseif tau(1,j)*tau(1,j-1)<0 & tau(6,j)*tau(6,j-1)>0
    fprintf('\npitchfork bifurcation  ')
    [bif(lb)]=bifdet(f,T,lambda(j-1:j),s(:,j-1:j),
    tau(1,j-1:j),tv(:,j-1:j),'pitchfork2',opt);
    lb=lb+1;
end
pre=0;
if lb>lbalt % integratin of new bifurcation point in
    output solution curve
    fprintf('found at lambda = %2.6f .\n',bif(lb-1).lam)
    pre             =  1;
    lambda(j+1)     =  lambda(j);
    s(:,j+1)        =  s(:,j);
    tv(:,j+1)       =  tv(:,j);
    charmult(:,j+1) =  charmult(:,j);
    lambda(j)    =  bif(lbalt).lam;
    s(:,j)       =  bif(lbalt).s;
    tv(:,j)      =  bif(lbalt).tv;
    charmult(:,j)=  bif(lbalt).char;
    ha(j-1)      =  norm([s(:,j)-s(:,j-1);
                    lambda(j)-lambda(j-1)]);
    ha(j)        =  norm([s(:,j+1)-s(:,j);
                    lambda(j+1)-lambda(j)]);
    al(j)        =  al(j-1)+ha(j-1);
    al(j+1)      =  al(j)+ha(j);
```

```
                    j            =  j+1;
                end
            end
        end    % stability analysis


        % step size control

        rho     =   max(min((4+jj)/kk,2),0.5);
        ha(j)   =   min(opt.MaxStepSize,rho*ha(j-1));

        if ha(j)<minstepsize   % break if stepsize too small
            fprintf('\nStepwidth to small for continuation.\n')
            break
        elseif (tv(:,1)'*tv(:,j)>0.9) & norm([s(:,1)-s(:,j);
               lambda(1)-lambda(j)])
               <2*ha(j-1) & j>10%break if closed curve detected
            fprintf('\nClosed Curve detected.\n')
            lambda(j+1)=lambda(1);
            s(:,j+1)=s(:,1);
            tv(:,j+1)=tv(:,1);
            if nargout>2
                charmult(:,j+1)=charmult(:,1);
            end
            break
        end
    end % if tangent constraint a priori
end % if tangent constraint
if ~isempty(opt.SavePath)
    save(opt.SavePath,'lambda','s','charmult','bif');
end
end % while continuation


% close status bar
close(wb)
```

```
% output

if j>1
    fprintf('\nContinuation finished.\n')
    fprintf('\nElapsed time:  %3.2f secs\n',toc)
    fprintf('\nNumber of calculated solutions: %4.0f \n',j)
else
    fprintf('\nContinuation aborted.\n')
    fprintf('\nElapsed time:  %3.2f secs\n',toc)
    fprintf('\nNumber of calculated solutions:   0\n')
end
```

A.2 xnumdiff

```
function A=xnumdiff(f,lambda,s)
% numerical approximation of df/dx and df/dlam for a solution
% of the system of differential equations dx/dt = f(x,lam).
%
% inputs:
% f       right hand side of diffential equation [function handle]
% n       dimension of f [positive integer]
% lambda  parameter value [real]
% s       Value of solutions x [n-dimensional column vector]
%
% outputs:
% A             Jacobian matrix [ (n,n+1)-matrix ]
%
n   =   length(s);     % ode dimension
n1  =   n+1;
sig =   zeros(n,n1);
A=sig; h=sqrt(eps);

xs  =   feval(f,s,lambda); for j=1:n
    hh(j)   =   h*(1+abs(s(j)));
    sig(:,j)=   s;
    sig(j,j)=   sig(j,j)+hh(j);
    x       =   feval(f,sig(:,j),lambda);
    A(:,j)  =   (x - xs)/hh(j);
end

hlambda =   h*(1+abs(lambda)); x       =
feval(f,s,lambda+hlambda); A(:,n1) =  (x - xs)/hlambda;
```

A.3 xshoot-pepar

```
function [s,M,suc] = xshoot_pepar(f,s0,lambda,tol)
% computation of equilibrium solutions of differential equations
% dx/dt = f(x,lambda), with fixed parameter lambda
%
% inputs:
% f       right hand side of diffential equation [function handle]
% s0      Initial approximation of the solution
          [n-dimensional column vector]
% lambda  Value of the fixed parameter lambda [float]
%
% outputs:
% s         Computed solution [n-dim column vector]
% M         Jacobian matrix [ (n,n) - matrix ]
% suc       flag [ 1 | 0 ]
%              1 - solution found.
%              0 - not found.

