Diplomarbeit

Automatisches Differenzieren mit Anwendung in der Optimierung bei chemischen Reaktionssystemen

Gerd Rücker

Betreuer: Prof. Dr. H. G. Bock

Dezember 1999

Ruprecht-Karls-Universität Heidelberg
Fakultät für Mathematik

Interdisziplinäres Zentrum
für Wissenschaftliches Rechnen (IWR)

Erklärung

Hiermit erkläre ich, daß ich diese Arbeit selbständig verfaßt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und alle Stellen, die dem Wortlaut oder Sinne nach anderen Werken entnommen sind, durch Angabe der Quellen als Entlehnungen kenntlich gemacht habe.

Heidelberg, im Dezember 1999

Zusammenfassung

Die effiziente und genaue Berechnung von Ableitungen erster und höherer Ordnung ist eine wichtige Teilaufgabe bei der Behandlung nichtlinearer Probleme. Neben der numerischen Differentiation mit Hilfe von Differenzenquotienten wurden in den letzten Jahren vermehrt Methoden des Automatischen Differenzierens (AD) zur Erzeugung von Ableitungen verwendet.

In dieser Diplomarbeit werden die Prinzipien des AD auf die konkrete Problemklasse der Differentiell-Algebraischen Gleichungssysteme (DAE) angewendet, welche beispielsweise den Reaktionsverlauf chemischer Reaktionssysteme beschreiben. Diese Gleichungen beinhalten neben Zustandsvariablen und Steuergrößen auch unbekannte reaktionsspezifische Parameter, zu deren Bestimmung zunächst Versuche entworfen werden, deren Auswertung Meßdaten liefert, die bei der Parameterschätzung genutzt werden.

Zu Beginn der Arbeit werden grundlegende Aspekte aus der chemischen Reaktionskinetik und der nichtlinearen Optimierung aufgeführt. Anschließend wird die Theorie des Automatischen Differenzierens genauer betrachtet. Im praktischen Teil wird ein Modellgenerator für chemische Reaktionssysteme entwickelt, und in der objektorientierten Programmiersprache Java implementiert. Die Spezifikation des chemischen Reaktionssystems (instationär und homogen) mit verschiedenen Möglichkeiten der Wärmebilanzmodellierung erfolgt dabei über eine graphische Benutzeroberfläche. Aus der chemischen Beschreibung des Reaktionsprozesses werden sowohl die Modellgleichungen, die das DAE-System beschreiben, als auch die für die Optimierung notwendigen ersten und zweiten Ableitungen der Modellgleichungen automatisch in Form von Fortran- oder C-Dateien bereitgestellt. Die Dateien werden unter Verwendung der Prinzipien des AD in einer Modifikation des Vorwärtsmodus automatisch und strukturausnutzend erzeugt. Durch die gemeinsame Generierung von Modellgleichungen und dazugehörigen Ableitungen hat man Zugriff auf alle Informationen über das Modell, und die benötigten Ableitungsroutinen können effizient generiert werden. Die wiederholte und somit redundante Berechnung gleicher Teilausdrücke kann so verhindert werden. In einem Sparse-Modus wird zudem die Dünnbesetztheit von Richtungsmatrizen bei der Ableitungserzeugung berücksichtigt.

Die numerischen Tests anhand ausgewählter Reaktionssysteme aus der chemischen Industrie am Ende der Arbeit zeigen eine erhebliche Laufzeitverbesserung bei der Berechnung der Ableitungen sowohl gegenüber der Verwendung von Differenzenquotienten als auch beim Einsatz des Software-Pakets ADIFOR.

Inhaltsverzeichnis

Einleitung

Die Bedeutung der Angewandten Mathematik für die Naturwissenschaften, die Technik und die Wirtschaft ist in den letzten Jahrzehnten aufgrund immer leistungsfähigerer Rechner weiter gestiegen. Reale Situationen und Vorgänge lassen sich immer besser durch mathematische Modelle abbilden, die auf Computern simuliert werden können. Diese Modelle ergänzen Experimente und ermöglichen es, Prozesse zu behandeln, deren Durchführung sehr schwierig oder sehr kostenintensiv ist. Mit solchen und ähnlichen Fragestellungen beschäftigt sich das Wissenschaftliche Rechnen (*Scientific Computing*), eine interdisziplinäre Kombination aus Mathematik, Informatik und Anwendungswissenschaften wie z.B. Physik, Chemie, Medizin und Wirtschaftswissenschaften.

Problemstellungen in der Industrie führen dabei sehr häufig auf Modelle, welche die numerische Lösung von Systemen nichtlinearer Differentialgleichungen erfordern. Beispiele hierfür sind chemische Reaktionen in einem Reaktor, Transportprozesse von Chemikalien in Böden, die Produktion in verfahrenstechnischen Anlagen oder die Verbrennung in einem Motor. Jedes dieser Modelle enthält dabei in der Regel Systemparameter, beispielsweise Reaktionsgeschwindigkeiten, Permeabilitäten oder mechanische Konstanten, deren Bestimmung zentral für den konkreten Einsatz der Modelle ist.

In dieser Arbeit werden chemische Reaktionssysteme in einem instationären homogenen Rührkessel durch Differentiell-Algebraische Gleichungssysteme mathematisch beschrieben. Diese Gleichungen enthalten als Zustandsvariablen beispielsweise die zeitlich variablen Stoffmengen und die Reaktortemperatur sowie Steuergrößen (z.B. Anfangsstoffmengen und Anfangstemperaturen). Die Gleichungen enthalten auch unbekannte reaktionsspezifische Parameter, die mit Hilfe von Parameterschätzung aus experimentellen Daten bestimmt werden. Da die Durchführung von Experimenten zur Erhebung von Meßdaten oft sehr teuer ist, wird in der optimalen Versuchsplanung ein möglichst signifikanter und kostengünstiger Meßentwurf erstellt, aus dessen Messungen die unbekannten Parameter genau bestimmt werden können. Sind die Modelle hinreichend validiert, können Prozeßszenarien simuliert oder optimale Fahrweisen berechnet werden.

Versuchsplanungsprobleme sind komplexe optimale Steuerungsprobleme, deren Zielfunktion bereits von den Sensitivitäten der Trajektorien des zugrundeliegenden DAE-Modells nach den Modellparametern abhängt. In einem gradientenbasierten Optimierungsverfahren werden daher auch zweite Ableitungen der Lösungen des DAE-Systems benötigt. Für die Leistungsfähigkeit des Optimierungsverfahrens wird eine hohe Genauigkeit der berechneten Ableitungen gefordert. Die Berechnung der Lösungs- und Ableitungstrajektorien des DAE-Systems mit einem impliziten Integrationsverfahren erfordert erste und zweite Richtungsableitungen der Modellfunktionen, die das Reaktionssystem und die verwendeten Meßverfahren modellieren. Diese Ableitungen wurden bisher in der am IWR implementierten Versuchsplanungssoftware VPLAN98 (Körkel (1999)) mit dem Software-Paket ADIFOR (Bischof et al. (1995)) erzeugt. Profiling-Messungen ergaben dabei, daß ein großer Anteil der Laufzeit des Programms durch die erzeugten Ableitungen verbraucht wird.

Ziel der vorliegenden Diplomarbeit ist es, diese Ableitungen vollautomatisch, mit hoher Genauigkeit und so effizient wie möglich zu erzeugen sowie bei der Simulation und Optimierung praxisnaher Beispiele einzusetzen. Die dabei untersuchten Beispiele wurden in einer Kooperation des IWR mit dem Chemie-Unternehmen BASF AG (Ludwigshafen) im

Rahmen eines BMBF[1]-Projektes modelliert und untersucht (BAUER ET AL. (1998b)).

Im Gegensatz zu herkömmlichen Software-Paketen, wie z.B. ADIFOR (BISCHOF ET AL. (1992)), bei denen das abzuleitende Programm als Black-Box betrachtet wird und ein Parsen des Quelltextes unabdingbar ist, wird in dieser Arbeit ein anderer Ansatz gewählt: Die benötigten Ableitungsroutinen werden gleichzeitig mit denjenigen Routinen erzeugt, die das DAE-System beschreiben. Um dies zu ermöglichen, wurde ein Modellgenerator entwickelt, der aus einer chemischen und verfahrenstechnischen Beschreibung eines Prozesses die Modellgleichungen automatisch generiert. Durch die gleichzeitige Erzeugung von Modellgleichungen und dazugehörigen Ableitungen können alle Strukturen zur effizienten Generierung von Ableitungen ausgenutzt werden.

Inhaltsübersicht

In Kapitel 1 werden die mathematischen, chemischen und physikalischen Sachverhalte kurz erläutert, die für das Verständnis der weiteren Kapitel benötigt werden. Zunächst werden in Abschnitt 1.1 Differentiell-Algebraische Gleichungssysteme definiert und Ableitungsregeln für die Differentiation von Matrizen nach Matrizen aufgestellt. Im darauffolgenden Abschnitt werden grundlegende Aspekte aus der Physikalischen Chemie sowie der chemischen Verfahrenstechnik betrachtet, die für die Erstellung der Modellgleichungen für reakionskinetische Prozesse von Bedeutung sind. In Abschnitt 1.3 werden Grundlagen der nichtlinearen Versuchsplanung und die Spezifikation des in dieser Arbeit betrachteten Problems angeführt. Nach einer Einführung in das betrachtete nichtlineare Optimierungsproblem werden die beiden Teilprobleme, ein Versuchsplanungsproblem sowie ein Parameterschätzproblem, formuliert und die Zielfunktion des Versuchsplanungsproblems aufgestellt, die eine implizite Funktion der Jacobi-Matrix des Parameterschätzproblems ist. Zur Lösung dieses Optimierungsproblems wird ein Verfahren der sequentiellen quadratischen Optimierung (SQP) eingesetzt, das den Gradienten der Lagrange-Funktion und der Nebenbedingungen benötigt. Die Herleitung aller benötigten Ableitungen schließt dieses Kapitel ab.

Kapitel 2 beschäftigt sich mit der Theorie des Automatischen Differenzierens (AD). Es werden die Verfahren zur automatischen Ableitungserzeugung erklärt und algorithmisch notiert. Nach einem kurzen historischen Abriß und einer Begriffsdefinition des AD werden die Zusammenhänge zwischen einem beliebigen mathematischen Ausdruck und einem gerichteten azyklischen Graphen erläutert (Abschnitt 2.1). Danach werden in Abschnitt 2.2 die beiden wichtigsten Verfahren des AD, Vorwärts- und Rückwärtsmodus, sowie deren Komplexität betrachtet. Anschließend werden Verfahren untersucht, die die Dünnbesetztheit von Jacobi-Matrizen effizient ausnutzen oder höhere Ableitungen behandeln können. In Abschnitt 2.3 wird das sogenannte Numerische Differenzieren mit Hilfe von Differenzenquotienten genauer betrachtet. Dabei steht die Untersuchung der Fehlerabschätzung theoretisch sowie an konkreten Beispielen im Vordergrund. Abschließend werden in Abschnitt 2.5 die kommerziellen Software-Pakete ADIFOR und ADOL-C (GRIEWANK ET AL. (1996)) zur automatischen Ableitungserzeugung genauer untersucht. Im Zentrum der Betrachtung steht dabei die Frage, wie gut diese Pakete für die Erzeugung der in dieser Arbeit benötigten Ableitungen geeignet sind.

[1] Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie

In Kapitel 3 werden Prinzipien aus der chemischen Reaktionskinetik verwendet, um das DAE-System aufzustellen. In Abschnitt 3.1 wird die Herleitung des Gleichungssystems aus einer physikalisch-chemischen Beschreibung erläutert. Daraufhin werden die Ableitungen der Modellgleichungen nach den Optimierungsvariablen in Abschnitt 3.2 analytisch aufgestellt.

Kapitel 4 beschäftigt sich mit der praktischen Implementierung der Arbeit. Nach allgemeinen Angaben zur Implementierung (Abschnitt 4.1) wird zunächst die Entwicklung eines Modellgenerators für chemische Reaktionssysteme in einem instationären homogenen Rührreaktor beschrieben. Dabei wird in Abschnitt 4.3 auf die in dieser Arbeit erstellte graphische Benutzeroberfläche und auf die Erzeugung der Quelltexte für die Modellgleichungen eingegangen. In den Abschnitten 4.4 und 4.5 werden die Prinzipien des AD aus Kapitel 2 angewendet, um die partiellen ersten und zweiten Ableitungen der erzeugten Modellgleichungen effizient und strukturausnutzend zu erzeugen.

Im letzten Teil der Arbeit (Kapitel 5) werden die Laufzeiten der Algorithmen zur Ableitungsberechnung untersucht. Zuerst werden einige praktische Beispiele von chemischen Reaktionssystemen aus der Industrie (BASF) angeführt. Danach wird anhand der beschriebenen Beispiele die Laufzeit der erzeugten Ableitungen ermittelt. Es werden die erzielten Laufzeiten mit den von ADIFOR erzeugten Ableitungen und dem Numerischen Differenzieren verglichen. Zuletzt werden die Ergebnisse diskutiert und weitere Einsatzmöglichkeiten des in der Arbeit erstellten Programms aufgezeigt.

In dieser Arbeit werden Grundkenntnisse über Differential- und Integralrechnung im $\mathbb{R}^n$ (Heuser (1990)) sowie über nichtlineare Optimierung (Gill et al. (1981), Bock (1998)) vorausgesetzt, auf die im einzelnen nicht näher eingegangen wird.

