Zur rechnergestützten Approximation nichtlinearer Systeme durch Polynomsysteme

Michael Spielmann und Mohieddine Jelali

Forschungsbericht Nr. 20/96

Meß-, Steuer- und Regelungstechnik

Übersicht: Während die Linearisierung von nichtlinearen Differentialgleichungen um einen Arbeitspunkt in der Regel keine Schwierigkeit bereitet und „von Hand“ durchgeführt werden kann, ist die Bestimmung höherer Terme der Taylorreihe problematisch. Da in der Literatur die Taylorreihenentwicklung von vektorwertigen Funktionen nur angedeutet wird, gibt dieser Bericht Hilfestellung bei der praktischen Bestimmung von Approximationen nichtlinearer Systeme durch Polynomsysteme beliebigen Grades. Die Implementierung erfolgt in Form von anwenderfreundlichen Funktionsaufrufen des symbolverarbeitenden Programmsystems MAPLE.

Gerhard-Mercator-Universität - GH Duisburg
Meß-, Steuer- und Regelungstechnik
Prof. Dr.-Ing. H. Schwarz

Inhaltsverzeichnis

Nomenklatur

Skalare Größen:

l	Approximationsgrad
m	Anzahl der Systemeingänge, Matrixdimension
n	Systemordnung, Matrixdimension
n_r	Dimension des reduzierten Kronecker–Produkts
p	Anzahl der Systemausgänge, Matrixdimension
q	Matrixdimension

Vektoren, Vektorfelder und Matrizen:

$\boldsymbol{A}, \boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{A}_k$	Systemmatrix
$\boldsymbol{B}, \boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{B}_k$	Eingangsmatrix
$\boldsymbol{P}_0$	Arbeitspunkt
$\boldsymbol{T}_k$, $\boldsymbol{T}_k^+$	Transformationsmatrix
$\boldsymbol{U}_0$	Arbeitspunkt
$\boldsymbol{X}_0$	Arbeitspunkt
$\boldsymbol{b}$	Eingangsvektor
$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$	Systemvektorfeld eines allgemein nichtlinearen Systems
$\boldsymbol{h}(\boldsymbol{x})$	Ausgangsvektorfeld eines allgemein nichtlinearen Systems
$\boldsymbol{u}(t)$,$\boldsymbol{U}(t)$	Eingangsvektor
$\boldsymbol{x}(t)$,$\boldsymbol{X}(t)$	Zustandsvektor
$\boldsymbol{y}(t)$,$\boldsymbol{Y}(t)$	Ausgangsvektor

Funktionen:

spur$\{\cdot\}$	Spur einer Matrix
$\otimes$, $(\cdot)^{(\cdot)}$	Kronecker–Produkt
$(\cdot)^{[\cdot]}$	Reduziertes Kronecker–Produkt

Abkürzungen:

ALS	Analytisches System mit linear eingehender Steuerung
BLS	Bilineares System
LS	Lineares System
MIMO	Mehrgößensystem (Multi-Input/Multi-Output)
PLS	Polynomsystem mit linear eingehender Steuerung
QLS	Quadratisches System mit linear eingehender Steuerung

1 Einführung

Die Systemtheorie im Bereich der Regelungstechnik hat die Systemanalyse und die Reglersynthese zum Inhalt. In beiden Aufgabenschwerpunkten steht die Modellbildung an erster Stelle, wobei zwischen Identifikation und theoretischer Modellbildung unterschieden werden kann. Wird ein Systemmodell aufgrund der physikalischen Gesetzmäßigkeiten gewonnen, entsteht in Abhängigkeit von dem betrachteten System ein komplexes nichtlineares System in Form von Differentialgleichungen, die wiederum in ein System von Differentialgleichungen erster Ordnung überführbar sind.

Für den Entwurf von Regelgesetzen werden Approximationen der nichtlinearen Gleichungen benötigt. Einen wichtigen Ansatz stellt dabei die Annäherung durch Taylorreihen dar, die je nach Ordnung der Approximation nach einer gewissen Anzahl von Summanden abgebrochen werden. Bei den bisher in der Regelungstechnik häufig verwendeten Systemklassen der linearen Systeme (LS) und bilinearen Systeme (BLS) bereitet das Ermitteln der Koeffizienten der Taylorreihe und das Darstellen in der gewohnten Form (vgl. Abschnitt 2.2) in der Regel keine Probleme. Bei der Approximation von nichtlinearen Systemen durch Polynomsysteme vom Grade $l \geq 2$ ist die Bestimmung der relevanten Matrizen dagegen mit sehr hohem Aufwand verbunden.

An dieser Stelle setzt der vorliegende Bericht an. Der folgende Abschnitt geht auf die zum Verständnis notwendigen theoretischen Grundlagen ein. Dazu gehört die Kenntnis des Kronecker–Produkts und die Notation mittels reduziertem Kronecker–Produkt auf. Dieser Abschnitt ist deshalb von großer Bedeutung, da in der Literatur zwar die Notation mittels reduziertem Kronecker–Produkt angedeutet wird, aber in der Regel aufgrund der dort fehlenden systematischen Rechenvorschrift nicht angewendet werden kann. Desweiteren erfolgt ein Überblick über die Linearisierung mittels Taylorreihenentwicklung.

Für die Bestimmung der Matrizen in den einzelnen Systemklassen, die in Abschnitt 2.3 vorgestellt werden, ist der Einsatz einer symbolverarbeitenden Programmiersprache (CAS) nützlich und bei Approximationen höheren Grades sogar erforderlich. Zu diesem Zweck wurden zwei Funktionen in MAPLE implementiert, deren Anwendung in Abschnitt 3 dokumentiert ist. Die Funktionen selbst stehen im Anhang A als MAPLE–Code für die eigene Verwendung zur Verfügung.