% Defaults
format long e
maxit   =   40;             % max. number of newton steps
h       =   sqrt(eps);      % discretization step size
n       =   length(s0);     % ode dimension
n1      =   n+1;            %
k       =   0;              % counter newton steps
s       =   s0;             %
hh      =   ones(1,n);      %
sig     =   zeros(n,n1);    %
d       =   ones(n,1);      %
mu      =   1; suc     =   1;

% Newton method
while (norm(d)>tol*(1+norm(s)))
    mu =min(2*mu,1);

    if k==0
        x =   feval(f,s,lambda);
```

```
    end;
    sig(:,n1)   =   x;
    x_alt       =   x;

    for j=1:n
        hh(j)   =   h*(1+abs(s(j)));
        sig(:,j)=   s;
        sig(j,j)=   sig(j,j)+hh(j);

        x       =   feval(f,sig(:,j),lambda);
        sig(:,j)=   x;
        M(:,j)  =   (sig(:,j)-sig(:,n1))/hh(j);
     end

     d   =   M \ sig(:,n1);
     y   =   s - d;
     x   =   feval(f,y,lambda);

     while norm(x)^2 > (1-2/100*mu)*norm(x_alt)^2
        mu =   0.5*mu;
        y  =   s - mu*d;
        x  =   feval(f,y,lambda);
        if mu < 1e-6
            k=maxit;
            break
        end
     end % while

     s   =   y;
     k   =   k+1;

     if k >= maxit
         suc = 0;
         break
     end
end % while
```

A.4 xstart

```
function s = xstart
% cont-Aufrufe aller berechneten L\"{o}sungszweige mit
  unterschiedlichen Optionsvarianten
% Einfach entsprechende Kommentarmarkierungen entfernen,
  Optionen ggf. anpassen und abspeichern
% Mittels Aufruf von start wird die gew\"{u}nschte
  Berechnung durchgef\"{u}hrt



% Beispiel 1:  Eindimensionales Problem von Vogt (2008)
% ----------------------------------------------------

% L\"{o}sungszweig
% global p
% p = 2.0
% [lambda,s,eigen]=xcont(@bsp1,[-10 20],[-1.01], xcontset
                  ('SolTol',1e-8,'MaxStepSize',.2,
                  'MinStepSize' ,1e-5,  'MaxArcLength', 100 ,
                  'TanCon'   ,.5e-1,   'OdeSol'   ,@dopri87,
                  'OdeOpt',odeset('RelTol',1e-8,'AbsTol',1e-8),
                  'BifTol'   ,1e-4,'BifOdeSol'   ,@dopri87,
                  'BifOdeOpt'   ,odeset('RelTol',1e-10,
                  'AbsTol',1e-10),'Cor','nl2','MaxNewSteps',10,
                  'MaxRegSteps'   ,10, 'SavePath',''));
% plot(lambda,s(1,:)), title('L\"{o}sungsfortsetzung:
  N\"{a}herungsl\"{o}sung (blau)')
% grid on
% disp('ENTER druecken!'); pause
% xx = -0.1:0.05:2.1;
% yy = par(xx,p);
% plot(yy,xx,'b.',lambda,s(1,:),'r'), title('Exakte L\"{o}sung
      (blau) und N\"{a}herungsl\"{o}sung (rot)')
% grid on
% disp('ENTER druecken!'); pause
% plot(lambda,eigen,'r.'), title('Eigenwerte der
```

```
  Jacobimatrix \"{u}ber lambda')
% grid on
% disp('ENTER druecken!'); pause
%
% figure(2);
%     h=gca;
%     hold on
%     grid on
%     xlabel('\lambda');
%     title('Stabile L\"{o}sungen (blau) und instabile
      L\"{o}sungen (rot)');
% for i = 1:length(lambda)
%   if max(real(eigen(:,i)))<0
%       plot(lambda(i),s(1,i),'b.');
%   else
%       plot(lambda(i),s(1,i),'r.');
%   end
% end
% disp('ENTER druecken!'); pause
%
% figure(3);
%     hold on
%     grid on
%     xlabel('\lambda');
%     title('Stabile L\"{o}sungen (blau) und instabile
      L\"{o}sungen (rot)');
% for p = 0:0.5:2.0
%   [lambda,s,eigen]=xcont(@bsp1,[-20 30],[-1.01], xcontset
                  ('SolTol',1e-8,'MaxStepSize',.2,'MinStepSize'
                  ,1e-5,'MaxArcLength',100 , 'TanCon',.5e-1,
                  'OdeSol'   ,@dopri87,  'OdeOpt',odeset('RelTol',
                  1e-8,'AbsTol',1e-8),'BifTol',1e-4,'BifOdeSol',
                  @dopri87,  'BifOdeOpt'   ,odeset('RelTol',1e-10,
                  'AbsTol',1e-10), 'Cor','nl2', 'MaxNewSteps',
                     10, 'MaxRegSteps'   ,10, 'SavePath',''));
%   for i = 1:length(lambda)
%      if max(real(eigen(:,i)))<0
%          plot(lambda(i),s(1,i),'b.');
```