Danksagung

Danken möchte ich Prof. Dr. H. G. Bock und Dr. J. P. Schlöder für die Vergabe der Diplomarbeit und die interessante Themenauswahl sowie für die gute Betreuung dieser Arbeit. Mein besonderer Dank gilt Stefan Körkel für die umfangreiche Betreuung, die geduldige Unterstützung bei allen auftretenden mathematischen, physikalischen und chemischen Problemstellungen, lehrreichen Diskussionen in den Bereichen Numerik, Optimierung und Software-Engineering sowie dem Korrekturlesen. Danke auch an Irene Bauer und Angelika Dieses für zahlreiche Tips und Anregungen. Ein Dank geht auch an Zahra Hararat-Tehrani, Robert Jany, Rainer Rücker, Sebastian Sager und Christian Schepp für das sorgfältige Korrekturlesen, an Jörg Huber und Rüdiger Lang für Ratschläge und Tips zur Durchführung einer Diplomarbeit sowie an Dr. D. Homeister und Sebastian Mennicke für zahlreiche Inspirationen und Ratschläge bei der Programmierung. Ein Dank geht auch an die Mitglieder der AG Bock für die freundliche Aufnahme in ihren Arbeitskreis. Für die Bereitstellung der chemischen Beispiele bedanke ich mich bei Dr. A. Kud und allen weiteren Mitarbeitern der BASF AG (Ludwigshafen), die an der Modellierung beteiligt waren. Ein besonderer Dank geht auch an meine Eltern für ihre Unterstützung während meines Mathematik-Studiums.

Kapitel 1

Grundlagen

1.1 Mathematische Grundlagen

In den folgenden Abschnitten werden einige wichtige mathematische Definitionen und Sätze angegeben, die für die weiteren Kapitel von Bedeutung sind.

1.1.1 Differentiell-Algebraische Gleichungssysteme

Definition 1.1.1 *Sei* $t \in \mathcal{D} \subset I\!R$, $y(t) = (y_1(t), \ldots, y_{n_1}(t))^T$, $z(t) = (z_1(t), \ldots, z_{n_2}(t))^T$ *mit* $y_i, z_j : \mathcal{D} \longrightarrow I\!R$, $1 \leq i \leq n_1$, $1 \leq j \leq n_2$, $n_1, n_2 > 0$.

Das Gleichungssystem

$$\begin{aligned} \dot{y}(t) &= f(t, y(t), z(t)) \\ 0 &= g(t, y(t), z(t)) \end{aligned} \tag{1.1}$$

mit den stetig differenzierbaren Funktionen

$$\begin{aligned} f &: I\!R \times I\!R^{n_1} \times I\!R^{n_2} \longrightarrow I\!R^{n1} \\ g &: I\!R \times I\!R^{n_1} \times I\!R^{n_2} \longrightarrow I\!R^{n2} \end{aligned}$$

wird als Differentiell-Algebraisches Gleichungssystem, im folgenden mit DAE[1] *abgekürzt, bezeichnet.*[2]

Gilt zusätzlich:

1. Die Matrix $\left(\frac{\partial g}{\partial z}\right)$ ist quadratisch und regulär (Index-1 Annahme),
2. Die Anfangswerte $y(t_0) = (y_1(t_0), \ldots, y_{n_1}(t_0)) = y_0$ sind fest vorgegeben,
3. $\exists z_0 = (z_1(t_0), \ldots, z_{n_2}(t_0))$ mit $g(t_0, y_0, z_0) = 0$ (Konsistenzbedingung),

[1] Differential-Algebraic-Equation

[2] Es handelt sich bei dieser Darstellung um eine spezielle DAE in expliziter Form.

so existiert nach dem Satz über implizite Funktionen (1.1.2) eine lokal eindeutige Lösung. Das DAE-System (1.1) kann auch als Differentialgleichungssystem auf einer Mannigfaltigkeit interpretiert werden.

Bei der mathematischen Beschreibung von chemischen Reaktionsprozessen betrachten wir in den folgenden Kapiteln eine Variante des Gleichungssystems (1.1):

$$\begin{aligned} \dot{y}(t) &= f(t, y(t), z(t), p, q) \\ 0 &= g(t, y(t), z(t), p, q) \end{aligned} \tag{1.2}$$

Hierbei bezeichnet t die Zeit, y den Vektor der differentiellen Variablen, z den Vektor der algebraischen Variablen, sowie p bzw. q zusätzlich Parameter bzw. Steuergrößen. Ein Teil der Steuergrößen q kann sich dabei durch Diskretisierung von sogenannten Steuerfunktionen $u(t)$ ergeben.

1.1.2 Satz über implizite Funktionen

Der Satz über implizite Funktionen ist Grundlage vieler Aussagen und Beweise dieser Arbeit. Für einen vollständigen Beweis sei auf das Lehrbuch von HEUSER (1990) verwiesen.

Satz 1.1.2 *Gegeben seien nichtleere, offene Mengen $G \subset \mathbb{R}^p$ und $H \subset \mathbb{R}^q$ $(p, q \in \mathbb{N})$ sowie eine stetig differenzierbare Funktion $F : G \times H \to \mathbb{R}^q$. Seien weiter $x^* \in G$ und $y^* \in H$, für die gilt:*

1. $F(x^*, y^*) = 0$
2. $\dfrac{\partial F}{\partial y}(x^*, y^*)$ *ist invertierbar.*

Dann gibt es eine Umgebung $U \subset G$ von x^*, eine Umgebung $V \subset H$ von y^* und genau eine stetige Funktion

$$f : U \to V \text{ mit } f(x^*) = y^* \text{ und } F(x, f(x)) = 0 \quad \forall x \in U.$$

Für jedes feste $x \in U$ ist $f(x)$ sogar die einzige in V liegende Lösung der Gleichung $F(x, y) = 0$.

Folgender Satz stellt geringere Anforderungen an F als Satz 1.1.2, verlangt aber die Existenz einer stetigen, durch die Gleichung $F(x, y) = 0$ implizit definierten Funktion f. Die dabei verwendeten Matrizen $\partial F/\partial x$ und $\partial F/\partial y$ sind wie folgt definiert:

$$\frac{\partial F}{\partial x} := \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_p} \\ \vdots & & \vdots \\ \frac{\partial F_q}{\partial x_1} & \cdots & \frac{\partial F_q}{\partial x_p} \end{pmatrix} \text{ und } \frac{\partial F}{\partial y} := \begin{pmatrix} \frac{\partial F_1}{\partial y_1} & \cdots & \frac{\partial F_1}{\partial y_q} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_q}{\partial y_1} & \cdots & \frac{\partial F_q}{\partial y_q} \end{pmatrix} \tag{1.3}$$

Satz 1.1.3 *Die Mengen G und H sowie die Punkte x^* und y^* seien wie in Satz 1.1.2 definiert. $F : G \times H \to \mathbb{R}^q$ sei eine Funktion mit folgenden Eigenschaften:*

$$F(x^*, y^*) = 0,\ F'(x^*, y^*) \text{ ist vorhanden, und } \frac{\partial F}{\partial y}(x^*, y^*) \text{ ist invertierbar.}$$

Gibt es darüberhinaus eine Umgebung $U \subset G$ von x^*, eine Umgebung $V \subset H$ von y^* und eine stetige Funktion

$$f : U \to V \text{ mit } f(x^*) = y^* \text{ und } F(x, f(x)) = 0 \quad \forall x \in U,$$

so ist f an der Stelle x^* differenzierbar, und es gilt folgende Formel:

$$f'(x^*) = -\left(\frac{\partial F}{\partial y}(x^*, y^*)\right)^{-1} \frac{\partial F}{\partial x}(x^*, y^*) \tag{1.4}$$

Die Formel (1.4) erhält man dabei durch implizites Differenzieren von $F(x, f(x))$ mit Hilfe der Kettenregel (2.2.1):

$$\begin{aligned} & F(x, f(x)) = 0 \quad \forall x \in U \\ \Longrightarrow\ & \frac{d}{dx} F(x, f(x)) = \frac{\partial F}{\partial f}\frac{df}{dx} + \frac{\partial F}{\partial x} = 0 \\ \Longleftrightarrow\ & \frac{df}{dx} = -\left(\frac{\partial F}{\partial f}\right)^{-1} \frac{\partial F}{\partial x} \end{aligned}$$

1.1.3 Berechnung von Ableitungen

Um den Gradienten der Zielfunktion des in Abschnitt 1.3.4 dieser Arbeit betrachteten Optimierungsproblems zu bestimmen, benötigen wir auch Ableitungen von Funktionen nach Matrizen. Wir interessieren uns hierbei für Richtungsableitungen und benutzen dabei die folgende Schreibweise:

Sei $F : \mathbb{R}^{m \times n} \longrightarrow \mathbb{R}^N$, $X \longmapsto F(X)$ eine Matrix-Funktion und $\Delta X \in \mathbb{R}^{m \times n}$ eine gegebene Richtungsmatrix. Dann bezeichnen wir mit

$$\frac{d}{dX} F(X)\Delta X := \frac{d}{dt} F(X + t\Delta X)\Big|_{t=0} \in \mathbb{R}^N$$

die Richtungsableitung von F nach X in der Richtung ΔX.

In den folgenden Lemmata werden jetzt Ableitungsregeln von Matrizen nach Matrizen aufgestellt.

Lemma 1.1.4 *Seien $A \in \mathbb{R}^{l \times m}, X, \Delta X \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{n \times p}$ $(l, m, n, p \in \mathbb{N})$, dann gilt:*

$$\frac{d}{dX}(A \cdot X)\,\Delta X = A \cdot \Delta X \quad \text{und} \quad \frac{d}{dX}(X \cdot B)\,\Delta X = \Delta X \cdot B.$$

Beweis. Für $i \in \{1,\dots,l\},\ j \in \{1,\dots,n\},\ r \in \{1,\dots,m\}$ und $s \in \{1,\dots,n\}$ gilt unter Verwendung des Kronecker-Symbols δ_{ij} mit

$$\delta_{ij} := \begin{cases} 1, & i = j \\ 0, & \text{sonst} \end{cases}$$

$$(A \cdot X)_{ij} = \sum_{k=1}^{m} a_{ik} x_{kj} \Longrightarrow \frac{\partial}{\partial x_{rs}} (A \cdot X)_{ij} = \sum_{k=1}^{m} a_{ik} \frac{\partial}{\partial x_{rs}} x_{kj} = \sum_{k=1}^{m} a_{ik} \delta_{kr} \delta_{js} = a_{ir} \delta_{js}$$

$$\begin{aligned} \Longrightarrow \left(\frac{d}{dX} (A \cdot X)\, \Delta X \right)_{ij} &= \sum_{r=1}^{m} \sum_{s=1}^{n} \frac{\partial}{\partial x_{rs}} (A \cdot X)_{ij}\, \Delta x_{rs} = \sum_{r=1}^{m} \sum_{s=1}^{n} a_{ir} \delta_{js} \Delta x_{rs} \\ &= \sum_{r=1}^{m} a_{ir} \Delta x_{rj} = (A \cdot \Delta X)_{ij} \end{aligned}$$

Für $i \in \{1,\dots,m\},\ j \in \{1,\dots,p\},\ r \in \{1,\dots,m\}$ und $s \in \{1,\dots,n\}$ gilt:

$$(X \cdot B)_{ij} = \sum_{k=1}^{n} x_{ik} b_{kj} \Longrightarrow \frac{\partial}{\partial x_{rs}} (X \cdot B)_{ij} = \sum_{k=1}^{n} \frac{\partial}{\partial x_{rs}} x_{ik} b_{kj} = \sum_{k=1}^{n} \delta_{ir} \delta_{ks} b_{kj} = \delta_{ir} b_{sj}$$

$$\begin{aligned} \Longrightarrow \left(\frac{d}{dX} (X \cdot B)\, \Delta X \right)_{ij} &= \sum_{r=1}^{m} \sum_{s=1}^{n} \frac{\partial}{\partial x_{rs}} (X \cdot B)_{ij}\, \Delta x_{rs} = \sum_{r=1}^{m} \sum_{s=1}^{n} \delta_{ir} b_{sj} \Delta x_{rs} \\ &= \sum_{s=1}^{n} \Delta x_{is} b_{sj} = (\Delta X \cdot B)_{ij} \end{aligned}$$

■

Korollar 1.1.5 *Seien $A, X, \Delta X, B$ wie in Lemma 1.1.4, dann gilt:*

$$\frac{d}{dX} (A \cdot X \cdot B)\, \Delta X = A \cdot \Delta X \cdot B$$

Beweis. Mit 1.1.4 gilt:

$$\begin{aligned} \frac{d}{dX} (A \cdot X \cdot B)\, \Delta X &= \frac{d}{dX} ((A \cdot X) \cdot B)\, \Delta X = \frac{d}{dX} (A \cdot X)\, \Delta X \cdot B \\ &= A \cdot \Delta X \cdot B. \end{aligned}$$

■

Lemma 1.1.6 *Sei $X, \Delta X \in \mathbb{R}^{m \times n}$ $(m, n \in \mathbb{R})$, dann gilt:*

$$\frac{d}{dX} \left(X^T \right) \Delta X = \Delta X^T.$$

Beweis. Für $i, r \in \{1,\dots m\}$ und $j, s \in \{1,\dots,n\}$ gilt:

$$\begin{aligned} \left(\frac{d}{dX} \left(X^T \right) \Delta X \right)_{ij} &= \sum_{r=1}^{m} \sum_{s=1}^{n} \frac{\partial}{\partial x_{rs}} \left(X^T \right)_{ij} \Delta x_{rs} = \sum_{r=1}^{m} \sum_{s=1}^{n} \frac{\partial}{\partial x_{rs}} x_{ji} \Delta x_{rs} \\ &= \sum_{r=1}^{m} \sum_{s=1}^{n} \delta_{jr} \delta_{is} \Delta x_{rs} = \Delta x_{ji} = \left(\Delta X^T \right)_{ij} \end{aligned}$$

■

Satz 1.1.7 *Sei $X \in \mathbb{R}^{n\times n}$ invertierbar, dann gilt:*

$$\boxed{\frac{d}{dX}(X^{-1})\Delta X = -X^{-1}\Delta X X^{-1}.} \qquad (1.5)$$

Beweis. Wir setzen $F : \mathbb{R}^{n\times n} \times R^{n\times n} \longrightarrow \mathbb{R}^{n\times n}$ mit $F(X,Y) := Y \cdot X - I_n$. Für X invertierbar ist $Y := f(X) = X^{-1}$ die Lösung von $F(X,Y) = 0$:

$$F(X,Y) = F(X, f(X)) = F(X, X^{-1}) = X^{-1} \cdot X - I_n = I_n - I_n = 0.$$

F ist stetig differenzierbar und mit Lemma 1.1.4 gilt:

$$\frac{\partial}{\partial X}F\,(X,Y)\,\Delta X = Y \cdot \Delta X, \quad \frac{\partial}{\partial Y}F\,(X,Y)\,\Delta Y = \Delta Y \cdot X \qquad (1.6)$$

und weiter, daß

$$\frac{\partial F}{\partial Y}\,(X,Y) = X$$

invertierbar ist.