Abschnitt 4 gibt eine kurze Zusammenfassung und einen Ausblick auf offengebliebene Aufgabenstellungen, die Gegenstand zukünftiger Forschungsarbeiten sein können.

2 Theoretische Grundlagen

2.1 Kronecker–Produkt–Notationen

Dieser Abschnitt stellt die herkömmliche (redundante) Kronecker–Produkt–Notation vor und führt eine reduzierte (nichtredundante) Schreibweise ein.

Definition 2.1 Kronecker–Produkt (Lancaster und Tismenetsky 1985)

a) Unter dem Kronecker–Produkt $\boldsymbol{A} \otimes \boldsymbol{B}$ zweier Matrizen $\boldsymbol{A} = (a_{ij})$ und $\boldsymbol{B} = (b_{kl})$ der Dimension $m \times n$ bzw. $p \times q$ wird die Matrix

$$\boldsymbol{A} \otimes \boldsymbol{B} := \begin{bmatrix} a_{11}\boldsymbol{B} & \dots & a_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & \dots & a_{mn}\boldsymbol{B} \end{bmatrix} \in \mathbb{R}^{mp \times nq} \tag{2.1}$$

verstanden.

b) Für zwei Vektoren $\boldsymbol{a} \in \mathbb{R}^m$ und $\boldsymbol{b} \in \mathbb{R}^n$ gilt:

$$\boldsymbol{a} \otimes \boldsymbol{b} = \left[a_1\boldsymbol{b}^{\mathrm{T}},\ a_2\boldsymbol{b}^{\mathrm{T}}, \dots ,\ a_m\boldsymbol{b}^{\mathrm{T}} \right]^{\mathrm{T}} \in \mathbb{R}^{mn} . \tag{2.2}$$

c) Das i–fache Kronecker–Produkt eines Vektors $\boldsymbol{x} \in \mathbb{R}^n$ mit sich selbst lautet:

$$\boldsymbol{x}^{(i)} = \underbrace{\boldsymbol{x} \otimes \boldsymbol{x} \otimes \dots \otimes \boldsymbol{x}}_{i\text{–mal}} . \tag{2.3}$$

□

Beispiel 2.1
Für

$$\boldsymbol{A} = \begin{pmatrix} 1 & -1 \\ 2 & 1 \end{pmatrix} , \qquad \boldsymbol{B} = \begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & -1 \\ 2 & 0 & 0 & 1 \end{pmatrix}$$

ergibt sich

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{pmatrix} \boldsymbol{B} & -\boldsymbol{B} \\ 2\boldsymbol{B} & \boldsymbol{B} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 2 & 0 & -1 & 0 & -2 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 \\ 2 & 0 & 0 & 1 & -2 & 0 & 0 & -1 \\ 2 & 0 & 4 & 0 & 1 & 0 & 2 & 0 \\ 0 & 2 & 0 & -2 & 0 & 1 & 0 & -1 \\ 4 & 0 & 0 & 2 & 2 & 0 & 0 & 1 \end{pmatrix} .$$

□

Für ausführliche Zusammenstellungen der Rechenregeln zum Kronecker–Produkt sei auf die Literaturstellen (Vetter 1973, Brewer 1978, Jelali 1994) verwiesen.

Ein Nachteil der herkömmlichen Kronecker–Produkt–Notation liegt allerdings in der Redundanz der Terme $x_i x_j,\ i \neq j$ aufgrund der Kommutativität der Multiplikation. Diese Redundanz führt beispielsweise zu Problemen bei der Systemidentifikation (Yin 1995). Die Koeffizienten der Terme $x_i x_j,\ i \neq j$ sind linear abhängig, so daß aus Ein–/Ausgangsmessungen diese Koeffizienten nicht unterscheidbar sind. Desweiteren wird bei der Implementation der in Abschnitt 3 vorgestellten Funktion zur Approximation durch Polynomsysteme auf Routinen zurückgegriffen, die keine redundanten Terme enthalten. Aus diesem Grund wird im folgenden das nichtredundante Kronecker–Produkt (reduziertes Kronecker–Produkt) eingeführt:

Definition 2.2 Reduziertes Kronecker–Produkt
Das reduzierte Kronecker–Produkt $\boldsymbol{x}^{[k]}$ wird durch die Rekursionsformel

$$\boldsymbol{x}^{[k]} = \begin{pmatrix} x_1 \boldsymbol{b}_1^{k-2} \\ x_2 \boldsymbol{b}_2^{k-2} \\ \vdots \\ x_n \boldsymbol{b}_n^{k-2} \end{pmatrix} \quad , \qquad k > 1 \tag{2.4a}$$

mit

$$\boldsymbol{b}_i^0 := \begin{pmatrix} x_i \\ x_{i+1} \\ \vdots \\ x_n \end{pmatrix} \quad , \qquad \boldsymbol{b}_i^k = \begin{pmatrix} x_i \boldsymbol{b}_i^{k-1} \\ x_{i+1} \boldsymbol{b}_{i+1}^{k-1} \\ \vdots \\ x_n \boldsymbol{b}_n^{k-1} \end{pmatrix} \quad , \qquad k > 0 \quad , \quad i = 1, 2, \ldots, n \tag{2.4b}$$