```
%      else
%          plot(lambda(i),s(1,i),'r.');
%      end
%   end
%   hold on
%   grid on
%   disp('ENTER druecken!'); pause
% end
% hold off




% Beispiel 2:  Eindimensionales Problem
% ------------------------------------------------
%
% L\"{o}sungszweig
% [lambda,s,eigen]=xcont(@bsp1,[-5 5],[0.0],xcontset('SolTol',
                  1e-8,'MaxStepSize',.2, 'MinStepSize' ,1e-5,
                  'MaxArcLength',100 ,  'TanCon'   ,.5e-1,
                  'OdeSol'   ,@dopri87,  'OdeOpt',odeset
                  ('RelTol',1e-8,'AbsTol',1e-8),'BifTol',1e-4,
                  'BifOdeSol',@dopri87,'BifOdeOpt',odeset('RelTol',
                  1e-10,'AbsTol',1e-10),'Cor','nl2', 'MaxNewSteps',
                    10,    'MaxRegSteps'   ,10, 'SavePath',''));
% plot(lambda,s(1,:)), title('L\"{o}sungsfortsetzung:
  N\"{a}herungsl\"{o}sung (blau)')
% grid on
% disp('ENTER druecken!'); pause
% xx = -0.1:0.05:2.1;
% yy = par(xx,p);
% plot(yy,xx,'b.',lambda,s(1,:),'r'), title('Exakte
       L\"{o}sung (blau) und N\"{a}herungsl\"{o}sung (rot)')
% grid on
% disp('ENTER druecken!'); pause
% plot(lambda,eigen,'r.'), title('Eigenwerte der
       Jacobimatrix \"{u}ber lambda')
% grid on
```

```
% disp('ENTER druecken!'); pause
%
% figure(2);
%     h=gca;
%     hold on
%     grid on
%     xlabel('\lambda');
%     title('Stabile L\"{o}sungen (blau) und instabile
      L\"{o}sungen (rot)');
% for i = 1:length(lambda)
%   if max(real(eigen(:,i)))<0
%       plot(lambda(i),s(1,i),'b.');
%   else
%       plot(lambda(i),s(1,i),'r.');
%   end
% end
% disp('ENTER druecken!'); pause
% hold off
```

```
% Beispiel 3:  Fuenfdimensionales Problem
% -------------------------------------
%
% L\"{o}sungszweig
% global u
% u =0
% [lambda,s,eigen]=xcont(@bsp3,[-1.0 1.0],[0.;0.;0.;0.;0.],
                  contset('SolTol',1e-8,'MaxStepSize',.2,
                  'MinStepSize' ,1e-5,  'MaxArcLength', 100 ,
                   'TanCon'   ,.5e-1,   'OdeSol'   ,@dopri87,
                    'OdeOpt'   ,odeset('RelTol',1e-8,'AbsTol',
                    1e-8),   'BifTol'   ,1e-4,  'BifOdeSol' ,
                   @dopri87,  'BifOdeOpt'   ,odeset('RelTol',
                   1e-10,'AbsTol',1e-10),'Cor' ,'nl2','MaxNewSteps',
```

```
                    10, 'MaxRegSteps' ,10, 'SavePath',''));
% plot(lambda,s(1,:)), title('L\"{o}sungsfortsetzung:
       N\"{a}herungsl\"{o}sung (blau)')
% grid on
% disp('ENTER druecken!'); pause
% figure(2)
% plot(lambda,s(2,:)), title('L\"{o}sungsfortsetzung:
       N\"{a}herungsl\"{o}sung (blau)')
% grid on
% disp('ENTER druecken!'); pause
% figure(3)
% plot(lambda,s(3,:)), title('L\"{o}sungsfortsetzung:
       N\"{a}herungsl\"{o}sung (blau)')
% grid on
% disp('ENTER druecken!'); pause
% figure(4)
% plot(lambda,s(4,:)), title('L\"{o}sungsfortsetzung:
       N\"{a}herungsl\"{o}sung (blau)')
% grid on
% disp('ENTER druecken!'); pause
% figure(5)
% plot(lambda,s(5,:)), title('L\"{o}sungsfortsetzung:
       N\"{a}herungsl\"{o}sung (blau)')
% grid on
% disp('ENTER druecken!'); pause
% plot(lambda,eigen,'r.'), title('Eigenwerte der
       Jacobimatrix \"{u}ber lambda')
% grid on
% disp('ENTER druecken!'); pause
%
% figure(6);
%     h=gca;
%     hold on
%     grid on
%     xlabel('\lambda');
%     title('Stabile L\"{o}sungen (blau) und instabile
      L\"{o}sungen (rot)');
% for i = 1:length(lambda)-1
```