Somit sind alle Voraussetzungen für den Satz über implizite Funktionen 1.1.3 erfüllt, und man erhält:

$$\frac{\partial}{\partial X}F\,(X,Y)\,\Delta X + \frac{\partial}{\partial Y}F\,(X,Y) \cdot \underbrace{\frac{dY}{dX}\Delta X}_{\Delta Y} = 0 \qquad (1.7)$$

Mit Gleichung (1.6) folgt:

$$\begin{array}{rrcl} & Y \cdot \Delta X + \Delta Y \cdot X & = & 0 \\ \Leftrightarrow & \Delta Y \cdot X & = & -Y\Delta X \\ \Leftrightarrow & \Delta Y = \dfrac{dY}{dX}\Delta X & = & -Y\Delta X X^{-1} \end{array}$$

Mit $Y = X^{-1}$ ergibt sich somit:

$$\frac{dX^{-1}}{dX}\Delta X = -X^{-1}\Delta X X^{-1} \Rightarrow \text{Behauptung.} \qquad (1.8)$$

■

1.2 Grundlagen der chemischen Verfahrenstechnik

Aufgabe dieses Abschnitts ist es, die DAEs, die den Reaktionsverlauf chemischer Reaktionen beschreiben, aus physikalisch-chemischen Gesetzen herzuleiten. Dafür werden zu Beginn grundlegende Zusammenhänge aus der chemischen Reaktionskinetik und der chemischen Verfahrenstechnik beschrieben.

1.2.1 Reaktionskinetik

Die chemische Reaktionskinetik, die ein Teilgebiet der Physikalischen Chemie darstellt, beschäftigt sich mit der Frage, wie die Geschwindigkeit bei chemischen Reaktionen von den Mengen der an der Reaktion beteiligten Stoffe, der Temperatur und anderen äußeren Faktoren zeitlich abhängt.

Reaktionsgeschwindigkeit

Unter der *Reaktionsgeschwindigkeit* einer Reaktion versteht man die pro Zeiteinheit reagierende Zahl der Moleküle bzw. die Molzahl, bezogen auf das Volumen. Die Reaktionsgeschwindigkeit ist daher eine zeitlich abhängige Größe, die diejenige Rate angibt, mit der eine Spezies im Reaktionsverlauf zu einem gegebenen Zeitpunkt zu- bzw. abnimmt. Die Reaktionsgeschwindigkeit ist daher die Ableitung der Konzentrationsfunktion nach der Zeit. Die Konzentrationen der Spezies werden in der Regel in $[mol/l]$ angegeben (BARROW (1973)).

Für eine chemische Reaktion mit den Konzentrationen c_{A_i} der Edukte A_i, den Konzentrationen c_{B_j} der Produkte B_j, den stöchiometrischen Koeffizienten α_i und β_i und der Reaktionsgleichung

$$\alpha_1 A_1 + \cdots + \alpha_m A_m \longrightarrow \beta_1 B_1 + \cdots + \beta_l B_l$$

definiert man als Reaktionsgeschwindigkeit der Edukte A_i bzw. der Produkte B_j:

$$r_v(t) = -\frac{1}{\alpha_i}\frac{dc_{A_i}}{dt} \qquad \text{bzw.} \qquad r_v(t) = \frac{1}{\beta_j}\frac{dc_{B_j}}{dt}. \tag{1.9}$$

Die Reaktionsgeschwindigkeit r_v läßt sich im allgemeinen durch einen temperaturabhängigen Term $k(T)$ sowie einen konzentrationsabhängigen Term $F(c)$ darstellen. Es gilt:

$$r_v(t) = k(T) \cdot F(c) \tag{1.10}$$

Eine wichtige Aufgabe der numerischen Mathematik besteht im Lösen (Integration) der Differentialgleichung (1.10) bzw. von Differentialgleichungssystemen (beispielsweise das DAE-System (1.2)), die bei mehreren gekoppelten chemischen Reaktionen entstehen.

Der Ansatz von Arrhenius

Im Jahre 1889 fand der Chemiker Arrhenius anhand empirischer Untersuchungen heraus, daß sich der temperaturabhängige Term $k(T)$ durch folgende Funktion darstellen läßt:

$$\boxed{k(T) = k_{ref} \cdot \exp\left(-\frac{E_a}{R \cdot T}\right)} \tag{1.11}$$

Dabei stellen die Ausdrücke E_a die *Aktivierungsenergie* (in $[kJ/mol]$) der chemischen Reaktion, und k_{ref} den sogenannten *Frequenzfaktor* dar.

Die Stoßtheorie

Die Stoßtheorie wurde von Lewis im Jahre 1918 entwickelt und basiert auf den Ergebnissen der *kinetischen Gastheorie*. Eine einfache bimolekulare Reaktion $A + B \longrightarrow C$ kann nur durch Zusammenstöße von Molekülen der Spezies A und B erfolgen. Die Geschwindigkeit der beiden Moleküle muß aber größer sein als eine kritische *Relativgeschwindigkeit*, bei der es zur Überwindung der *van-der-Waalschen* Abstoßungskräfte kommt. Daraus folgt, daß von den Stößen zweier Eduktmoleküle A und B nur diejenigen mit einem Mindestbetrag an kinetischer Energie erfolgreich sind. Diese Mindestenergie wird *Aktivierungsenergie* genannt.
Nach VAUCK & MÜLLER (1978) ist die Anzahl *aller* Stöße zwischen den Molekülen A und B durch folgende Gleichung gegeben:

$$Z_{AB} = \pi \sigma_{AB}^2 \sqrt{\frac{8 \cdot R \cdot T}{\pi \mu_{AB}}} \cdot \frac{N_A \cdot N_B}{V^2} \tag{1.12}$$

mit den Größen:

- der reduzierten molekularen Masse μ_{AB} von A und B mit $\mu_{AB} = \frac{M_A \cdot M_B}{M_A + M_B}$ (M_A und M_B bezeichnen die molekulare Massen von A bzw. B in $[kg/mol]$)
- dem Stoßdurchmesser σ_{AB} mit $\sigma_{AB} = r_A + r_B$ aus den Radien der Moleküle A und B
- der Teilchenanzahl N_A und N_B beider Moleküle
- sowie der allgemeinen Gaskonstante $R = 8.314\ J/mol$

Von diesen Stößen sind nur diejenigen Z_{AB}^1 erfolgreich, die die Schwellenenergie E_a überschreiten:

$$Z_{AB}^1 = Z_{AB} \cdot \exp\left(-\frac{E_a}{R \cdot T}\right) \tag{1.13}$$

Der Vergleich des Arrhenius-Ansatzes mit der Stoßtheorie zeigt, daß der präexponentielle Frequenzfaktor k_{ref} proportional zu $\sqrt{T}$ ist. Nach diesem Ansatz kann man für einfach gebaute Moleküle k_{ref} in der richtigen Größenordnung bestimmen. Bei komplizierter gebauten Molekülen wird noch ein Korrektur-Koeffizent, der *sterische Faktor*, an die rechte Seite multipliziert, da nicht jeder Stoß, bei dem die kinetischen Energien der Stoßpartner groß genug sind, zur Reaktion führt. Somit läßt sich der temperaturabhängige Term der Reaktionsgeschwindigkeit durch den erweiterten Arrhenius-Ansatz

$$k(T) = T^p \cdot \exp\left(-\frac{E_a}{R \cdot T}\right), \quad (p < 1) \tag{1.14}$$

annähernd bestimmen.

Die Konzentrationsfunktion

Die Abhängigkeit der Reaktionsgeschwindigkeit von den einzelnen Konzentrationen der an der Reaktion beteiligten Spezies wird in der Regel durch einen Potenzansatz beschrieben. (Hierbei steht α_i für die *Reaktionsordnung* der Spezies A_i.):

$$F(c) = \prod_i c_i^{\alpha_i} \tag{1.15}$$

Als Beispiel betrachte man die Reaktion $\alpha_1 A_1 + \alpha_2 A_2 + \cdots + \alpha_m A_m \longrightarrow \beta_1 B_1$. Nach den Gleichungen (1.14) und (1.15) ergibt sich der formalkinetische Ansatz:

$$r_v(t) = -\frac{1}{\alpha_1}\frac{dc_{A_1}}{dt} = -\frac{1}{\alpha_2}\frac{dc_{A_2}}{dt} = \cdots = -\frac{1}{\alpha_m}\frac{dc_{A_m}}{dt} = \frac{1}{\beta_1}\frac{dc_{B_1}}{dt} = k_{ref} \cdot \prod_{i=1}^{m} c_{A_i}^{\alpha_i} \tag{1.16}$$

Die Exponenten $\alpha_1, \alpha_2, \ldots, \alpha_m$ stellen die *Einzelordnungen* der Spezies $A_1, A_2, \ldots, A_m$ dar. Die Summe $\sum_{i=1}^{m} \alpha_i$ wird auch als *Gesamtordnung* der Reaktion bezeichnet (siehe auch Abschnitt 1.2.3).

Massenerhaltungssatz

Betrachten wir beispielsweise eine chemische Gleichung der Form:

$$3A + 2B \longrightarrow 5C$$

In dieser Gleichung werden die Reaktionsedukte A und B eindeutig mit dem Reaktionsprodukt C verknüpft. Aufgrund der Massenerhaltung bei chemischen Reaktionen läßt sich die Gleichung in Form einer algebraischen Gleichung umschreiben, indem die abreagierenden Edukte als negative und die gebildeten Produkte als positive Größen formuliert werden. Man erhält daher mit den molekularen Massen M_A, M_B, M_C der beteiligten Spezies:

$$-3M_A - 2M_B + 5M_C = 0$$

Die allgemeine Formulierung des Massenerhaltungssatzes lautet:

$$\sum_i \nu_i M_i = 0 \tag{1.17}$$

Man nennt die Koeffizienten ν_i auch *stöchiometrische Koeffizienten*. Für die ν_i gilt:

Edukte:	$\nu_i < 0$
Produkte:	$\nu_i > 0$
Inertstoffe:	$\nu_i = 0$

Die Summe aller stöchiometrischen Koeffizienten v_i einer Reaktion wird in diesem Kontext durch $\bar{v} = \sum_i \nu_i$ festgelegt.

Chemische Reaktionen können unter *Stoffmengenkonstanz* oder unter *Stoffmengenänderung* ablaufen. Eine Stoffmengenänderung kann eine Auswirkung auf das *Reaktionsvolumen* haben. Kann man diesen Effekt in nicht zu konzentrierten Lösungen im allgemeinen vernachlässigen, ist er bei Gasreaktionen dagegen bedeutend. In diesem Fall spricht man von einer *Volumenänderung* einer chemischen Reaktion. Folgende Stoffmengenänderungen chemischer Reaktionen können auftreten:

$\bar{\nu} = 0$	:	Reaktionen *ohne Stoffmengenänderung* z.B. $H_2 + J_2 \longrightarrow 2HJ$, $\bar{\nu} = 0$
$\bar{\nu} > 0$	:	Reaktionen *mit Stoffmengenzunahme* z.B. $C_3H_8 + 5O_2 \longrightarrow 3CO_2 + 4H_20$, $\bar{\nu} = 1$
$\bar{\nu} < 0$	:	Reaktionen *mit Stoffmengenabnahme* z.B. $N_2 + 3H_2 \longrightarrow 2NH_3$, $\bar{\nu} = -2$

1.2.2 Klassifikation chemischer Reaktionen

Die Einteilung chemischer Reaktionen wird im folgenden nach JAKUBITH (1991) vorgenommen.

Homogene und heterogene Reaktionen

Bei der kinetischen Behandlung von chemischen Reaktionen erweist es sich als zweckmäßig eine Trennung in *homogene* und *heterogene* Reaktionen vorzunehmen. Homogene Reaktionen laufen in einer einzigen Mischphase ab, heterogene Reaktionen laufen in mindestens zwei Phasen ab, wobei den Phasengrenzen und den Transportvorgängen große Bedeutung zukommt.

Heterogene Reaktionen unterscheiden sich in ihrer mathematischen Beschreibung von homogenen Reaktionen oft durch die Überlagerungen mit dem diffusiven Transportprozeß.

Elementarreaktionen

Man versucht in der chemischen Reaktionskinetik, ein komplexes Reaktionsgeschehen von Reaktionssystemen auf wenige, überschaubare Typen von Grundgleichungen zurückzuführen. Dies kann man mit der Einführung einfach strukturierter *Elementarreaktionen* machen. Man versucht daher, komplexere Reaktionsgleichungen mit einer nichttrivialen Stöchiometrie in Teilreaktionen zu zerlegen.