erklärt. □

$\boldsymbol{x}^{[k]}$ stellt also einen Vektor der Dimension

$$n_r = \binom{n+k-1}{k} = \frac{(n+k-1)!}{k!\,(n-1)!} \tag{2.5}$$

dar, die um genau $\Delta n = n^k - n_r$ geringer ist als die des Vektors $\boldsymbol{x}^{(k)}$, dem herkömmlichen Kronecker–Produkt. Weiterhin ist es möglich, die beiden Produkte durch lineare Transformationen

$$\boldsymbol{x}^{(k)} = \boldsymbol{T}_k \boldsymbol{x}^{[k]} \tag{2.6}$$

$$\boldsymbol{x}^{[k]} = \boldsymbol{T}_k^{+} \boldsymbol{x}^{(k)} \tag{2.7}$$

ineinander zu überführen. Hierbei berechnet sich die linke Pseudo–Inverse $\boldsymbol{T}_k^+$ gemäß

$$\boldsymbol{T}_k^+ = (\boldsymbol{T}_k^{\mathrm{T}}\boldsymbol{T}_k)^{-1}\boldsymbol{T}_k^{\mathrm{T}} \ . \tag{2.8}$$

Beispiel 2.2 : Für $k = 2$ und $n = 3$ ergibt sich

$$\boldsymbol{x}^{(2)} = \begin{pmatrix} x_1^2 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2^2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3^2 \end{pmatrix} \quad , \qquad \boldsymbol{T}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad , \qquad \boldsymbol{x}^{[2]} = \begin{pmatrix} x_1^2 \\ x_1x_2 \\ x_1x_3 \\ x_2^2 \\ x_2x_3 \\ x_3^2 \end{pmatrix}$$

und mit Gl. (2.8) auch

$$\boldsymbol{T}_2^+ = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \ .$$

□

Die Transformationsmatrizen $\boldsymbol{T}_k$ und $\boldsymbol{T}_k^+$ besitzen die Dimensionen $p \times q$ bzw. $q \times p$ mit

$$p = n^k \tag{2.9}$$

$$q = \binom{n+k-1}{k} \tag{2.10}$$

sowie folgende Eigenschaften (Jelali 1995):

i) $\boldsymbol{T}_k^+\boldsymbol{T}_k = \boldsymbol{I}_q$,

ii) $\boldsymbol{T}_k\boldsymbol{T}_k^+$ ist eine symmetrische $p \times p$–Matrix und besitzt den Rang q,

iii) $\boldsymbol{T}_k^{\mathrm{T}}\boldsymbol{T}_k$ ist eine $q \times q$–Diagonalmatrix und besitzt den Rang q und

iv) spur $\left\{\boldsymbol{T}_k^{\mathrm{T}}\boldsymbol{T}_k\right\} = p$.

2.2 Approximation nichtlinearer Systeme durch Taylorreihen

Ausgehend von einem System aus gekoppelten Differentialgleichungen erster Ordnung

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \boldsymbol{y}(t) &= \boldsymbol{h}(\boldsymbol{x}(t)) \end{aligned} \tag{2.11}$$

werden in der Literatur verschiedene Unterklassen von nichtlinearen Systemen definiert (Schwarz 1991, Jelali 1994). Ausgangspunkt ist die Entwicklung aller Terme in Form der Taylorreihe. Für skalarwertige Funktionen gilt:

$$f(x) = \sum_{v=0}^{\infty} \frac{f^{(v)}(x_0)}{v!}(x - x_0)^v \quad , \tag{2.12}$$

die für den Fall $x_0 = 0$ auch als *Mac Laurinsche Reihe* bezeichnet wird (Bronstein und Semendjajew 1987). Dabei wird vorausgesetzt, daß die Gleichungen hinreichend oft differenzierbar sind. Da die Terme in Gl. (2.11) in der Regel nicht nur eine Zustandsgröße enthalten, kommt die allgemeinere Form der Taylorreihe zum Einsatz. In (Duschek 1963) findet sich beispielsweise die Darstellung für zwei Variablen:

$$f(x,y) = \sum_{i,k} a_{ik}(x - x_0)^i\,(y - y_0)^k \tag{2.13}$$

mit

$$a_{ik} = \frac{1}{i!k!}\frac{\partial^{i+k} f}{\partial x^i\,\partial y^k}(x_0, y_0), \quad i,k = 0,1,2,\ldots \quad . \tag{2.14}$$

Diese, auch in anderen Literaturstellen zu findende Form enthält keine redundanten Terme, wie beispielsweise:

$$\frac{\partial^2 f}{\partial x \partial y} \quad \text{und} \quad \frac{\partial^2 f}{\partial y \partial x} \tag{2.15}$$

Bei den in Abschnitt 2.3 beschrieben Näherungen nichtlinearer Systeme treten diese aber aufgrund des Kronecker-Produktes auf. Folglich muß gemäß Gl. (2.6) eine Transformation vorgenommen werden.