```
%   if max(real(eigen(:,i)))<0
%       plot([lambda(i),lambda(i+1)],[s(4,i),s(4,i+1)],'b');
%   else
%       plot([lambda(i),lambda(i+1)],[s(4,i),s(4,i+1)],'r');
%   end
% end
% disp('ENTER druecken!'); pause
%
% hold off




% Beispiel 4:  Vierdimensionales Problem
% --------------------------------------
%
% L\"{o}sungszweig

% [lambda1,s1,eigen1]=xcont(@bsp5,[1.3 0.5],[3.03;3.03;0.0;0.0],
                      contset('SolTol',1e-8,'MaxStepSize',.2,
                      'MinStepSize' ,1e-5,  'MaxArcLength', 100,
                      'TanCon'   ,.5e-1,   'OdeSol'   ,@dopri87,
                      'OdeOpt',odeset('RelTol',1e-8,'AbsTol',1e-8),
                      'BifTol',1e-4,  'BifOdeSol',@dopri87,'BifOdeOpt',
                      odeset('RelTol',1e-10,'AbsTol',1e-10),'Cor' ,
                      'nl2', 'MaxNewSteps',10,'MaxRegSteps',
                      10,'SavePath',''));
% plot(lambda1,s1(1,:)), title('L\"{o}sungsfortsetzung:
  N\"{a}herungsl\"{o}sung (blau)')
% [lambda2,s2,eigen2]=xcont(@bsp5,[1.3 0.5],[8.03;0.0;0.0;0.0],
                      contset('SolTol',1e-8,'MaxStepSize',.2,
                      'MinStepSize' ,1e-5,  'MaxArcLength', 100 ,
                      'TanCon'   ,.5e-1,   'OdeSol'   ,@dopri87,
                      'OdeOpt'   ,odeset('RelTol',1e-8,'AbsTol',1e-8),
                      'BifTol'   ,1e-4,  'BifOdeSol',@dopri87,
                      'BifOdeOpt'   ,odeset('RelTol',1e-10,'AbsTol',
                      1e-10),'Cor','nl2', 'MaxNewSteps,10,'MaxRegSteps',
```

```
                    10, 'SavePath','));
% plot(lambda2,s2(1,:)), title('L\"{o}sungsfortsetzung:
  N\"{a}herungsl\"{o}sung (blau)')
% grid on
% disp('ENTER druecken!'); pause
%
% figure(2);
%     h=gca;
%     hold on
%     grid on
%     xlabel('\lambda');
%     title('Stabile L\"{o}sungen (blau) und instabile
      L\"{o}sungen (rot)');
% for i = 1:length(lambda1)-1
%   if max(real(eigen1(:,i)))<0
%       plot([lambda1(i),lambda1(i+1)],[s1(1,i),s1(1,i+1)],'b');
%   else
%       plot([lambda1(i),lambda1(i+1)],[s1(1,i),s1(1,i+1)],'r');
%   end
% end
% for i = 1:length(lambda2)-1
%   if max(real(eigen2(:,i)))<0
%       plot([lambda2(i),lambda2(i+1)],[s2(1,i),s2(1,i+1)],'b');
%   else
%       plot([lambda2(i),lambda2(i+1)],[s2(1,i),s2(1,i+1)],'r');
%   end
% end
% disp('ENTER druecken!'); pause
%
% hold off



% Beispiel 5:  Vierdimensionales Problem
% -------------------------------------
%
% L\"{o}sungszweig
```