Man unterscheidet in der chemischen Kinetik folgende Elementarreaktionen, die als die wichtigsten erachtet werden:

1.	Monomolekularer Zerfall	:	$AB \longrightarrow A + B$	$r_{v,1}$	=	$k_1 c_{AB}$
2.	Rekombination	:	$A + B \longrightarrow AB$	$r_{v,2}$	=	$k_2 c_A c_B$
3.	Bimolekularer Austausch	:	$AB + C \longrightarrow AC + B$	$r_{v,3}$	=	$k_3 c_{AB} c_C$
4.	Umlagerung	:	$ABC \longrightarrow ACB$	$r_{v,4}$	=	$k_4 c_{ABC}$

Dabei bezeichnen die Terme $r_{v,1}$ bis $r_{v,4}$ die Reaktionsgeschwindigkeiten der jeweiligen Reaktionen. Diese Elementarreaktionen laufen stets irreversibel ab. Ein Reaktionsmechanismus kann als aufgeklärt gelten, wenn die Art der Elementarreaktionen und deren gegenseitige Verknüpfung ermittelt ist. In diesem Fall läßt sich das Zeitgesetz einer chemischen Reaktion aus den Elementarreaktionen ermitteln.

Parallelreaktion

Die kinetische Behandlung von Parallelreaktionen ist vor allem in der organischen Chemie von Bedeutung. Als Beispiel läßt sich die Ethenoxidation zu Kohlendioxid einerseits und zu Acetaldehyd andererseits anführen:

$$\begin{aligned} C_2H_4 + 3O_2 &\xrightarrow{k_1} 2CO_2 + 2H_2O \\ C_2H_4 + \tfrac{1}{2}O_2 &\xrightarrow{k_2} CH_3CHO \end{aligned}$$

Dieser Typ von Reaktion läßt sich allgemein durch folgendes Reaktionsschema beschreiben.

$$A \xrightarrow{k_1} B_1, \quad A \xrightarrow{k_2} B_2, \quad \ldots, \quad A \xrightarrow{k_n} B_n$$

Für die Reaktionsgeschwindigkeit ergibt sich folgendes Zeitgesetz:

$$-\frac{dc_A}{dt}(t) = \sum_{i=1}^{n} k_i c_A(t) := K \cdot c_A(t) \tag{1.18}$$

Die einfache Integration dieser Gleichung durch Variablentrennung liefert das Zeitgesetz:

$$c_A(t) = c_A(t_0) \cdot \exp(-K \cdot t) \tag{1.19}$$

Folgereaktion und Kettenreaktion

Bei einer Folgereaktion reagiert das bei einer ersten Reaktion gebildete Produkt weiter zu einem Folgeprodukt. Betrachten wir in obigem Beispiel zur Ethenoxidation die partielle Weiteroxidation des Acetaldehyds zu Essigsäure und Kohlendioxid:

$$CH_3CHO + \frac{1}{2}O_2 \xrightarrow{k_1} CH_2COOH + 2O_2 \xrightarrow{k_2} 2CO_2 + 2H_2O$$

Wieder läßt sich die Folgereaktion allgemein beschreiben durch:

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C$$

Die Zeitgesetze bezüglich der Spezies A, B und C lauten somit:

$$-\frac{dc_a}{dt}(t) = k_1 c_A(t), \quad \frac{dc_B}{dt}(t) = k_1 c_A(t) - k_2 C_B(t), \quad \frac{dc_C}{dt}(t) = k_2 c_B(t) \tag{1.20}$$

Das Zeitgesetz $c_A(t)$ für die Spezies A erhält man wiederum durch Trennung der Variablen und Integration. Dieser Ausdruck wird in die Differentialgleichung bezüglich B eingesetzt, und es ergibt sich:

$$\frac{dc_B}{dt}(t) = k_1 c_A(t_0) \cdot \exp\left(-k_1 t\right) - k_2 c_B(t) \tag{1.21}$$

Diese Differentialgleichung läßt sich einfach mit Hilfe einer *Laplace-Transformation* lösen. Im Falle einer nichtlinearen Differentialgleichung kann auch die *Variation der Konstanten* nach *Lagrange* zur analytischen Lösung angewendet werden (vergleiche WALTER (1991)).

Besteht das chemische Reaktionssystem aus mehreren Folgereaktionen, bei denen der Ausgangsstoff A immer mit einem gebildeten Produkt reagiert, so spricht man von einer *Kettenreaktion.*

1.2.3 Ordnung von Reaktionen

Die reellwertige Ordnung einer chemischen Reaktion kann nach JAKUBITH (1991) meist nicht vorhergesagt werden. Sie läßt sich daher im allgemeinen Fall *nicht* aus der Stöchiometrie der Reaktion entnehmen. Man kann dies nur bei den sogenannten Elementarreaktionen aus Abschnitt 1.2.2 machen.

Die Geschwindigkeitsgleichung erster Ordnung

$$-\frac{dc_{A_1}(t)}{dt} = k_r c_{A_1}(t) \tag{1.22}$$

stellt eine Differentialgleichung dar, die man durch Variablentrennung integrieren kann.

$$\begin{aligned}
\int_{c_{A_1}(t_0)}^{c_{A_1}(t)} -\frac{dc_{A_1}}{c_{A_1}(t)} &= -k_r \int_{t_0}^{t} dt \\
\ln\left(\frac{c_{A_1}(t)}{c_{A_1}(t_0)}\right) &= k_r(t_0 - t) \\
c_{A_1}(t) &= c_{A_1}(t_0) e^{k_r(t_0-t)}
\end{aligned}$$

Geschwindigkeitsgleichungen für Reaktionen zweiter Ordnung haben folgende Struktur:

$$-\frac{dc_{A_1}}{dt}(t) = k_r \cdot c_{A_1}(t) \cdot c_{A_2}(t) \quad \text{oder} \quad -\frac{dc_{A_1}}{dt}(t) = k_r \cdot [c_{A_1}(t)]^2$$

Im ersten Fall besteht nicht immer die Möglichkeit, die Konzentrationen beider Reaktionspartner gleich groß zu wählen. Hier ist es zweckmäßiger, die Eduktkonzentration c in der Geschwindigkeitsgleichung durch die Anfangskonzentrationen minus der Produktkonzentrationen der bereits entstandenen Produkte auszudrücken. Für eine Reaktion $A_1 + A_2 \longrightarrow A_3$ werden die Größen $a_1 :=$ Anfangskonzentration von A_1, $a_2 :=$ Anfangskonzentration von A_2 und $x :=$ Konzentration des Endproduktes A_3 zum Zeitpunkt t gesetzt. Mit diesen abgeänderten Größen ergibt sich folgende Geschwindigkeitsgleichung:

$$v(t) = \frac{dx}{dt} = k_r(a_1 - x)(a_2 - x) \tag{1.23}$$

Diese Differentialgleichung kann man mit Hilfe einer Partialbruchzerlegung analytisch lösen:

$$\begin{aligned}
\frac{dx}{(a_1 - x)(a_2 - x)} &= k_r dt \\
\frac{1}{a_1 - a_2} \int_{x=0}^{x} \left(-\frac{1}{a_1 - x} + \frac{1}{a_2 - x}\right) dx &= k_r \int_{t=0}^{t} t dt
\end{aligned}$$

Dadurch erhält man:

$$\frac{1}{a_1 - a_2} \left[\ln(a_1 - x) - \ln(a_2 - x)\right]_{x=0}^{x} = k_r t$$

Nach Einsetzen der Grenzen ergibt sich:

$$\frac{1}{a_1 - a_2} \ln\left(\frac{a_1 - x}{a_2 - x}\right) = k_r t$$

Die Differentialgleichung der Geschwindigkeitsgleichung für eine Reaktionsgleichung der Form $A_1 + A_2 + A_3 \longrightarrow C$ mit den Anfangskonzentrationen a_1, a_2, a_3 der Edukte und der Konzentration x des Produkts C lautet:

$$v(t) = \frac{dx}{dt} = k_r(a_1 - x)(a_2 - x)(a_3 - x) \tag{1.24}$$

Es handelt sich hierbei um eine Reaktion dritter Ordnung. Nach MOELWYN-HUGHES (1970) ergibt die analytische Integration:

- Fall 1: $a_1 \neq a_2 \neq a_3 \neq a_1$:

$$\frac{\ln\left(\left(\frac{a_1}{a_1-x}\right)^{a_2-a_3} \cdot \left(\frac{a_2}{a_2-x}\right)^{a_3-a_1} \cdot \left(\frac{a_3}{a_3-x}\right)^{a_1-a_2}\right)}{(a_1 - a_2)(a_2 - a_3)(a_3 - a_1)} = -k_r t$$

- Fall 2: $a_1 \neq a_2 = a_3$:

$$\frac{1}{(a_1 - a_2)^2}\left(\frac{(a_1 - a_2)x}{a_2(a_2 - x)} + \ln\left(\frac{a_1(a_2 - x)}{a_2(a_1 - x)}\right)\right) = k_r t$$

- Fall 3: $a_1 = a_2 = a_3$:

$$\frac{1}{2}\left(\frac{1}{(a_1 - x)^2} - \frac{1}{a_1^2}\right) = k_r t$$

Die analytische Integration einer Reaktionsgleichung n-ter Ordnung ist nur dann möglich, wenn die stöchiometrischen Koeffizienten aller Edukte gleich sind. D.h. die Reaktionsgleichung hat die allgemeine Form

$$\nu A_1 + \nu A_2 + \cdots + \nu A_n \longrightarrow P$$

mit den Anfangsbedingungen $a_1 = a_2 = \cdots = a_n = a$. Die Geschwindigkeitsgleichung lautet also:

$$\frac{dx}{dt} = k_r(a - x)^n \tag{1.25}$$

Variablentrennung und Integration liefern hierbei:

$$\frac{1}{(n-1)}\left[\frac{1}{(a-x)^{(n-1)}} - \frac{1}{a^{(n-1)}}\right] = k_r t \tag{1.26}$$

1.2.4 Chemische Reaktoren

Um die physikalischen Gleichungen in Kapitel 3 aufstellen zu können, benötigt man zuerst einige Definitionen nach VAUCK & MÜLLER (1978) aus der chemischen Verfahrenstechnik.

Reaktor: Ein Reaktor ist ein Apparat, in dem chemische Reaktionen zur Gewinnung bestimmter Produkte durchgeführt werden. Für die Reaktoren existieren verschiedene Formen der Gliederung. Oft teilt man sie in thermische, elektrochemische und photochemische Reaktoren ein. Dieser Form der Gliederung liegt die Art der Energiezufuhr zugrunde, da eine chemische Reaktion im allgemeinen erst dann erfolgt, wenn

die Aktivierungsschwelle der bei der Reaktion auftretenden Moleküle überwunden wird. Diese kann durch Zufuhr thermischer, elektrochemischer, mechanischer oder auch photochemischer Energie erfolgen. Dabei werden im folgenden nur diejenigen Reaktoren beschrieben, in denen thermische Vorgänge die Hauptrolle spielen, da sie in der chemischen Industrie am weitesten verbreitet sind.

Reaktionsmasse: Als Reaktionsmasse M_{ges} bezeichnet man die Gesamtmasse der einzelnen im Reaktor befindlichen Stoffe. Dies sind die an der chemischen Reaktion beteiligten Reaktanden (Reaktionsedukte und Reaktionsprodukte), Inert- und Begleitstoffe (Lösungsmittel, Inertgase, Zuschläge) sowie Katalysatoren.

Reaktionsvolumen: Bei festen und flüssigen Reaktionsgemischen ist das Reaktionsvolumen V mit den Molmassen M_i, den Stoffmengen n_i sowie den Dichten ϱ_i der Spezies mit folgender Gleichung gegeben:

$$V = \sum_i \frac{M_i \cdot n_i}{\varrho_i} \tag{1.27}$$

Umsatz: Man definiert den Umsatz U_i einer an der Reaktion beteiligten Komponente i durch:

$$U_i(t) = \frac{n_i(t_0) - n_i(t)}{n_i(t_0)} := \frac{V_i(t_0)c_i(t_0) - V_i(t)c_i(t)}{V_i(t_0)c_i(t_0)} \tag{1.28}$$

Dabei bezeichnet $n_i(t)$ die Stoffmenge des Stoffes A_i zum Zeitpunkt t. Läuft die Reaktion unter Stoffmengenkonstanz ab, so gilt $V_i(t) = V_i(t_0)$ für alle $t \in [t_0, \ldots, t_{end}]$. Nur in diesem Fall kürzt sich das Volumen heraus. Nach JAKUBITH (1991) gilt dies auch für stoffmengenändernde Reaktionen in Lösungen, jedoch keinesfalls für stoffmengenändernde Gasreaktionen. Hier ist die Definition des Umsatzes über die Stoffmengen n_i praktikabler.

Um das Volumen V im Reaktor zu verändern, können Zulauf- bzw. Ablaufhähne am Reaktor vorhanden sein. Dadurch kann der Experimentator den Reaktionsverlauf im Reaktor durch die Dosierung der Anfangskonzentrationen oder Anfangsstoffmengen der Spezies im Eintrag sowie der Zuflußraten der einzelnen Einträge steuern. Zum Beispiel kann bei der Reaktion zweier Stoffe A und B der Stoff A zu Beginn im Reaktor vorhanden sein und Stoff B wird über einen Zulauf in den Reaktor eingefügt. Die Modellierung eines Zulaufes bzw. Ablaufes kann durch Angabe einer zeitabhängigen Funktion $feed(t)$ erfolgen, die die Durchtrittsrate des jeweiligen Ein- bzw. Austrags zum Zeitpunkt t angibt (siehe Abschnitt 3.1.3). Die Abbildung 1.1 zeigt schematisch einen Rührkessel mit einem Zulauf. Die in Abschnitt 5.3 näher spezifizierte Phosphin-Reaktion wird z.B. in einem solchen Apparat durchgeführt.