Fleming (1976) gibt die Darstellung

$$\begin{aligned} f(\boldsymbol{x}) = f(\boldsymbol{x}_0) &+ \sum_{i_1=1}^{n} f_{\mathrm{i}_1}(\boldsymbol{x}_0)(x_{\mathrm{i}_1} - x_{\mathrm{i}_1,0}) + \frac{1}{2!}\sum_{i_1,i_2=1}^{n} f_{\mathrm{i}_1,\mathrm{i}_2}(\boldsymbol{x}_0)(x_{\mathrm{i}_1} - x_{\mathrm{i}_1,0})(x_{\mathrm{i}_2} - x_{\mathrm{i}_2,0}) \\ &+ \cdots + \frac{1}{l!}\sum_{i_1,\ldots,i_l=1}^{n} f_{\mathrm{i}_1,\ldots,\mathrm{i}_l}(\boldsymbol{x}_0)\prod_{j=1}^{l}(x_{\mathrm{i}_j} - x_{\mathrm{i}_j,0}) \quad . \end{aligned} \tag{2.16}$$

für Funktionen mit mehreren Veränderlichen an. Die Berechnung der Terme $f_{\mathrm{i_1,\dots,i_j}}$ läßt sich anhand eines Beispiels verdeutlichen:

$$f_{1,4,4,3} = \frac{\partial^4 f}{\partial x_1 \partial x_4 \partial x_4 \partial x_3} \quad . \tag{2.17}$$

Redundante Terme treten entsprechend ihrer Anzahl mehrfach in Gl. (2.16) auf. Daß die Anwendung für höhere Approximationsgrade nicht mehr „von Hand“ erfolgen kann, versteht sich von selbst.

Eine kompakte Schreibweise der Taylorreihenentwicklung als Erweiterung von Gl. (2.16) für ein Vektorfeld $\boldsymbol{F}(\boldsymbol{Z})$ ($\boldsymbol{F}, \boldsymbol{Z} \in \mathbb{R}^n$) um einen Punkt $\boldsymbol{Z}_0$ unter Beachtung des Fehlerterms findet sich in Vetter (1973):

$$\boldsymbol{F}(\boldsymbol{Z}) = \boldsymbol{F}(\boldsymbol{Z}_0) + \sum_{j=1}^{l} \frac{1}{j!} \frac{\partial^j \boldsymbol{F}(\boldsymbol{Z})}{\partial \boldsymbol{Z}^j}\bigg|_{\boldsymbol{Z}_0} (\boldsymbol{Z} - \boldsymbol{Z}_0)^{(j)} + \boldsymbol{R}_{l+1}(\boldsymbol{Z}, \boldsymbol{Z}_0) \tag{2.18}$$

mit dem Restgliedsummanden

$$\boldsymbol{R}_{l+1}(\boldsymbol{Z}, \boldsymbol{Z}_0) = \frac{1}{j!} \int_{\boldsymbol{Z}_0}^{\boldsymbol{Z}} \frac{\partial^{j+1} \boldsymbol{F}(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}^{j+1}} \left[\boldsymbol{I}_n \otimes (\boldsymbol{Z} - \boldsymbol{\xi})^{(j)} \right] \mathrm{d}\boldsymbol{\xi} \ , \tag{2.19}$$

der im folgenden weggelassen wird.

2.3 Herleitung von Näherungen für nichtlineare Systeme

Die mathematische Beschreibung technischer und natürlicher Prozesse über das Aufstellen geeigneter Bilanzgleichungen (z.B. Bilanzierungen für Kräfte, Momente, allgemeine Erhaltungssätze für Masse, Energie, Impuls, usw.) führt in der Regel zwangsläufig zu Zustandsmodellen der MIMO-(Multi–Input/Multi–Output)–Form

$$\dot{\boldsymbol{X}}(t) = \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U}) \; ; \quad \boldsymbol{X}_0 = \boldsymbol{X}(t_0) \; , \quad \boldsymbol{X}(t) \in \mathbb{R}^n \; , \quad \boldsymbol{U}(t) \in \mathbb{R}^m \tag{2.20a}$$

$$\boldsymbol{Y}(t) = \boldsymbol{C}\boldsymbol{X}(t) \; , \qquad \boldsymbol{Y}(t) \in \mathbb{R}^p \; . \tag{2.20b}$$

Vielfach ist dieses Modell so umfangreich und kompliziert, daß Vereinfachungen der exakten Zusammenhänge bzw. vereinfachende Annahmen für weitere Anwendungen getroffen werden müssen (Spielmann 1996). Insbesondere bei regelungstechnischen Aufgaben interessieren in erster Linie lokale Näherungen in der Umgebung eines Arbeitspunktes. Mit zunehmendem Anspruch an die Genauigkeit moderner Regelungssysteme kommt Systemmodellen, die einen größeren Gültigkeitsbereich besitzen, größere Bedeutung zu.

Betrachtet wird nun ein Arbeitspunkt $\boldsymbol{P}_0 = (\boldsymbol{X}_0, \boldsymbol{U}_0)$ der Zustandsvariablen und der Eingangsgrößen. Diese werden notiert zu:

$$\begin{aligned} \boldsymbol{x}(t) &= \boldsymbol{X}(t) - \boldsymbol{X}_0 \\ \boldsymbol{u}(t) &= \boldsymbol{U}(t) - \boldsymbol{U}_0 \; , \end{aligned}$$

wobei Großbuchstaben die über den gesamten Arbeitsbereich definierten Größen und Kleinbuchstaben die lokal gültigen kennzeichnen. Analog gilt auch

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U}) - \boldsymbol{F}(\boldsymbol{X}_0,\boldsymbol{U}_0) \; .$$

Mit Hilfe von Gl. (2.18) lassen sich explizit lineare, bilineare, quadratische, aber auch Polynom–Modelle aus der nichtlinearen Systembeschreibung von Gl. (2.20) wie folgt herleiten:

LS:

$$\dot{\boldsymbol{x}}(t) \; = \; \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \left.\frac{\partial \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U})}{\partial \boldsymbol{X}}\right|_{\boldsymbol{P}_0} \boldsymbol{x}(t) + \left.\frac{\partial \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U})}{\partial \boldsymbol{U}}\right|_{\boldsymbol{P}_0} \boldsymbol{u}(t) \tag{2.21a}$$

$$=: \boldsymbol{A}_1\boldsymbol{x}(t) + \boldsymbol{B}_0\boldsymbol{u}(t) \tag{2.21b}$$

$$\boldsymbol{y}(t) \; = \; \boldsymbol{C}\boldsymbol{x}(t) \; . \tag{2.21c}$$

BLS:

$$\dot{\boldsymbol{x}}(t) = \frac{\partial \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U})}{\partial \boldsymbol{X}}\Big|_{\boldsymbol{P}_0} \boldsymbol{x}(t) + \frac{\partial \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U})}{\partial \boldsymbol{U}}\Big|_{\boldsymbol{P}_0} \boldsymbol{u}(t) + \frac{\partial^2 \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U})}{\partial \boldsymbol{X}\partial \boldsymbol{U}}\Big|_{\boldsymbol{P}_0} \boldsymbol{x}(t) \otimes \boldsymbol{u}(t) \tag{2.22a}$$

$$=: \boldsymbol{A}_1 \boldsymbol{x}(t) + \boldsymbol{B}_0 \boldsymbol{u}(t) + \boldsymbol{B}_1 \boldsymbol{x}(t) \otimes \boldsymbol{u}(t) \tag{2.22b}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t)\ . \tag{2.22c}$$

QLS:

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= \frac{\partial \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U})}{\partial \boldsymbol{X}}\Big|_{\boldsymbol{P}_0} \boldsymbol{x}(t) + \frac{1}{2}\frac{\partial^2 \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U})}{\partial \boldsymbol{X}^2}\Big|_{\boldsymbol{P}_0} \boldsymbol{x}(t) \otimes \boldsymbol{x}(t) + \frac{\partial \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U})}{\partial \boldsymbol{U}}\Big|_{\boldsymbol{P}_0} \boldsymbol{u}(t) \\ &+ \frac{\partial^2 \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U})}{\partial \boldsymbol{X}\partial \boldsymbol{U}}\Big|_{\boldsymbol{P}_0} \boldsymbol{x}(t) \otimes \boldsymbol{u}(t) + \frac{1}{2}\frac{\partial^3 \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U})}{\partial \boldsymbol{X}^2\partial \boldsymbol{U}}\Big|_{\boldsymbol{P}_0} \boldsymbol{x}(t) \otimes \boldsymbol{x}(t) \otimes \boldsymbol{u}(t) \qquad (2.23a)\end{aligned}$$

$$\begin{aligned}&=: \boldsymbol{A}_1 \boldsymbol{x}(t) + \boldsymbol{A}_2 \boldsymbol{x}(t) \otimes \boldsymbol{x}(t) + \boldsymbol{B}_0 \boldsymbol{u}(t) + \boldsymbol{B}_1 \boldsymbol{x}(t) \otimes \boldsymbol{u}(t) \\ &+ \boldsymbol{B}_2 \boldsymbol{x}^{(2)}(t) \otimes \boldsymbol{u}(t) \qquad (2.23b)\end{aligned}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t)\ . \tag{2.23c}$$

PLS:

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= \frac{\partial \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U})}{\partial \boldsymbol{X}}\Big|_{\boldsymbol{P}_0} \boldsymbol{x}(t) + \sum_{j=2}^{r} \frac{1}{j!}\frac{\partial^j \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U})}{\partial \boldsymbol{X}^j}\Big|_{\boldsymbol{P}_0} \boldsymbol{x}^{(j)}(t) \\ &+ \frac{\partial \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U})}{\partial \boldsymbol{U}}\Big|_{\boldsymbol{P}_0} \boldsymbol{u}(t) + \sum_{j=2}^{r-1} \frac{1}{j!}\frac{\partial^{j+1} \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U})}{\partial \boldsymbol{X}^j\partial \boldsymbol{U}}\Big|_{\boldsymbol{P}_0} \boldsymbol{x}^{(j)}(t) \otimes \boldsymbol{u}(t) \qquad (2.24a)\end{aligned}$$

$$=: \boldsymbol{A}_1 \boldsymbol{x}(t) + \sum_{i=2}^{r} \boldsymbol{A}_i \boldsymbol{x}^{(i)}(t) + \boldsymbol{B}_0 \boldsymbol{u}(t) + \sum_{j=1}^{r} \boldsymbol{B}_j \boldsymbol{x}^{(j)}(t) \otimes \boldsymbol{u}(t) \tag{2.24b}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t)\ . \tag{2.24c}$$

Die zahlenmäßige Bestimmung der konstanten Matrizen $\boldsymbol{A}_i$ und $\boldsymbol{B}_j$ ist Gegenstand des Abschnitts 3.

3 Rechnergestützte Approximation

Wie in Abschnitt 2.2 gezeigt wurde, erfordert die Approximation nichtlinearer Systeme durch Polynomsysteme die Unterstützung eines Computer–Algebra–Systems (CAS). Insbesondere ist die Verwendung einer symbolverarbeitenden Software zur Bestimmung der konstanten Matrizen in den Systemgleichungen erforderlich. Da die im Fachgebiet Meß-, Steuer- und Regelungstechnik bereits entwickelten Tools in MAPLE implementiert wurden (Lemmen u. a. 1995) und von daher bereits Systeme in der klassischen ALS–Form

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{a}(\boldsymbol{x}(t)) + \boldsymbol{B}(\boldsymbol{x}(t))\,\boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{c}(\boldsymbol{x}(t)) \end{aligned} \tag{3.1}$$

als Dateien vorliegen, wurden die für die Approximation benötigten Funktionen ebenfalls für MAPLE geschrieben. Die Verwendung dieser Umgebung bietet zudem den Vorteil, daß bereits eine Routine zur Taylorreihenentwicklung für Funktionen mit mehreren Veränderlichen existiert. Die Aufgabe besteht im wesentlichen also darin, die Ausgabe in die übliche Systembeschreibung zu überführen und die Matrizenbelegung zu bestimmen.