```
% [lambda,s,eigen]=xcont(@bsp2,[0.0 0.11],[0.;0.;0.;0.],
                   contset('SolTol',1e-8,'MaxStepSize',
                   .2, 'MinStepSize' ,1e-5,  'MaxArcLength',
                   100 ,  'TanCon',.5e-1,'OdeSol',@dopri87,
                   'OdeOpt',odeset('RelTol',1e-8,'AbsTol',1e-8),
                    'BifTol' ,1e-4, 'BifOdeSol',@dopri87,
                    'BifOdeOpt'   ,odeset('RelTol',1e-10,
                    'AbsTol',1e-10),'Cor' ,'nl2', 'MaxNewSteps',
                    10,'MaxRegSteps',10, 'SavePath',''));
% plot(lambda,s(1,:)), title('L\"{o}sungsfortsetzung:
  N\"{a}herungsl\"{o}sung (blau)')
% grid on
% disp('ENTER druecken!'); pause
% figure(2)
% plot(lambda,s(2,:)), title('L\"{o}sungsfortsetzung:
  N\"{a}herungsl\"{o}sung (blau)')
% grid on
% disp('ENTER druecken!'); pause
% figure(3)
% plot(lambda,s(3,:)), title('L\"{o}sungsfortsetzung:
  N\"{a}herungsl\"{o}sung (blau)')
% grid on
% disp('ENTER druecken!'); pause
% figure(4)
% plot(lambda,s(4,:)), title('L\"{o}sungsfortsetzung:
  N\"{a}herungsl\"{o}sung (blau)')
% grid on
% disp('ENTER druecken!'); pause
% plot(lambda,eigen,'r.'), title('Eigenwerte der
  Jacobimatrix \"{u}ber lambda')
% grid on
% disp('ENTER druecken!'); pause
%
% figure(5);
%     h=gca;
%     hold on
%     grid on
%     xlabel('\lambda');
```

```
%     title('Stabile L\"{o}sungen (blau) und
      instabile L\"{o}sungen (rot)');
% for i = 1:length(lambda)-1
%   if max(real(eigen(:,i)))<0
%       plot([lambda(i),lambda(i+1)],[s(2,i),s(2,i+1)],'b');
%   else
%       plot([lambda(i),lambda(i+1)],[s(2,i),s(2,i+1)],'r');
%   end
% end
% disp('ENTER druecken!'); pause
%
% hold off



% Beispiel 6: Zweidimensionales Problem
% -------------------------------------
%
% L\"{o}sungszweig

% [lambda1,s1,eigen1]=xcont(@bsp6,[0.0 1.0],[0.0;0.0],contset
                     ('SolTol',1e-8,'MaxStepSize',.2, 'MinStepSize',
                     1e-5,  'MaxArcLength',100 ,'TanCon',.5e-1,
                     'OdeSol',@dopri87,'OdeOpt',odeset('RelTol',
                     1e-8,'AbsTol',1e-8), 'BifTol',1e-4,'BifOdeSol',
                     @dopri87,  'BifOdeOpt'   ,odeset('RelTol',1e-10,
                     'AbsTol',1e-10),   'Cor'   ,'nl2', 'MaxNewSteps' ,
                        10,    'MaxRegSteps'   ,10, 'SavePath',''));
% plot(lambda1,s1(1,:)), title('L\"{o}sungsfortsetzung:
  N\"{a}herungsl\"{o}sung (blau)')
% [lambda2,s2,eigen2]=xcont(@bsp6,[0.0 1.0],[1.0;0.0],contset
                     ('SolTol',1e-8,'MaxStepSize',.2,'MinStepSize',
                     1e-5,  'MaxArcLength',100,'TanCon,.5e-1,
                     'OdeSol'   ,@dopri87,  'OdeOpt'   ,odeset
                     ('RelTol',1e-8,'AbsTol',1e-8), 'BifTol',1e-4,
                     'BifOdeSol', @dopri87, 'BifOdeOpt',odeset
                     ('RelTol',1e-10,'AbsTol',1e-10),'Cor' ,'nl2',
                     'MaxNewSteps',10,'MaxRegSteps',10,'SavePath','']));
```

```
% plot(lambda2,s2(1,:)), title('L\"{o}sungsfortsetzung:
  N\"{a}herungsl\"{o}sung (blau)')
% [lambda3,s3,eigen3]=xcont(@bsp6,[0.0 1.0],[0.3333;2],contset
                     ('SolTol',1e-8,'MaxStepSize',.2,'MinStepSize',
                     1e-5,  'MaxArcLength',100,'TanCon,.5e-1,
                     'OdeSol' ,@dopri87,'OdeOpt',odeset('RelTol',
                     1e-8,'AbsTol',1e-8),'BifTol' ,1e-4,'BifOdeSol',
                     @dopri87,  'BifOdeOpt'   ,odeset('RelTol',1e-10,
                     'AbsTol',1e-10),'Cor','nl2', 'MaxNewSteps',10,
                     'MaxRegSteps' ,10, 'SavePath',''));
% plot(lambda3,s3(1,:)), title('L\"{o}sungsfortsetzung:
  N\"{a}herungsl\"{o}sung (blau)')
% grid on
% disp('ENTER druecken!'); pause
%
%
%
% figure(2);
%     h=gca;
%     hold on
%     grid on
%     xlabel('\lambda');
%     title('Stabile L\"{o}sungen (blau) und
      instabile L\"{o}sungen (rot)');
% for i = 1:length(lambda1)-1
%   if max(real(eigen1(:,i)))<0
%       plot([lambda1(i),lambda1(i+1)],[s1(1,i),s1(1,i+1)],'b');
%   else
%       plot([lambda1(i),lambda1(i+1)],[s1(1,i),s1(1,i+1)],'r');
%   end
% end
% for i = 1:length(lambda2)-1
%   if max(real(eigen2(:,i)))<0
%       plot([lambda2(i),lambda2(i+1)],[s2(1,i),s2(1,i+1)],'b');
%   else
%       plot([lambda2(i),lambda2(i+1)],[s2(1,i),s2(1,i+1)],'r');
%   end
% end
```