Nach JAKUBITH (1991) werden in der chemischen Verfahrenstechnik folgende unterschiedliche Betriebsarten chemischer Reaktoren unterschieden:

Die diskontinuierliche Reaktionsführung (batch-Betrieb)

Hier wird der Reaktor mit der Reaktionsmasse gefüllt und am Ende der Reaktion wieder entleert. Man bezeichnet diese Reaktionsführung daher auch als *instationär* bzw. *Chargen-*

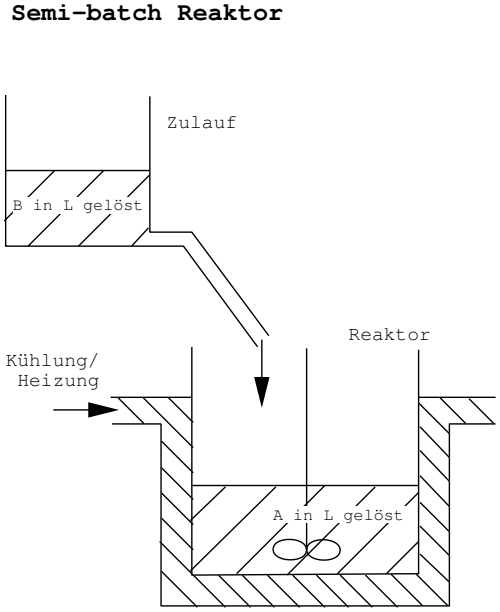


Abbildung 1.1: Reaktionsanordnung bei einer semi-batch Reaktion

oder *Satzbetrieb*. In diesem Fall kommt stets ein *Rührkessel* zum Einsatz, der durch eine stetige Rührung für eine gleichmäßige Verteilung der Reaktionsmasse sorgt.

Als Beispiel für einen Reaktor mit diskontinuierlicher Reaktionsführung kann der diskontinuierlich arbeitende, homogene Reaktionskessel, auch Rührkessel genannt, betrachtet werden. Hier hat die Reaktionsmasse an jeder Stelle die gleiche Zusammensetzung und Temperatur, die sich aber zeitlich als Folge des fortschreitenden Umsatzes ändern kann. Für diesen Rührkessel werden durch die Mischwirkung des Rührers die Temperaturen und Konzentrationen wie beim kontinuierlichen Rührkessel örtlich konstant gehalten.

Die kontinuierliche Reaktionsführung

Im Gegensatz zur diskontinuierlichen Reaktionsführung wird beim kontinuierlichen Reaktor ständig ein Stoffstrom der Edukte in den Reaktor eingespeist sowie ein Stoffstrom der Produkte entnommen. Diese Reaktionsführung, auch als *stationäre* Reaktionsführung bezeichnet, zeichnet sich durch die zeitliche Konstanz der Reaktionsparameter Konzentration und Temperatur aus. Als typische Vertreter dieser Art der Reaktionsführung gelten folgende Reaktoren:

- Kontinuierlich arbeitender homogener Reaktionskessel:

  Bei diesem Reaktortyp wird stetig Reaktionsmasse hinzugeführt und abgeführt. Ein Rührer sorgt dafür, daß die Reaktionsmasse an jeder Stelle im Kessel die gleiche Zusammensetzung und Temperatur hat. Eine Variation davon ist eine *Serie von kontinuierlich arbeitenden homogenen Reaktionskesseln*. Hier wird der gesamte Reaktionsprozeß auf n Kessel verteilt. Die Endkonzentration $c_{i,t_{end}}$ des i-ten Kessels ist gleich der Anfangskonzentration $c_{(i+1),t_0}$ des $(i+1)$-ten Kessels $(i = 1, \ldots, n-1)$.

- Kontinuierlich arbeitender Rohrreaktor:

  Die Reaktionsmasse durchströmt hier kontinuierlich einen langgestreckten Reaktor (auch *Strömungsrohr* genannt). Nach einer kurzen Anlaufperiode stellt sich an jedem Ort des Strömungsrohrs eine bestimmte Konzentration ein, die sich mit der Zeit nicht mehr verändert. Die Konzentration ist also *stationär*. Zu diesem Typ von Reaktor gehören alle Reaktoren, die eine kontinuierliche Konzentrationsänderung besitzen (stationäre Konzentration). Beispiel hierfür sind der Großraumreaktor und Reaktoren für gasförmig-flüssige und gasförmig-feste Systeme.

  Eine Variante davon ist eine *Serie von adiabatischen Reaktionsrohren*. Kennzeichnend für das adiabatische Reaktionsrohr ist die thermische Isolierung des Reaktionsraums, d.h. daß die Reaktionsmasse von außen weder gekühlt noch erwärmt werden kann.

Die halbkontinuierliche Reaktionsführung (semi-batch Betrieb)

Bei diesem Reaktortyp wird mindestens ein Edukt diskontinuierlich in den Rührkessel eingeführt und mindestens ein Edukt kontinuierlich der Reaktionsmasse zugeführt. Das Reaktionsprodukt kann kontinuierlich abgeführt werden. Diese Art der Reaktionsführung wird nach JAKUBITH (1991) bevorzugt bei stark exothermen Reaktionen gewählt, da die Temperatur des Reaktionsgemischs durch die stetige Zugabe eines Eduktes gut gesteuert werden kann.

1.3 Optimale Versuchsplanung

1.3.1 Aufgaben der Versuchsplanung

Der Gegenstand der optimalen Versuchsplanung besteht darin, Versuchspläne, die Versuchsszenarien beschreiben, herzuleiten, mit dem Ziel, die relevanten Aussagen (z.B. die Bestimmung unbekannter Parameter) mit möglichst großer (optimaler) Genauigkeit oder mit möglichst geringem (optimalem) Versuchsaufwand zu erhalten.[3] Versucht man, die Auslegung der Experimente und die Auswahl der Messungen so vorzunehmen, daß die statistische Güte der aus den Meßdaten geschätzten Parameter maximal ist, so wird eine Funktion ϕ auf der Kovarianzmatrix $C(s, q, w)$, die ein Maß für die Güte der geschätzten Parameter darstellt, minimiert (vgl. BAUER ET AL. (1998b)). Die Grenzen für den Aufwand zur Durchführung der Experimente bzw. der Messungen werden als Nebenbedingungen des Versuchsplanungsproblems formuliert.

Das Versuchsplanungsproblem ist also ein nichtlineares Optimierungsproblem mit einer Zielfunktion, die auf der Kovarianzmatrix des zugrundeliegenden Parameterschätzproblems definiert wird. Diese beiden Probleme werden in den nächsten Abschnitten formuliert.

[3] Handelt es sich bei den betrachteten Modellen um nichtlineare Systeme, so spricht man auch von nichtlinearer Versuchsplanung.

1.3.2 Zugrundeliegendes Parameterschätzproblem

Die unbekannten Parameter eines Modells lassen sich in der Regel nicht direkt messen. Deshalb werden sie aus verschiedenen Meßdaten η_{ij} $(i = 1, \ldots, m,\ j = 1, \ldots, n)$, die an den Zeitpunkten t_j $(j = 1, \ldots, n)$ erhoben werden, geschätzt. Die Modellantwort der Messungen kann durch Auswertung von Meßfunktionen h_i berechnet werden. Bei der Modellierung chemischer Systeme kann es sich bei diesen Meßdaten z.B. um Massenprozente oder Molzahlen der an der Reaktion beteiligten Spezies handeln. Die Meßfunktionen h_i sind somit Funktionen, die von den Zustandsvariablen $x := (y, z)$, den Parametern p und den Steuergrößen q abhängen. Wir nehmen im folgenden an, daß die Meßdaten mit einem additiven Meßfehler ε_{ij} behaftet sind. Es gilt somit:

$$\eta_{ij} = h_i\left(x\left(t_j\right), p, q\right) + \varepsilon_{ij}. \tag{1.29}$$

Im folgenden wird nach BAUER ET AL. (1998b) von den Voraussetzungen ausgegangen, daß die ε_{ij} normalverteilt, der Erwartungswert $\mu = 0$ (also kein systematischer Meßfehler vorliegt) und die Standardabweichungen σ_{ij} gegeben sind, also die Meßfehler $\mathcal{N}\left(0, \Sigma^2\right)$-verteilt (mit $\Sigma = \mathrm{diag}\left(\sigma_{ij}\right)$) sind. Die Aufgabe der Parameterschätzung liegt nun darin, die Parameter so zu bestimmen, daß die Modellantwort die Meßdaten möglichst gut approximiert. Als Maß für die Güte der Schätzung wird hier das gewichtete Least-Squares-Funktional gewählt:

$$\min_{x,p} \sum_{i,j} \frac{\left(\eta_{ij} - h_i\left(x(t_j), p, q\right)\right)^2}{\sigma_{ij}^2}. \tag{1.30}$$

Die Zustandsvariablen $x : \mathbb{R}^n \longrightarrow \mathbb{R}$ erfüllen dabei die DAE-Modellgleichungen

$$\begin{aligned} \dot{y}(t) &= f(t, y(t), z(t), p, q) \\ 0 &= g(t, y(t), z(t), p, q). \end{aligned}$$

Da die Variablen des Optimierungsproblems (1.30), die Parameter p und die unendlichdimensionale Lösungstrajektorie x des DAE-Systems, unendlichdimensional sind, parametrisieren wir die Lösung x der DAE durch den endlichdimensionalen Vektor $s \in \mathbb{R}^{n_s}$. Der Vektor s kann dabei z.B. aus den Anfangswerten des DAE-Systems (1.31) bestehen. Die Lösung der DAE kann dann in Abhängigkeit der Variablen s, p und q dargestellt werden. Wir bezeichnen dies mit

$$x := \hat{x}(s, p, q), \tag{1.31}$$

dem Lösungsoperator der DAE.

Setzt man nun

$$r_1 := \begin{pmatrix} r_{11} \\ \vdots \\ r_{1n} \\ r_{21} \\ \vdots \\ r_{mn} \end{pmatrix}, \eta := \begin{pmatrix} \eta_{11} \\ \vdots \\ \eta_{1n} \\ \eta_{21} \\ \vdots \\ \eta_{mn} \end{pmatrix}, h := \begin{pmatrix} h_1(\hat{x}(t_1, s, p, q), p, q) \\ \vdots \\ h_1(\hat{x}(t_n, s, p, q), p, q) \\ h_2(\hat{x}(t_1, s, p, q), p, q) \\ \vdots \\ h_m(\hat{x}(t_n, s, p, q), p, q) \end{pmatrix},$$

so gilt folgende Darstellung des Residuenvektors r_1:

$$r_1 = \Sigma^{-1} \cdot (\eta - h)\,. \tag{1.32}$$

Man erhält somit ein endlichdimensionales Optimierungsproblem der Form

$$\begin{aligned} \min_{s,p} \quad & \sum_{i,j} \frac{\eta_{ij} - h_i\left(\hat{x}\left(t_j, s, p, q\right), p, q\right)}{\sigma_{ij}^2} \\ 0 \;&=\; g(t_0, s, p, q) \\ r_2\left(\hat{x}(t_0, s, p, q), \ldots, \hat{x}(t_m, s, p, q), s, p, q\right) &= 0 \end{aligned} \tag{1.33}$$

wobei r_2 zusätzliche Beschränkungen an die Parametrisierung sind.

Sind s beispielsweise die Anfangswerte der Zustandsvariablen, so beinhaltet r_2 die Bedingung $x(t_0) = s$. Es sind auch allgemeinere Ansätze der Parametrisierung möglich, die z.B. aus einem Randwert-Ansatz (siehe BOCK (1985)) resultieren können.

Ausführliche Beschreibungen und theoretische Aussagen zur Lösung dieses Parameterschätzproblems findet man in BOCK (1985) und SCHLÖDER (1988). Die numerische Lösung des Parameterschätzproblems (1.33) erfolgt mit einem Randwertproblem-Optimierungsansatz, implementiert in dem Programm-Paket PARFIT.

Da nach Voraussetzung die Meßwerte η_{ij} mit einem Streuungsfehler behaftet und somit Zufallsvariablen sind, gilt dies auch für die geschätzten Parameter $(\tilde{p}, \tilde{s})$. Die Streuung der Parameter wird durch die sogenannte Kovarianzmatrix $C(\tilde{p}, \tilde{s})$ wiedergegeben. Eine Näherung dafür ist nach BOCK (1985) gegeben durch:

$$C(\tilde{p}, \tilde{s}) = J^{+} \begin{pmatrix} I_{n_p} & 0 \\ 0 & 0 \end{pmatrix} J^{+^T} \tag{1.34}$$

Dabei bezeichnet J^{+} die verallgemeinerte Inverse der Jacobi-Matrix J, ausgewertet an der Stelle $(\tilde{p}, \tilde{s})$ und n_p die Anzahl der Parameter.

$$\begin{aligned} J := \begin{pmatrix} J_1 \\ J_2 \end{pmatrix} \;&=\; \begin{pmatrix} \dfrac{\partial r_1}{\partial(p,s)} \\ \dfrac{\partial r_2}{\partial(p,s)} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial}{\partial(p,s)} \Sigma^{-1}\left(\eta - h\right) \\ \dfrac{\partial r_2}{\partial(p,s)} \end{pmatrix} \\ &= \begin{pmatrix} \Sigma^{-1}\left(-\dfrac{\partial h}{\partial \hat{x}} \dfrac{\partial \hat{x}}{\partial(p,s)} - \dfrac{\partial h}{\partial(p,s)}\right) \\ \dfrac{\partial r_2}{\partial(p,s)} \end{pmatrix} \end{aligned}$$

wobei sich J^{+} durch

$$J^{+} = \begin{pmatrix} I & 0 \end{pmatrix} \begin{pmatrix} J_1^T J_1 & J_2^T \\ J_2 & 0 \end{pmatrix}^{-1} \begin{pmatrix} J_1^T & 0 \\ 0 & I \end{pmatrix} \tag{1.35}$$

berechnet.