Die Vorgehensweise bei der rechnergestützten Approximation wird in diesem Abschnitt anhand eines Beispiels verdeutlicht. Das nichtlineare System

$$\begin{aligned} \dot{\boldsymbol{x}} &= \begin{bmatrix} \mathrm{e}^{x_2} \\ x_1^2 \end{bmatrix} + \begin{bmatrix} \cos x_1 \\ 0 \end{bmatrix} u \\ y &= x_2 \end{aligned} \tag{3.2}$$

wird zur besseren Verfügbarkeit als Datei abgelegt:

```
# bsp1.als
#

n := 2;    # Systemordnung
m := 1;    # Anzahl der Eingaenge
p := 1;    # Anzahl der Ausgaenge

x := linalg[vector](n,[]);

a := linalg[vector](n,[ exp(x[2]),
                          x[1]^2
                      ]);
```

```
B := linalg[matrix](n,m,[
                        cos(x[1]),
                        0
                        ]);

c := linalg[vector]([ x[2] ]);
```

Bei der Übertragung auf andere Systeme ist folgendes zu beachten:

- Das System sollte immer als ALS mit den Vektoren $\boldsymbol{a}$, $\boldsymbol{c}$ und der Matrix $\boldsymbol{B}$ abgelegt werden.
- Bei der Zuweisung sind immer die von dem Paket `linalg` vorgegebenen Funktionen `matrix` und `vector` zu verwenden.
- Auch wenn, wie in dem Beispiel oben, keine „echten“ Vektoren oder Matrizen vorliegen (z. B. ist der Vektor $\boldsymbol{c}$ in `bsp1.als` eigentlich ein Skalar), sind `a`, `B` und `c` als solche zu definieren.

Unter der Annahme, daß die im Anhang A aufgeführten Funktionen in einer Datei namens `mytool` abgelegt sind und sich im gleichen Verzeichnis wie die Systemdatei `bsp1.als` befinden, zeigt die folgende MAPLE-Session die Approximation als Polynomsystem der Ordnung $l = 3$:

```
> with(linalg):
> readlib(coeftayl):
> read(‘bsp1.als‘):
> read(‘mytool‘):
> x0:=vector(2,[0,0]);
```

$$x0 := [0\ 0]$$

```
> u0:=vector(1,[0]);
```

$$u0 := [0]$$

```
> ant:=myPls([x,a,B,c],[x0,u0],3);
```

$$ant := A_1, A_2, A_3, B_0, B_1, B_2, B_3, C, TA_2, TA_3, TB_2, TB_3$$

```
> A3:=evalm(ant[3]);
```

$$\begin{bmatrix} 0 & 0 & 0 & \frac{1}{6} \\ & & & \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Die ersten Aufrufe `with(linalg)` und `readlib(coefftayl)` werden für die beiden Funktionen in `mytool` benötigt und können prinzipiell auch dort eingebunden werden. Bei der Festlegung des Arbeitspunktes, der auch allgemein angegeben werden kann, ist wiederum die Zuweisung als Vektor verbindlich.

Der Aufruf von `myPls(...)` liefert eine MAPLE-Liste (`ant`) deren Elemente nach einer Zuweisung durch

```
> A1:=evalm(ant[1]); A2:=evalm(ant[2]); A3:=evalm(ant[3]);
> B0:=evalm(ant[4]); B1:=evalm(ant[5]); B2:=evalm(ant[6]);
> B3:=evalm(ant[7]); TA2:=evalm(ant[8]); TA3:=evalm(ant[9]);
> TB2:=evalm(ant[10]); TB3:=evalm(ant[11]);
```

extrahiert werden können. Für das Beispiel ergibt sich die Systemdarstellung zu:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \mathtt{A}_1\boldsymbol{x} + \mathtt{A}_2\boldsymbol{x}^{[2]} + \mathtt{A}_3\boldsymbol{x}^{[3]} + \mathtt{B}_0\,u + \mathtt{B}_1\boldsymbol{x}\,u + \mathtt{B}_2\boldsymbol{x}^{[2]}\,u + \mathtt{B}_3\boldsymbol{x}^{[3]}\,u \\ y &= \mathtt{C}\boldsymbol{x} \qquad . \end{aligned} \tag{3.3}$$

Auf die Unterscheidung zwischen Vektoren und Matrizen durch Groß- und Kleinschreibung wird bei der Ausgabe verzichtet, was z. B. dazu führt, daß der Ausgangsvektor in Gl. (3.3) als Großbuchstabe erscheint.