```
% for i = 1:length(lambda3)-1
%   if max(real(eigen3(:,i)))<0
%       plot([lambda3(i),lambda3(i+1)],[s3(1,i),s3(1,i+1)],'b');
%   else
%       plot([lambda3(i),lambda3(i+1)],[s3(1,i),s3(1,i+1)],'r');
%   end
% end
% disp('ENTER druecken!'); pause
%
% hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Subharmonisch erregtes Netzwerk
% -------------------------------
%
% L\"{o}sungszweig 1 (2/3*Pi-periodisch)
%[lambda,s,charmult,bif]=cont(@subhar,2/3*pi,[-6 3.2  6],[1;0.7],
                          contset( Cor','nl2','MaxStepSize',.2,
                          'BifTol',1e-6,'SavePath','subhar1'));
%
% L\"{o}sungszweig 2 (2/3*Pi-periodisch)
%[lambda,s,charmult,bif]=cont(@subhar,2/3*pi,[-6 6],[1;0.7],contset
                         ('Cor','nl2','MaxStepSize',.2,'BifTol',1e-6,
                         'SavePath','subhar2'));
%
% L\"{o}sungszweig 3 (2Pi-periodisch)
%[lambda,s,charmult,bif]=cont(@subhar,2*pi,[-6 1 6],[1;0.7],contset
                          ('Cor','nl1','MaxStepSize',.1,'BifTol',1e-6,
                          'SavePath','subhar3'));



% Phase Converter
% ---------------
%
% L\"{o}sungszweig 1
% [lambda,s,charmult,bif]=cont(@phacon,2*pi,[-10 3.4 10],
  [ -3.786253252083587e-001
```

```
%   -1.366339590320399e+000
%    1.516201296674700e+000
%   -2.468384564780271e+000],contset('MaxStepSize',.3,
    'BifTol',1e-6,'Cor',
    'nl2','SavePath','phacon1'));
%
% L\"{o}sungszweig 2
% [lambda,s,charmult,bif]=cont(@phacon,2*pi,[-50 1 50],
  [-1.172664246539392e-001
%     1.967270815321289e+000
%     2.212760808503210e+000
%     1.654188957192403e-001],contset('MaxArcLength',
      80,'BifTol',1e-6,'MaxStepSize',
      .3,'Cor','nl2','SavePath','phacon2'));
%
% L\"{o}sungszweig 3
% [lambda,s,charmult,bif]=cont(@phacon,2*pi,[25 19.8 0],
  [-3.956644731638856e-026
%    -7.815136323735343e-026
%     4.852514116655537e+000
%    -3.596100638533391e+000],contset('MaxArcLength',80,
     'BifTol',1e-6,'MaxStepSize',
     .3,'Cor','nl2','SavePath','phacon3'));
%
% L\"{o}sungszweig 4
% [lambda,s,charmult,bif]=cont(@phacon,2*pi,[0 12.5 25],
  [-2.249782419622298e-001
%     2
%     3.794301093835468e+000
%     1.055736724087728e+000],contset('MaxStepSize',.3,
      'BifTol',1e-6,'Cor','nl2',
      'SavePath','phacon4'));
%
% L\"{o}sungszweig 5
% [lambda,s,charmult,bif]=cont(@phacon,2*pi,[25 17 0],
  [ 1.631282535497187e-030
%     1.732209027593825e-029
%     4.570540319951759e+000
```