1.3.3 Versuchsplanungsprobleme

Mit Hilfe der optimalen Versuchsplanung soll eine optimale Auslegung der durchzuführenden Experimente und Auswahl der Messungen erfolgen, so daß die Güte der aus den in den Experimenten ermittelten Meßdaten geschätzten Parameter maximal ist. Deshalb wird das betrachtete Parameterschätzproblem um Gewichte w_{ij} erweitert, die eine Selektion von Messungen bei einer geforderten Maximalzahl von Messungen beschreiben Jeder Summand r_{ij} des Least-Squares-Funktionals (1.30) wird mit einem Gewicht w_{ij} multipliziert. Dabei gibt das Gewicht w_{ij} an, ob eine Messung durchgeführt wird ($w_{ij} = 1$) oder nicht ($w_{ij} = 0$). Die obere Blockmatrix J_1 der Jacobi-Matrix J aus Gleichung (1.3.2) hat mit Gleichung (1.32) dadurch folgendes Aussehen:

$$J_1 := \left(W \cdot \frac{\partial r_1}{\partial (p,s)} \right) \text{ mit } W := \operatorname{diag}\left(\sqrt{w_1}, \ldots, \sqrt{w_n}\right) \tag{1.36}$$

Die klassischen Optimalitätskriterien

Um die Optimalität von Versuchsplänen beurteilen zu können, müssen geeignete Kriterien gefunden werden. Die Kovarianzmatrix C bestimmt nach BOCK (1985) das $100 \cdot (1-\alpha)\%$ Konfidenzellipsoid G_L mit $(0 \leq \alpha \leq 1)$ und

$$G_L\left(\alpha; \tilde{p}, \tilde{s}, q, w\right) = \left\{ (p,s) : \begin{pmatrix} p-\tilde{p} \\ s-\tilde{s} \end{pmatrix} C(\tilde{p}, \tilde{s}, q, w) \begin{pmatrix} p-\tilde{p} \\ s-\tilde{s} \end{pmatrix}^T \leq \gamma^2(\alpha) \right\}, \tag{1.37}$$

das eine Approximation für die (nichtlineare) $100 \cdot (1-\alpha)\%$ Konfidenzregion der geschätzten Parameter $\tilde{s}$ ist. Dabei bedeutet $\gamma(\alpha)$ das Quantil

$$\gamma^2(\alpha) = \chi^2_{m-n_2}(1-\alpha) \tag{1.38}$$

der χ^2-Verteilung, wobei m die Dimension von (p,s) und n_2 die Anzahl der linear unabhängigen Gleichungsnebenbedigungen beschreibt. Um den statistischen Fehler der Parameterschätzung zu minimieren, wird eine Funktion ϕ auf der Kovarianzmatrix C minimiert. Diese Funktion wird auch als *Optimalitätskriterium* bezeichnet. Mögliche Kriterien mit ihren geometrischen Bedeutungen sind nach PUKELSHEIM (1993):

- das *Durchschnittliche-Varianz-Kriterium* (A-Optimalität)
  $\phi_A = \frac{1}{n}\text{Spur}\left(C\left(s,p,q,w\right)\right) = \frac{1}{n}\sum_{i=1}^{n} C_{ii}\left(s,p,q,w\right)$, die durchschnittliche Länge der Halbachsen des Konfidenzellipsoids,

- das *Determinanten-Kriterium* (D-Optimalität)
  $\phi_D = \left(\det C\left(s,q,w\right)\right)^{\frac{1}{n}}$, das Volumen des Konfidenzellipsoids,

- das *Größter-Eigenwert-Kriterium* (E-Optimalität)
  $\phi_E = \lambda_{max} C(s,p,q,w)$ mit $\lambda_{max} = \max\{\lambda \mid \lambda \text{ ist größter Eigenwert von } C\}$, die längste Halbachse des Konfidenzellipsoids.

Formulierung des Optimal-Steuerungsproblems

In der Versuchsplanung chemischer Systeme lassen sich folgende Größen bei der Optimierung einstellen:

- Steuergrößen q, wie etwa die Anfangsmolzahl von Spezies oder die Anfangstemperatur im Reaktor.
- Steuerfunktionen $u(t)$, wie z.B. Zulaufprofile oder Temperaturprofile von Heiz- und Kühlelementen. Die Steuerfunktionen u werden durch endlich viele Variablen parametrisiert. Diese Variablen nehmen wir zum Vektor q hinzu (siehe Abschnitt 3.1.3).
- Die Auswahl der Messungen, der Meßverfahren und der Meßzeitpunkte, an denen die Messungen durchgeführt werden. Dies wird durch die Gewichte w realisiert.

Zusätzlich können noch Beschränkungen an die Zustandsvariablen x und an die Optimierungsvariablen auftreten. Als praktische Beispiele bei der Modellierung eines chemischen Reaktors können hier z.B. maximale oder minimale Temperaturen im Reaktor oder obere Schranken für Zulaufraten angeführt werden.

Man erhält dadurch ein zustandsbeschränktes nichtlineares Optimal-Steuerungsproblem, das als Nebenbedingung ein DAE-System erfüllt.

$$\begin{aligned} &\min_{s,q,w} \quad \phi\left(C\left(\hat{x}\left(s,p,q\right),p,q,w\right)\right) \\ &c(\hat{x}(t,s,p,q),q,w) \left\{ \begin{array}{c} = \\ \geq \end{array} \right\} 0 \\ &d(\hat{x}(t_0,s,p,q,w),\ldots,\hat{x}(t_n,s,p,q),p,q,w) \left\{ \begin{array}{c} = \\ \geq \end{array} \right\} 0 \end{aligned} \tag{1.39}$$

mit den Steuer- und Zustandsbeschränkungen c und Innere-Punkte-Bedingungen d.

Wie oben erfüllen dabei die Zustandsvariablen $x = (y, z) = \hat{x}(s, p, q)$ das DAE-System

$$\begin{aligned} \dot{y} &= f(t,x,p,q) \\ 0 &= g(t,x,p,q). \end{aligned}$$

Die Lösung x ist wieder durch die Variablen s parametrisiert. Insbesondere muß die Konsistenzbedingung

$$g(t_0, y(t_0), z(t_0), p, q) = 0$$

gelten. Wir nehmen auch hier an, daß das DAE-System die Index-1 Annahme erfüllt, d.h. die Matrix $\partial g/\partial z$ muß vollrangig sein.

1.3.4 Auftretende Ableitungen bei der Optimierung

Um die Lösung des Optimal-Steuerungsproblems (1.39) mit Hilfe eines SQP-Verfahrens zu bestimmen, benötigt man in jedem Iterationsschritt den Gradienten der Zielfunktion

$\nabla\phi$ und der Nebenbedingungen. Für die konkrete Optimierung der Beispiele aus Kapitel 5 wird das SQP-Verfahren SNOPT (siehe GILL ET AL. (1998)) verwendet.

Die Berechnung der Ableitungen der Zielfunktion $\phi = \phi\left(C\left(J\left(x\left(q\right), q, w\right)\right)\right)$ nach den Variablen des Optimierungsproblems, den Steuergrößen q sowie den Meßgewichten w bedarf der Anwendung der Kettenregel.[4]

Ableitung der Nebenbedingungen und der Zielfunktion nach den Versuchsplanungsgrößen

Man erhält die Ableitung der Nebenbedingungen c, d und g nach den Optimierungsvariablen v mit $v = (s, q, w)$ durch Anwendung der Kettenregel:

$$\frac{d(c,d,g)}{dv} = \frac{\partial(c,d,g)}{\partial x} \cdot \frac{\partial x}{\partial v} + \frac{\partial(c,d,g)}{\partial v}. \tag{1.40}$$

Die Ableitungen der Nebenbedingungen nach x und v werden durch den in Kapitel 4.1 beschriebenen Ableitungsgenerator berechnet. Die Ableitung der Lösungstrajektorie $x(t)$ nach den Variablen s und q wird in Abschnitt 1.3.5 beschrieben. Da die Variable $x(t)$ nicht von w abhängt, ist die Ableitung von x nach den Gewichten trivialerweise gleich Null.

Nun werden die Ableitungsformeln für die Berechnung der Ableitungen der Zielfunktion nach den Optimierungsvariablen $v = (s, q, w)$ aufgestellt. Da in der Praxis meistens nur Richtungsableitungen benötigt werden, wird im folgenden nur $\frac{d\phi}{dv}\Delta v$ mit einer vorgegebenen Richtung Δv berechnet:

$$\Delta\phi := \frac{d\phi}{dv}\Delta v = \frac{d\phi}{dC}\frac{dC}{dJ}\frac{dJ}{dv}\Delta v. \tag{1.41}$$

In den folgenden Abschnitten wird die Berechnung der Terme

$$\Delta J := \frac{dJ}{dv}\Delta c, \quad \Delta C := \frac{dC}{dJ}\Delta J, \quad \Delta\phi = \frac{d\phi}{dC}\Delta C$$

beschrieben.

Ableitung der Gütekriterien nach der Kovarianzmatrix C

Die Ableitungen des Zielfunktionals nach der Kovarianzmatrix C ergeben sich nach PUKELSHEIM (1993) aus den folgenden Differenzierbarkeitsaussagen:

A-Kriterium:
Für die Ableitung der Zielfunktion ϕ_A im Falle des Durchschnittliche-Varianz-Kriteriums gilt:

$$\Delta\phi_A = \frac{\partial\phi_A}{\partial C}\Delta C = \frac{\partial}{\partial C}\left(\frac{1}{n}\mathrm{Spur}\left(C\right)\right)\Delta C = \frac{1}{n}\cdot\mathrm{Spur}\left(\Delta C\right) \tag{1.42}$$

[4] Der Einfachkeit halber schreiben wir im folgenden x für $\hat{x}$.

Beweis. Mit $\phi_A = \frac{1}{n} \cdot \text{Spur}\,(C) = \frac{1}{n} \cdot \sum_{k=1}^{n} c_{kk}$ gilt:

$$\begin{aligned} \Delta\phi_A &= \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial \phi_A}{\partial c_{ij}} \Delta c_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial}{\partial c_{ij}} \left(\frac{1}{n} \cdot \sum_{k=1}^{n} c_{kk} \right) \Delta c_{ij} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} \left(\frac{1}{n} \cdot \sum_{k=1}^{n} \frac{\partial c_{kk}}{\partial c_{ij}} \Delta c_{ij} \right) = \frac{1}{n} \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} \delta_{ij} \cdot \Delta c_{ij} \\ &= \frac{1}{n} \cdot \sum_{i=1}^{n} \Delta c_{ii} = \frac{1}{n} \cdot \text{Spur}\,(\Delta C) \end{aligned} \tag{1.43}$$

■

D-Kriterium:
Für die Ableitung der Zielfunktion ϕ_D im Falle des Determinanten-Kriteriums gilt:

$$\Delta\phi_D = \frac{\partial \phi_D}{\partial C} \Delta C = \frac{\partial}{\partial C} \left((\det C)^{\frac{1}{n}} \right) \Delta C = \frac{1}{n} \cdot (\det C)^{\frac{1}{n}} \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} \left(C^{-1} \right)_{ij} \Delta c_{ij} \tag{1.44}$$

Beweis. Nach dem Entwicklungssatz von Laplace gilt:

$$\det C = \sum_{j=1}^{n} (-1)^{i+j}\, c_{ij} \det C_{ij} \quad \textit{(Entwicklung nach der i-ten Zeile)}, \tag{1.45}$$

wobei $C_{ij} \in \mathbb{R}^{(n-1)\times(n-1)}$ aus $C \in \mathbb{R}^{n\times n}$ durch Streichen der i-ten Zeile und j-ten Spalte entsteht. Die zu $C \in \mathbb{R}^{n\times n}$ komplementäre Matrix $\tilde{C} \in \mathbb{R}^{n\times n}$ wird definiert durch

$$\tilde{c}_{ij} := (-1)^{i+j} \det C_{ji} \tag{1.46}$$

Damit folgt für $i, k \in \{1, \ldots, n\}$:

$$\sum_{j=1}^{n} \tilde{c}_{ij} c_{jk} = \sum_{j=1}^{n} (-1)^{i+j} \det C_{ji}\; c_{jk} = \delta_{ik} \cdot \det C \tag{1.47}$$

und

$$\sum_{j=1}^{n} c_{ij} \tilde{c}_{jk} = \sum_{j=1}^{n} c_{ij} (-1)^{i+j} \det C_{kj} = \delta_{ik} \cdot \det C. \tag{1.48}$$

Es gilt daher:

$$\tilde{C}C = C\tilde{C} = I_n \det C = (\det C)\, I_n\ . \tag{1.49}$$

$D = C\tilde{C}$ ist eine Diagonalmatrix mit Werten $d_{ii} = \det C$ auf der Hauptdiagonalen. Dadurch gilt für $i \in \{1, \ldots, n\}$:

$$d_{ii} = \det C = \sum_{j=1}^{n} c_{ij} \tilde{c}_{ji} \tag{1.50}$$

Da C nach Voraussetzung positiv definit und somit invertierbar ist, gilt nach Gleichung (1.49):

$$\tilde{C} = (\det C)\, C^{-1} \tag{1.51}$$

Die Gleichungen (1.48) und (1.50) ergeben unter Verwendung der Symmetrie von C:

$$\frac{\partial (\det C)}{\partial c_{ij}} = \frac{\partial}{\partial c_{ij}} \sum_{k=1}^{n} c_{ik}\tilde{c}_{ki} = \tilde{c}_{ji} = \det C \left(C^{-1}\right)_{ji} = \det C \left(C^{-1}\right)_{ij} \tag{1.52}$$

Mit $\phi_D = (\det C)^{\frac{1}{n}}$ gilt mit Gleichung (1.52):

$$\begin{aligned}
\Delta\phi_D = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial \phi_D}{\partial c_{ij}} \Delta c_{ij} &= \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial \left((\det C)^{\frac{1}{n}}\right)}{\partial c_{ij}} \Delta c_{ij} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \left(\frac{1}{n} \cdot (\det C)^{\left(\frac{1}{n}-1\right)} \cdot \frac{\partial (\det C)}{\partial c_{ij}} \Delta c_{ij} \right) \\
&= \frac{1}{n} \cdot (\det C)^{\frac{1}{n}} \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} (\det C)^{-1} \cdot \frac{\partial (\det C)}{\partial c_{ij}} \Delta c_{ij} \qquad (1.53) \\
&= \frac{1}{n} \cdot (\det C)^{\frac{1}{n}} \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} (\det C)^{-1} \det C \left(C^{-1}\right)_{ij} \Delta c_{ij} \\
&= \frac{1}{n} \cdot (\det C)^{\frac{1}{n}} \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} \left(C^{-1}\right)_{ij} \Delta c_{ij}
\end{aligned}$$