Wie ebenfalls zu erkennen ist, werden die Matrizen für die Darstellung mit reduziertem Kronecker–Produkt bestimmt. Für die Umrechnung auf das „normale“ Kronecker–Produkt stehen eine Reihe von Transformationsmatrizen zur Verfügung ($\mathtt{TA}_2, \mathtt{TA}_3, \mathtt{TB}_2, \mathtt{TB}_3$). Dies geschieht durch Multiplizieren der reduzierten Matrizen mit der zugehörigen Transformationsmatrix von rechts.

```
> A3:=evalm(ant[3]);
```

$$A3 := \begin{bmatrix} 0 & 0 & 0 & \frac{1}{6} \\ & & & \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> TA3:=evalm(ant[10]);
```

$$TA3 := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

```
> multiply(A3,TA3);
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \frac{1}{6} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

4 Zusammenfassung und Ausblick

Die Approximation nichtlinearer Systeme durch einfachere Systeme ist für die Regelungstechnik notwendig, da mit der Berücksichtigung von Nichtlinearitäten höherer Ordnung die Komplexität des zu entwerfenden Regelgesetzes zunimmt.

Vorliegender Bericht gibt eine Hilfestellung bei der Bestimmung der zu den unterschiedlichen Approximationsordnungen gehörenden Matrizen. Mit Hilfe zweier für das symbolverarbeitende Programm MAPLE geschriebenen Routinen läßt sich diese Aufgabe problemlos bewältigen.

Offen ist noch die Ergänzung der im Fachgebiet MSRT entstandenen Tool–Box NSAS (Nonlinear System Analysis and Synthesis) um die in diesem Bericht vorgestellten Routinen.

5 Literaturverzeichnis

Brewer, J. W. 1978. Kronecker products and matrix calculus in system theory. *IEEE Transactions on Circuits and Systems* CAS–25. 772–781.

Bronstein, I. N. und **K. A. Semendjajew**. 1987. *Taschenbuch der Mathematik*. Leipzig: Teubner Verlagsgesellschaft.

Duschek, A. 1963. *Vorlesungen über höhere Mathematik. Zweiter Band*. Wien: Springer.

Fleming, W. 1976. *Functions of Several Variables*. New York: Springer.

Jelali, M. 1994. *Zur Modellierung nichtlinearer Prozesse durch quadratische Systeme mit linearer Steuerung (QLS)*. Forschungsbericht 5/94. MSRT. Universität Duisburg.

Jelali, M. 1995. *Systematischer Beobachterentwurf für nichtlineare Systeme*. Forschungsbericht 1/95. MSRT. Universität Duisburg.

Lancaster, P. und **M. Tismenetsky**. 1985. *The Theory of Matrices. Computer Science and Applied Mathematics*. Orlando: Academic Press.

Lemmen, M., **T. Wey** und **M. Jelali**. 1995. *NSAS – ein Computer-Algebra-Paket zur Analyse und Synthese nichtlinearer Systeme*. Forschungsbericht 20/95. MSRT. Universität Duisburg.

Schwarz, H. 1991. *Nichtlineare Regelungssysteme – Systemtheoretische Grundlagen*. München: Oldenbourg.

Spielmann, M. 1996. *Zustandsmodelle hydraulischer Antriebselemente*. Forschungsbericht 15/96. MSRT. Universität Duisburg.

Vetter, W. 1973. Matrix calculus operations and Taylor expansions. *SIAM Review* 15. 352–369.

Yin, X. 1995. *Zur Identifikation zeitkontinuierlicher nichtlinearer Systeme. VDI Fortschritt-Berichte. Reihe 8*. 385. Düsseldorf: VDI.

A MAPLE-Funktionen

A.1 myKronecker

```
#########################################################################
# myKronecker([x1],[x2])
#
# determine the kronecker product, the reduced kronecker product of to
# vectors x1, x2 and the transformation matrices
#
#########################################################################

myKronecker := proc(x1,x2)
  local x1dim, x2dim, \
        z1, z2, \
        vset1, vset2, Kron, RedKron,\
        T,Tplus:

  x1dim:=vectdim(x1):
  x2dim:=vectdim(x2):

  vset1:=[];

  for z1 from 1 to x1dim do
    for z2 from 1 to x2dim do
      vset1:=[vset1[],x1[z1]*x2[z2]];
    od:
  od:

  vset2:=[vset1[1]];
  for z1 from 2 to nops(vset1) do
    if not has(vset2,vset1[z1]) then
      vset2:=[vset2[],vset1[z1]];
    fi:
  od:

  T:=matrix(nops(vset1),nops(vset2),[]):

  for z1 from 1 to nops(vset1) do
    for z2 from 1 to nops(vset2) do
      if vset1[z1]=vset2[z2] then
```

```
        T[z1,z2]:=1
      else
        T[z1,z2]:=0
      fi:
    od:
  od:

  Tplus:=multiply(inverse(multiply(transpose(T),T)),transpose(T));

  Kron:=vector(nops(vset1),vset1);
  RedKron:=vector(nops(vset2),vset2);

  op([Kron,RedKron,T,Tplus]);

end:
```