```
%     1.187101951148982e+000],contset('MaxArcLength',80,
      'BifTol',1e-6,'MaxStepSize',
      .3,'Cor','nl2','SavePath','phacon5'));
%
% L\"{o}sungszweig 6
% [lambda,s,charmult,bif]=cont(@phacon,2*pi,[0 19.8 25],
  [-3.956644731638856e-026
%    -7.815136323735343e-026
%     4.852514116655537e+000
%    -3.596100638533391e+000],contset('MaxArcLength',100,
     'BifTol',1e-6,'MaxStepSize',
     .3,'Cor','nl2','SavePath','phacon6'));




% Drei-Phasen-System mit Nulleiter !Lange Rechenzeiten,
  ggf.max.Bogenl\"{a}nge/Parameterintervalle heruntersetzen!
% --------------------------------

% L\"{o}sungszweig 1
% [lambda,s,charmult,bif]=cont(@dreimit,2*pi,[0.1 100],
                         [0;0;0;0;0;0;0;0;0;0],contset
                         ('MaxArcLength',200,'MaxStepSize',
                         .5,'BifTol',1e-6));

% L\"{o}sungszweig 2
% [lambda,s,charmult,bif]=cont(@dreimit,2*pi,
                          [0.1 3.812963516588616e+001 100],
                          [-1.501459550331506e+000
%     1.061595946929979e+000
%     1.443006019837471e+000
%    -2.201439814164836e+000
%     2.316126872555085e+000
%    -3.378858136086848e+000
%     2.767296604822004e+000
%     2.990462914831721e-001
%     1.446618276404159e-001],contset('MaxArcLength',100,
```

```
      'MaxStepSize',0.1,
      'BifTol',1e-6));

% L\"{o}sungszweig 3
% [lambda,s,charmult,bif]=cont(@dreimit,2*pi,
                         [0.1 4.650000000000000e+001 100],
                         [-1.158404269535132e+000
%     1.391691538625611e+000
%     1.493252230460173e+000
%    -1.450600618594241e+001
%     2.495977436849514e+000
%     3.062331644406575e+000
%    -7.982031274835327e-001
%     1.081456285504747e+000
%     1.833639006696231e+000],contset('MaxArcLength',
      100,'BifTol',1e-6,
      'MaxStepSize',.5));




% Graphische Ausgabe
% ------------------
%
% Zuvor mu{\ss} der entsprechende L\"{o}sungszweig [lambda,s,
(charmult,bif)] durch Aufruf von cont ermittelt oder
% durch >>load dateiname  geladen werden. Schon berechnete
L\"{o}sungszweige sind in
% subhar1,...,subhar3,phacon1,...,phacon6,dreimit1,...,dreimit3(.mat)
% zu finden.
%
%
% Mittels des 5. Eingabeparameters bestimmt man die Indizes von lambda,
  an denen die zugeh\"{o}rige
% periodische L\"{o}sung zeitparametrisiert geplottet werden soll.
%
% Mittels der 6. Eingabegr\"{o}{\ss}e bestimmt man die zu plottenden
  Komponenten des Anfangswertes bez\"{u}glich lambda.
```

```
% Z.B. erzeugt [[1,2];[7,4]] zwei Diagramme, eines im lambda-s1-s2-Raum
  und eines im lambda-s7-s4-Raum.
% Durch geeignetes Drehen der Grafiken erh\"{a}lt man auch 2D-Diagramme.
%
%
% ohne Stabilit\"{a}t und Bifurkation
% contplot(@subhar,2/3*pi,lambda,s,[],[1 2])
% contplot(@subhar,2*pi,lambda,s,[],[1 2])
% contplot(@phacon,2*pi,lambda,s,[],[[1 2];[2 3];[4 3]])
% contplot(@dreimit,2*pi,lambda,s,[],[[1 2];[5 6]])
%
% mit Stabilit\"{a}t ohne Bifurkation
% contplot(@subhar,2/3*pi,lambda,s,[],[1 2],charmult)
% contplot(@subhar,2*pi,lambda,s,[],[1 2],charmult)
% contplot(@phacon,2*pi,lambda,s,[],[[1 2];[2 3];[4 3]],charmult)
% contplot(@dreimit,2*pi,lambda,s,[],[[1 2];[5 6]],charmult)
%
% mit Stabilit\"{a}t und Bifurkation
% contplot(@subhar,2/3*pi,lambda,s,[],[1 2],charmult,bif)
% contplot(@subhar,2*pi,lambda,s,[1:20:length(lambda)],[1 2],charmult,bif)
% contplot(@phacon,2*pi,lambda,s,[],[[1 2];[2 3];[4 3]],charmult,bif)
% contplot(@dreimit,2*pi,lambda,s,[],[[1 2];[5 6]],charmult,bif)
```

A.5 bsp1