■

E-Kriterium:

Für die Ableitung der Zielfunktion ϕ_E im Falle des Größter-Eigenwert-Kriteriums gilt:

$$\Delta\phi_E = \frac{\partial \phi_E}{\partial C} \Delta C = \frac{\partial (\lambda_{max}(C))}{\partial C} \Delta C = z^T \Delta C z\,, \tag{1.54}$$

wobei z der auf 1 normierte Eigenvektor von C zum einfachen maximalen Eigenwert λ_{max} ist. Um dies zu zeigen, wird zunächst folgender Satz bewiesen (vgl. PUKELSHEIM (1993)):

Satz 1.3.1 *Sei $C \in \mathbb{R}^{n\times n}$ symmetrisch, $\lambda(C)$ sei einfacher Eigenwert von C und $z \in \mathbb{R}^n$ sei der bis auf das Vorzeichen eindeutige auf Eins normierte Eigenvektor zu $\lambda(C)$. Dann gilt:*

$$\frac{\partial \lambda(C)}{\partial C} := \left(\frac{\partial \lambda(C)}{\partial c_{ij}} \right)_{1\le i,j\le n} = z^T z.$$

Beweis. Zuerst werden die Eigenwerte von $C \in \mathbb{R}^{n\times n}$ der Größe nach geordnet, dabei sei $\lambda(C)$ ein eindeutiger Eigenwert von C:

$$\lambda_1(C) \le \cdots \le \lambda_{k-1}(C) < \lambda_k(C) = \lambda(C) \le \lambda_{k+1}(C) \le \cdots \le \lambda_n(C). \tag{1.55}$$

Dann werden Eigenvektoren $z_1, \ldots, z_{k-1}, z_k = z, z_{k+1}, \ldots, z_n$ von $C \in \mathbb{R}^{n\times n}$ gewählt, die eine Orthonormalbasis des $\mathbb{R}^n$ bilden.

Definiere $U \in \mathbb{R}^{n\times n}$ mit:

$$U := (z, Z) := (z, z_1, \ldots, z_{k-1}, z_{k+1}, \ldots, z_n) \tag{1.56}$$

und $E_{ij} \in \mathbb{R}^{n\times n}$ durch Verwendung des dyadischen Produkts der Einheitsvektoren $e_i, e_j \in \mathbb{R}^n$ durch:

$$E_{ij} := \frac{1}{2}\left(e_i e_j^T + e_j e_i^T\right) \tag{1.57}$$

Seien weiter

$$C(\varepsilon) := C + \varepsilon E_{ij}, \quad \delta(\varepsilon) := \lambda_k(C(\varepsilon)) - \lambda_k(C), \quad \tilde{C}(\varepsilon) := C(\varepsilon) - \lambda_k(C(\varepsilon))I_n. \tag{1.58}$$

Da C nach Voraussetzung symmetrisch ist, ist auch $C(\varepsilon) \in \mathbb{R}^{n\times n}$ symmetrisch, und die Eigenwerte von $C(\varepsilon)$ sind reell. Der Eigenwert $\lambda(C(\varepsilon)) := \lambda_k(C(\varepsilon))$ bleibt auch für kleine Störungen $\varepsilon \in \mathbb{R}$ einfach und verändert die Reihenfolge der Eigenwerte nicht.

Die Matrix $U \in \mathbb{R}^{n\times n}$ ist unitär, und es gilt:

$$\begin{aligned} 0 &= \det(\tilde{C}(\varepsilon)) = \det(U^T\tilde{C}(\varepsilon)U) \\ &= \det\begin{pmatrix} z^T\tilde{C}(\varepsilon)z & z^T\tilde{C}(\varepsilon)Z \\ Z^T\tilde{C}(\varepsilon)z & Z^T\tilde{C}(\varepsilon)Z \end{pmatrix} \end{aligned} \tag{1.59}$$

$\lambda(C)$ ist einfacher Eigenwert von $C \in \mathbb{R}^{n\times n}$, und es folgt mit der Definition von $Z \in \mathbb{R}^{n\times(n-1)}$, daß

$$\begin{aligned} Z^T\tilde{C}(0)Z &= Z^T(C - \lambda_k(C)I_n)Z \\ &= \operatorname{diag}(\lambda_1(C) - \lambda_k(C), \ldots, \lambda_{k-1}(C) - \lambda_k(C), \\ &\qquad \lambda_{k+1}(C) - \lambda_k(C), \ldots, \lambda_n(C) - \lambda_k(C)) \in \mathbb{R}^{(n-1)\times(n-1)} \end{aligned} \tag{1.60}$$

invertierbar ist.

Somit ist auch für kleine Störungen $\varepsilon \in \mathbb{R}$ das Matrix-Produkt $Z^T\tilde{C}(\varepsilon)Z \in \mathbb{R}^{(n-1)\times(n-1)}$ invertierbar, und unter Verwendung von

$$\begin{pmatrix} A & B \\ B^T & C \end{pmatrix}\begin{pmatrix} I & 0 \\ -C^{-1}B^T & I \end{pmatrix} = \begin{pmatrix} A - BC^{-1}B^T & B \\ 0 & C \end{pmatrix} \tag{1.61}$$

ergibt sich

$$\begin{aligned} 0 &= \det(\tilde{C}(\varepsilon)) \\ &= \left(z^T\tilde{C}(\varepsilon)z - z^T\tilde{C}(\varepsilon)Z\left(Z^T\tilde{C}(\varepsilon)Z\right)^{-1}Z^T\tilde{C}(\varepsilon)z\right)\det\left(Z^T\tilde{C}(\varepsilon)Z\right) \\ &= z^T\tilde{C}(\varepsilon)z - z^T\tilde{C}(\varepsilon)Z\left(Z^T\tilde{C}(\varepsilon)Z\right)^{-1}Z^T\tilde{C}(\varepsilon)z. \end{aligned} \tag{1.62}$$

Einsetzen der Definition von $\tilde{C}(\varepsilon)$ ergibt:

$$\begin{gathered} \tilde{C}(\varepsilon)z = \varepsilon E_{ij}z - \delta(\varepsilon)z, \quad Z^T\tilde{C}(\varepsilon)z = \varepsilon Z^T E_{ij}z, \\ \varepsilon z^T E_{ij}z - \delta(\varepsilon) - \varepsilon^2\left(z^T E_{ij}Z\left(Z^T\tilde{C}(\varepsilon)Z\right)^{-1}Z^T E_{ij}z\right) = 0 \end{gathered} \tag{1.63}$$

Es gilt dann:

$$\delta(\varepsilon) = \varepsilon z^T E_{ij} z - \varepsilon^2 \left(z^T E_{ij} Z \left(Z^T \tilde{C}(\varepsilon) Z \right)^{-1} Z^T E_{ij} z \right) \tag{1.64}$$

Für die Berechnung der Ableitung ergibt sich:

$$\begin{aligned} \lim_{\varepsilon \to 0} \frac{\lambda_k\left(C(\varepsilon)\right) - \lambda_k\left(C\right)}{\varepsilon} &= \lim_{\varepsilon \to 0} \frac{\delta(\varepsilon)}{\varepsilon} \\ &= \lim_{\varepsilon \to 0} \left(z^T E_{ij} z - \varepsilon \left(z^T E_{ij} Z \left(Z^T \tilde{C}(\varepsilon) Z \right)^{-1} Z^T E_{ij} z \right) \right) \\ &= z^T E_{ij} z \end{aligned} \tag{1.65}$$

■

Die Anwendung von Satz 1.3.1 auf den größten Eigenwert λ_{max} liefert folglich für eine beliebige Richtungsmatrix ΔC die Gleichung (1.54).

Ableitung der Kovarianzmatrix nach der Jacobi-Matrix

Verwendet wird im folgenden die Darstellung der verallgemeinerten Inverse J^+ (Gleichung 1.35) der Jacobi-Matrix des Parameterschätzproblems (1.30). Die Kovarianzmatrix hat nach BAUER ET AL. (1998b) folgende Gestalt:

$$\begin{aligned} C &= J^+ \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} J^{+^T} \\ &= \begin{pmatrix} I & 0 \end{pmatrix} \begin{pmatrix} J_1^T J_1 & J_2^T \\ J_2 & 0 \end{pmatrix}^{-1} \begin{pmatrix} J_1^T & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} \\ &\quad \cdot \left(\begin{pmatrix} I & 0 \end{pmatrix} \begin{pmatrix} J_1^T J_1 & J_2^T \\ J_2 & 0 \end{pmatrix}^{-1} \begin{pmatrix} J_1^T & 0 \\ 0 & I \end{pmatrix} \right)^T \\ &= \begin{pmatrix} I & 0 \end{pmatrix} \begin{pmatrix} J_1^T J_1 & J_2^T \\ J_2 & 0 \end{pmatrix}^{-1} \begin{pmatrix} J_1^T & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} J_1 & 0 \\ 0 & I \end{pmatrix} \\ &\quad \cdot \begin{pmatrix} \left(J_1^T J_1\right)^T & J_2^T \\ J_2 & 0 \end{pmatrix}^{-1} \begin{pmatrix} I \\ 0 \end{pmatrix} \\ &= \underbrace{\begin{pmatrix} I & 0 \end{pmatrix}}_{:=I_1} \underbrace{\begin{pmatrix} J_1^T J_1 & J_2^T \\ J_2 & 0 \end{pmatrix}^{-1}}_{:=M} \underbrace{\begin{pmatrix} J_1^T J_1 & 0 \\ 0 & 0 \end{pmatrix}}_{:=S} \begin{pmatrix} J_1^T J_1 & J_2^T \\ J_2 & 0 \end{pmatrix}^{-1} \begin{pmatrix} I \\ 0 \end{pmatrix} \\ &= I_1 \cdot M^{-1} \cdot S \cdot M^{-1} \cdot I_1^{-1} \end{aligned} \tag{1.66}$$

mit $J = \begin{pmatrix} J_1 \\ J_2 \end{pmatrix}$.

Nun werden die folgenden Hilfsgrößen gesetzt durch:

$$\Delta M := \frac{\partial M}{\partial J}\Delta J = \frac{\partial}{\partial J}\begin{pmatrix} J_1^T J_1 & J_2^T \\ J_2 & 0 \end{pmatrix}\Delta J = \begin{pmatrix} \Delta J_1^T J_1 + J_1^T \Delta J_1 & \Delta J_2^T \\ \Delta J_2 & 0 \end{pmatrix} \tag{1.67}$$

und

$$\begin{aligned} \Delta S &:= \frac{\partial S}{\partial J}\Delta J = \frac{\partial}{\partial J}\begin{pmatrix} J_1^T J_1 & 0 \\ 0 & 0 \end{pmatrix}\Delta J = \begin{pmatrix} \Delta J_1^T J_1 + J_1^T \Delta J_1 & 0 \\ 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} \Delta J_1^T J_1 + J_1^T \Delta J_1 \\ 0 \end{pmatrix} \cdot I_1 \end{aligned} \tag{1.68}$$

sowie

$$\Delta J := \begin{pmatrix} \Delta J_1 \\ \Delta J_2 \end{pmatrix}, \quad H := \begin{pmatrix} J_1^T J_1 \\ 0 \end{pmatrix} \tag{1.69}$$

Die Anwendung der Produktregel und des Satzes 1.1.2 ergibt für die Ableitung von C nach J unter Verwendung der Gleichungen (1.67), (1.68) und (1.69):

$$\begin{aligned} \Delta C &= \frac{\partial C}{\partial J}\Delta J = \frac{\partial}{\partial J}\left(I_1 \cdot M^{-1} \cdot S \cdot M^{-1} \cdot I_1^{-1}\right)\Delta J \\ &= -I_1 \cdot \left(M^{-1} \cdot \Delta M \cdot M^{-1}\right) \cdot S \cdot M^{-1} \cdot I_1^T \\ &\quad +I_1 \cdot M^{-1} \cdot \Delta S \cdot M^{-1} \cdot I_1^T \\ &\quad -I_1 \cdot M^{-1} \cdot S \cdot \left(M^{-1} \cdot \Delta M \cdot M^{-1}\right) \cdot I_1^T \\ &= I_1 \cdot M^{-1} \cdot \left(-\left(S \cdot M^{-1} \cdot \Delta M\right)^T + \Delta S - \left(S \cdot M^{-1} \cdot \Delta M\right)\right) M^{-1} \cdot I_1^T \\ &= \left(I_1 \cdot M^{-1}\right) \cdot \left(\left(I_1 \cdot M^{-1}\right) \cdot \left(-\left(H \cdot \left(I_1 \cdot M^{-1}\right) \cdot \Delta M\right)^T + \Delta S \right.\right. \\ &\quad \left.\left. -\left(H \cdot \left(I_1 \cdot M^{-1}\right) \cdot \Delta M\right)\right)^T\right)^T \end{aligned} \tag{1.70}$$

Man erhält jetzt Darstellungen von C und ΔC, die man numerisch sehr effizient berechnen kann. Hierbei wird wiederholt der Term $(I_1 \cdot M^{-1})$ mit verschiedenen Matrizen multipliziert. Durch Anwendung der sogenannten *Nullraummethode* werden diese Ausdrücke ausgewertet. Die hierbei benötigten Zerlegungen der Matrizen J_1 und J_2 werden dabei nur einmal berechnet.