A.2 myPls

```
#########################################################################
# myPls(system,OP,order)
#
# determine PLS from ALS
#
# system as [x,a,B,c]
# OP - Operating Point [xw,uw]
# order of approximation
#########################################################################
myPls := proc(system,OP,order)
  local xdim, mdim, pdim, bdim, u, \
        L0, L1, L2, f , vh, pos, \
        z1, z2, z3, z4,  \
        adim, A, Ac, B, Bc, C, \
        ab, Ah, TA, TB, kron:

  # get the matrix and vector dimensions of system

  xdim:=linalg[vectdim](system[1]):
  mdim:=linalg[coldim](system[3]):
  pdim:=linalg[vectdim](system[4]):

  # initialize some variables
```

```
u:=linalg[vector](mdim,[]):

L0:=[seq(xw[z1],z1=1..xdim),seq(uw[z1],z1=1..mdim)]:
L1:=[seq(x[z1],z1=1..xdim),seq(u[z1],z1=1..mdim)]:

# build the general nonlinear system

f:=linalg[add](system[2],(system[3] &* u)):

# get the dimension of a system matrix including all
# system matrizes of the polynomial system

adim:=0:
for z1 from 1 to order do
  adim:=adim+binomial(xdim+z1-1,z1):
od:

# create a help matrix Ac including the necessary derivatives

Ac:=linalg[matrix](xdim,adim,[]):

pos:=0:

for z1 from 1 to order do
  if (z1=1) then
    vh:=system[1]:
    TA[1]:=diag(seq(z2-z2+1,z2=1..xdim)):
  else
    kron:=myKronecker(system[1],vh):
    vh:=kron[2]:
    TA[z1]:=evalm(kron[4]):
  fi:
  for z2 from 1 to xdim do
    for z3 from 1 to vectdim(vh) do
      Ac[z2,pos+z3]:=vh[z3]
    od
  od:
  pos:=pos+vectdim(vh):
od:
```

```
# get the dimension of an input matrix including all
# input matrizes of the polynomial system

bdim:=(adim+1)*mdim:

# create a help matrix Bc including the necessary derivatives

Bc:=linalg[matrix](xdim,bdim,[]):

pos:=0:

for z1 from 0 to order do
  if (z1=0) then
    vh:=u:
    TB[0]:=diag(seq(z2-z2+1,z2=1..mdim)):
  else
    kron:=myKronecker(system[1],vh):
    vh:=kron[2]:
    TB[z1]:=evalm(kron[4]):
  fi:
  for z2 from 1 to xdim do
    for z3 from 1 to vectdim(vh) do
      Bc[z2,pos+z3]:=vh[z3]
    od
  od:
  pos:=pos+vectdim(vh)
od:

# linearization using the Maple-function coeftayl
# (system-matrix)

for z1 from 1 to xdim do
  for z2 from 1 to adim do
    for z3 from 1 to xdim do
       ab[z3]:=0:
       Ah:=Ac[z1,z2]:
       for z4 from 1 to order do
         Ah:=diff(Ah,x[z3]):
         if Ah <> 0 then
           ab[z3]:=ab[z3]+1:
         fi:
```

```
      od:
    od:
    L2:=[seq(ab[z3],z3=1..xdim),seq(z3-z3,z3=1..mdim)]:
    Ac[z1,z2]:=coeftayl(f[z1],L1=L0,L2):
  od:
od:

# linearization using the Maple-function coeftayl
# (input-matrix)

for z1 from 1 to xdim do
  for z2 from 1 to bdim do
    for z3 from 1 to xdim do
      ab[z3]:=0:
      Ah:=Bc[z1,z2]:
      for z4 from 1 to order do
        Ah:=diff(Ah,x[z3]):
        if Ah <> 0 then
          ab[z3]:=ab[z3]+1:
        fi:
      od:
    od:
    for z3 from 1 to mdim do
      ab[z3+xdim]:=0:
      Ah:=Bc[z1,z2]:
      for z4 from 1 to order do
        Ah:=diff(Ah,u[z3]):
        if Ah <> 0 then
          ab[z3+xdim]:=ab[z3+xdim]+1:
        fi:
      od:
    od:
    L2:=[seq(ab[z3],z3=1..xdim+mdim)]:
    Bc[z1,z2]:=coeftayl(f[z1],L1=L0,L2):
  od:
od:

# replace the operating point
# in the general system matrix

for z1 from 1 to xdim do
```

```
  for z2 from 1 to adim do
    for z3 from 1to xdim do
      Ac[z1,z2]:=eval(subs(xw[z3]=OP[1][z3],Ac[z1,z2])):
    od:
  od:
od:

for z1 from 1 to xdim do
  for z2 from 1 to adim do
    for z3 from 1to mdim do
      Ac[z1,z2]:=eval(subs(uw[z3]=OP[2][z3],Ac[z1,z2])):
    od:
  od:
od:

# replace the operating point
# in the general input matrix

for z1 from 1 to xdim do
  for z2 from 1 to bdim do
    for z3 from 1to xdim do
      Bc[z1,z2]:=eval(subs(xw[z3]=OP[1][z3],Bc[z1,z2])):
    od:
  od:
od:

for z1 from 1 to xdim do
  for z2 from 1 to bdim do
    for z3 from 1to mdim do
      Bc[z1,z2]:=eval(subs(uw[z3]=OP[2][z3],Bc[z1,z2])):
    od:
  od:
od:

# linearization of the output

C:=linalg[matrix](pdim,xdim,[]):
for z1 from 1 to pdim do
  for z2 from 1 to xdim do
    C[z1,z2]:=diff(system[4][z1],x[z2]):
  od:
```

```
  od:

  # separate matrices from the general matrices

  pos:=1:

  for z1 from 1 to order do
    z2:=binomial(xdim+z1-1,z1):
    A[z1]:=submatrix(Ac,1..xdim,pos..pos+z2-1):
    pos:=pos+coldim(A[z1]):
  od:

  pos:=2:
  B[0]:=submatrix(Bc,1..xdim,1..1):

  for z1 from 1 to order do
    z2:=mdim*binomial(xdim+z1-1,z1):
    B[z1]:=submatrix(Bc,1..xdim,pos..pos+z2-1):
    pos:=pos+coldim(B[z1]):
  od:

  # return results

  op([seq(A[z1],z1=1..order),
     seq(B[z1],z1=0..order),
     C,
     seq(TA[z1],z1=2..order),
     seq(TB[z1],z1=2..order)]);
end:
```