```
function f=bsp1(x,lambda)
global p
f = zeros(1,1);
f=-(40*p.*x.^5-200*p.*x.^4+(342*p+8).*x.^3-
  (226*p+24).*x.^2+0.5*(91*p+37).*x-lambda);
```

A.6 bsp2

```
function f=bsp2(x,lambda)
f = zeros(1,1);
f = -(x^3-x-lambda);
```

A.7 bsp3

```
function f=bsp3(x,lambda)
global u
x1 = x(1);
x2 = x(2);
x3 = x(3);
x4 = x(4);
x5 = x(5);
f = zeros(5,1);
f=[-3.933*x1+0.107*x2+0.126*x3-9.99*x5-45.83*lambda-
   0.727*x2*x3+8.39*x3*x4-684.4*x4*x5+63.5*x4*lambda;
   -0.987*x2-22.95*x4-28.37*u+0.949*x1*x3+0.173*x1*x5;
   0.002*x1-0.235*x3+5.67*x5-0.921*lambda-0.713*x1*x2
   -1.578*x1*x4+1.132*x4*lambda;
   x2-x4-0.168*u-x1*x5;
   -x3-0.196*x5-0.0071*lambda+x1*x4];
```

A.8 bsp4

```
function f=bsp5(x,lambda)
A = 3.03;
B = 10;
```

```
d = lambda;
D = 2*d;
x1= x(1);
x2 = x(2);
x3 = x(3);
x4 = x(4);
f = zeros(4,1);
f =[A-(B+1)*x1+x1^2*x3+d*(x2-x1);
    B*x1-x1^2*x3+D*(x4-x3);
    A-(B+1)*x2+x2^2*x4-d*(x2-x1);
    B*x2-x2^2*x4-D*(x4-x3)];
```

A.9 bsp5

```
function f=bsp2(x,lambda)
x1 = x(1);
x2 = x(2);
x3 = x(3);
x4 =x(4);
f = zeros(4,1);
f = [lambda*(1-x3)*exp(10*x1/(1+0.01*x1))-x3;
     22*lambda*(1-x3)*exp(10*x1/(1+0.01*x1))-30*x1;
     x3-x4+lambda*(1-x4)*exp(10*x2/(1+0.01*x2));
     10*x1-30*x2+22*lambda*(1-x4)*exp(10*x2/(1+0.01*x2))];
```

A.10 bsp6

```
function f=bsp6(u,lambda)
u1 = u(1);
u2 = u(2);
f = zeros(2,1);
f =[3*u1*(1-u1)-u1*u2-lambda*(1-exp(-5*u1));
    -u2+3*u1*u2];
```

Literatur

[1] Hoffmann, A; Marx, B; Vogt, W.: Mathematik für Ingenieure 1, Band 1, Pearson Studium, München 2005.

[2] D.V.Ansov, V.I.Arnold (Eds.): Dynamical System 1, in: Ordinary Differential Equations and Smooth Dynamical Systems, Springer-Verlag Berlin 1998.

[3] Nam Parshad Bhatia, Giorgio P.Szego: Stability Theory of Dynamical Systems, Springer-Verlag Berlin 2000.

[4] Stoer, Josef; Bulirsch Roland: Numerische Mathematik 2, in: Eine Einführung- und Berücksichtigung von Vorlesungen von F.L.Bauer, 5.Aufl., Berlin 2005.

[5] Vogt, Werner:(2007) Vorlesung von numerik dynamischer Systeme

[6] Deuflhard, P.: Newton Methods for Nnlinear Problems,Springer-Verlag Berlin 2004.

[7] Seydel, R.: Practical Bifurcation and Stability Analysis, Springer-Verlag New York 1994.

[8] Allgower, E.L.;Georg, K.: Numerical Continuation Methods, Springer-Verlag Berlin 1990.

[9] Argyris, J.; Faust, G.; Haase, M.: Die Erforschung des Chaos, Vieweg-Verlag Brauschweig 1995.

[10] Lynch, S.: Dynamical Systems with Applications using Matlab, Birkhäuser-Verlag Boston 2004.

[11] Peterseim, D.: Numerische Analyse parameterabhängiger periodischer Orbits nichtlinearer dynamischer Systeme mittels Mehrzielmethode und effizienter Fortsetzungstechniken, Diplomarbeit, TU Ilmenau 2004.

[12] Reitmann, V.: Reguläre und chaotische Dynamik, B.G.Teubner Verlagsgesellschaft Stuttgart. Leipzig 1996.

[13] Perko, L.: Differential Equations and Dynamical Systems, Springer-Verlag, 1991.

[14] Parker, T.S. und L.O. Chua: Practical Numerical Algorithmus for Chaotic Systems, Springer-Verlag, 1989.