Ableitung der Jacobi-Matrix nach den Versuchsplanungsgrößen

In Abschnitt 1.3.3 wurde die Jacobi-Matrix J_1 als Ableitung des Residuenvektors r_1 nach dem Vektor s definiert.

$$J_1 = \frac{dr_1}{d(p,s)} = -W \cdot \Sigma^{-1} \cdot \frac{dh}{d(p,s)} = -W \cdot \Sigma^{-1} \cdot \left(\frac{\partial h}{\partial x}\frac{\partial x}{\partial (p,s)} + \frac{\partial h}{\partial p}\right) \tag{1.71}$$

Bezeichne $h = h(x(t,p,s,q),p,q)$ im folgenden den Vektor der Meßfunktionsauswertungen. Es ergibt sich für die Ableitung der Jacobi-Matrix J nach Steuergrößen q und Meßgewichten w mit Hilfe der Kettenregel (vgl. BAUER ET AL. (1998b)):

$$\begin{aligned}
\Delta J_1 &:= \frac{dJ_1}{d(q,w)}\Delta(q,w) = \frac{\partial J_1}{\partial x}\frac{\partial x}{\partial q}\Delta q + \frac{\partial J_1}{\partial q}\Delta q + \frac{\partial J_1}{\partial w}\Delta w \\
&= -W\cdot\Sigma^{-1}\cdot\left(\frac{\partial}{\partial x}\left(\frac{\partial h}{\partial x}\frac{\partial x}{\partial(p,s)}+\frac{\partial h}{\partial p}\right)\frac{\partial x}{\partial q}\Delta q + \frac{\partial}{\partial q}\left(\frac{\partial h}{\partial x}\frac{\partial x}{\partial(p,s)}+\frac{\partial h}{\partial p}\right)\Delta q\right) \\
&\quad -\frac{\partial W}{\partial w}\Delta w\cdot\Sigma^{-1}\cdot\left(\frac{\partial h}{\partial x}\frac{\partial x}{\partial(p,s)}+\frac{\partial h}{\partial p}\right) \\
&= \left(\frac{\partial^2 h}{\partial x^2}\frac{\partial x}{\partial q}\Delta q\frac{\partial x}{\partial(p,s)} + \frac{\partial^2 h}{\partial x\partial p}\frac{\partial x}{\partial q}\Delta q + \frac{\partial^2 h}{\partial q\partial x}\Delta q\frac{\partial x}{\partial(p,s)} + \frac{\partial h}{\partial x}\frac{\partial^2 x}{\partial q\partial(p,s)}\Delta q\right. \\
&\quad \left. + \frac{\partial^2 h}{\partial q\partial p}\Delta q\right) - \operatorname{diag}\left(\frac{\Delta w_i}{2\sqrt{w_i}}\right)\cdot\Sigma^{-1}\cdot\left(\frac{\partial h}{\partial x}\frac{\partial x}{\partial(p,s)}+\frac{\partial h}{\partial p}\right)
\end{aligned}$$

Für die Ableitung der Jacobi-Matrix $J_2 = \dfrac{\partial r_2}{\partial s}$ der Nebenbedingungen, die vom Vektor w nicht abhängt, gilt nach der Kettenregel:

$$\Delta J_2 = \frac{\partial J_2}{\partial x}\frac{\partial x}{\partial q}\Delta q + \frac{\partial J_2}{\partial q}\Delta q = \frac{\partial^2 r_2}{\partial x\partial(p,s)}\frac{\partial x}{\partial q}\Delta q + \frac{\partial^2 r_2}{\partial q\partial(p,s)}\Delta q \tag{1.72}$$

1.3.5 Interne Numerische Differentiation

Um die beiden Formeln (1.72) und (1.72) praktisch zu berechnen, muß ein Anfangswertproblem für das DAE-System (1.2) gelöst werden. Sei nun $x(t,s,q)$ eine Lösung der DAE, so werden zusätzlich noch die Ableitungen

$$\frac{\partial x}{\partial(p,s)}, \frac{\partial x}{\partial q} \text{ und } \frac{\partial^2 x}{\partial q\partial(p,s)} \tag{1.73}$$

nach den Parametern bzw. den Steuergrößen dieser Lösung benötigt. Die Zustandsvariablen x sind implizit durch das DAE-System (1.2) definiert. Bei der Modellierung chemischer Reaktionsprozesse treten meistens steife DAE-Systeme auf. Dafür werden implizite Integratoren verwendet. Die Integration des betrachteten Systems erfolgt mit dem am IWR entwickelten Integrator DAESOL. Dabei handelt es sich um ein Mehrschrittverfahren mit rückwärtsgenommenen Differenzen (BDF[5]) mit einer variablen Ordnung und Schrittweite. Näheres dazu findet man in BAUER (1999).

In einem BDF-Verfahren wird die Ableitung $\dot{y}$ von Gleichung (1.2) im $(n+1)$-ten Integrationsschritt durch ein Polynom

$$P_{n+1}\left(y_{n+1-k},\ldots,y_n,y_{n+1}\right) = -\frac{1}{h_{n+1}}\sum_{i=0}^{k}\alpha_i\cdot y_{n+1-i} \tag{1.74}$$

vom Grade k approximiert. Der Faktor h_{n+1} bezeichne die Schrittweite der Integration zum Zeitpunkt t_{n+1}. Diese Diskretisierung wird in das DAE-System im Schritt $n+1$ eingesetzt.

[5] Backward Differentiation Formulae

Man erhält somit das diskretisierte System für die Nominaltrajektorie $x = (y, z)$.

$$\begin{aligned} \alpha_0 \cdot y_{n+1} + \sum_{i=1}^{k} \alpha_i \cdot y_{n+1-i} + h_{n+1} \cdot f(t_{n+1}, x_{n+1}, p, q) &= 0 \\ g(t_{n+1}, x_{n+1}, p, q) &= 0 \end{aligned} \tag{1.75}$$

Die Nominaltrajektorie $x_{n+1} = (y_{n+1}, z_{n+1})$ ist implizit durch das nichtlinerare Gleichungssystem (1.75) gegeben. Man erhält die Lösung (*Korrektor*) durch folgendes Verfahren:

Schreibt man das System (1.75) kurz als $F(x_{n+1}) = 0$, so ergibt sich die Newton-Iteration als

$$\begin{aligned} x^{k+1} &:= x^k + \Delta x^k \\ \Delta x^k &:= -(M^k)^{-1} F(x^k). \end{aligned} \tag{1.76}$$

Als Startwerte für x_{n+1} wird der aus $x_n, \ldots, x_{n-k-1}$ extrapolierte Wert genommen. Dieser Wert wird auch als *Prädiktor* bezeichnet. Die Matrix M^k ist eine Näherung der Jacobi-Matrix $J_{x_{n+1}}$ von $F(x_{n+1})$

$$M^k \approx \frac{\partial F}{\partial x}(x_{n+1}) =: J_{x_{n+1}} \tag{1.77}$$

mit

$$J_{x_{n+1}} = \left(\begin{array}{c|c} \alpha_0 + h_{n+1} \dfrac{\partial}{\partial y} f(t_{n+1}, y_{n+1}, z_{n+1}, p, q) & h_{n+1} \dfrac{\partial}{\partial z} f(t_{n+1}, y_{n+1}, z_{n+1}, p, q) \\ \hline \dfrac{\partial}{\partial y} g(t_{n+1}, y_{n+1}, z_{n+1}, p, q) & \dfrac{\partial}{\partial z} g(t_{n+1}, y_{n+1}, z_{n+1}, p, q) \end{array} \right) \tag{1.78}$$

Da sich die Jacobi-Matrix während der Iteration und von Schritt zu Schritt im allgemeinen nur wenig ändert, wird ihre Neuberechnung durch eine sogenannte *Monitorstrategie* überwacht und nur selten durchgeführt. Mehr über BDF-Verfahren für Index-1 DAEs findet man in Bock et al. (1994).

Um die Ableitung der Nominaltrajektorie des DAE-Systems nach den Parametern bzw. Steuergrößen zu erhalten, gibt es verschiedene Möglichkeiten. Zum einen lassen sich *Numerische Differenzen* verwenden, wobei man die Nominaltrajektorie und die variierte Trajektorie mit gestörten Anfangswerten, Parametern und Steuergrößen lösen muß. In Abschnitt 2.3 wird genauer erläutert, daß hier sehr hohe Fehler auftreten. Falls die Trajektorien mit der Genauigkeit TOL berechnet werden, so werden die korrespondierenden Ableitungen mit Numerischen Differenzen mit der Genauigkeit $\sqrt{TOL}$ berechnet. Dabei treten vor allem bei der Berechnung von zweiten Ableitungen, wie sie in (1.73) benötigt werden, für das Optimierungsverfahren inakzeptable Ungenauigkeiten auf.

Deswegen wird im folgenden eine Variante der in Bock (1985) beschriebenen Technik der *Internen Numerischen Differentiation* (IND) verwendet:

Mit Verwendung der Wronski-Matrix:

$$W_p = \begin{pmatrix} W_p^y \\ W_p^z \end{pmatrix} := \begin{pmatrix} \frac{\partial y}{\partial p} \\ \frac{\partial z}{\partial p} \end{pmatrix} \tag{1.79}$$

ergibt sich für die Ableitung des DAE-Systems (1.2) nach den Parametern p:

$$\begin{aligned} \frac{d}{dp}\dot{y} &= \frac{d}{dp} f(t, y, z, p, q) \\ 0 &= \frac{d}{dp} g(t, y, z, p, q) \end{aligned} \tag{1.80}$$

Die Auswertung von Gleichung (1.80) ergibt die Variations-DAE erster Ordnung nach p:

$$\begin{aligned} \dot{W}_p^y &= \frac{\partial f}{\partial y}\frac{\partial y}{\partial p} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial p} = f_y W_p^y + f_z W_p^z + f_p = f_x W_p + f_p \\ 0 &= \frac{\partial g}{\partial y}\frac{\partial y}{\partial p} + \frac{\partial g}{\partial z}\frac{\partial z}{\partial p} = g_y W_p^y + g_z W_p^z + g_p = g_x W_p + g_p \end{aligned} \tag{1.81}$$

Das diskretisierte System für die Variations-DAE (1.81) lautet:

$$\begin{aligned} \sum_{i=0}^{k} \alpha_i W_{p,n+1-i}^y + h_{n+1} f_x W_{p,n+1} + h_{n+1} f_p &= 0 \\ g_x W_{p,n+1} + g_p &= 0 \end{aligned} \tag{1.82}$$

Andererseits kann das diskretisierte Gleichungssystem (1.75) nach den Parametern p abgeleitet werden:

$$\begin{aligned} \sum_{i=0}^{k} \alpha_i \cdot \frac{\partial y_{n+1-i}}{\partial p} + \frac{\partial h_{n+1}}{\partial p} f(t_{n+1}, x_{n+1}, p, q) & \\ + h_{n+1} \cdot \left(\tfrac{\partial}{\partial x} f(t_{n+1}, x_{n+1}, p, q) \cdot \tfrac{\partial x_{n+1}}{\partial p} + \tfrac{\partial}{\partial p} f(t_{n+1}, x_{n+1}, p, q) \right) &= 0 \\ \frac{\partial}{\partial x} g(t_{n+1}, x_{n+1}, p, q) \cdot \frac{x_{n+1}}{\partial p} + \frac{\partial}{\partial p} g(t_{n+1}, x_{n+1}, p, q) &= 0 \end{aligned} \tag{1.83}$$

Vergleicht man die beiden Gleichungssysteme (1.82) und (1.83), so erkennt man, daß sie bis auf den Term mit $\frac{\partial h}{\partial p}$ identisch sind. Falls für die Nominaltrajektorie x und die Ableitungen W_p das gleiche Diskretisierungsschema verwendet wird, ist dieser Term Null. Damit ergibt sich, daß die Lösung der Variations-DAE erster Ordnung gleich der Ableitung des diskretisierten Gleichungssystems (1.83) ist.

Analog werden die Variations-DAEs für die Wronski-Matrizen $W_{x_0} = \frac{\partial x}{\partial x_0}$ und $W_q = \frac{\partial x}{\partial q}$ zur Berechnung der Nominaltrajektorie nach den Anfangswerten x_0 und den Steuergrößen q aufgestellt und gelöst.

Zur Berechnung der gemischten zweiten Ableitungen wird das obige Vorgehen wiederholt. Die Variations-DAE zweiter Ordnung ergibt sich durch die Differentiation des Gleichungssystems (1.81) nach den Steuergrößen q:

$$\begin{aligned} \dot{W}_{pq}^y &= f_{xx} W_q W_p + f_{xq} W_p + f_x W_{pq} + f_{px} W_q + f_{pq} \\ 0 &= g_{xx} W_p W_q + g_{xq} W_p + g_x W_{pq} + g_{px} W_q + q_{pq} \end{aligned} \tag{1.84}$$

Die Diskretisierung der Variations-DAE (1.84) zweiter Ordnung liefert analog dasselbe Gleichungssystem, wie wenn das Diskretisierungsschema (1.83) nach q abgeleitet wird. Wie oben wird vorausgesetzt, daß das gleiche Diskretisierungsschema wie für die Nominaltrajektorie und die Variations-DAE erster Ordnung verwendet wurde. Um die Wronski-Matrix $W_{x_0q} = \frac{\partial^2 x}{\partial q \partial x_0}$ zu berechnen, wird analog eine Variations-DAE zweiter Ordnung gelöst. Die Lösungen der Variations-DAE erster und zweiter Ordnung entsprechen dann genau den Ableitungen, der durch die BDF-Diskretisierung berechneten Nominaltrajektorien. Das Diskretisierungsschema für die zweiten Ableitungen führt auf die gleiche Matrix M^k wie das Schema für die ersten Ableitungen und der Nominaltrajektorie. Bei der Implementierung ergeben sich daraus große Einsparmöglichkeiten: Durch die Berechnung der Nominaltrajektorie und der Wronski-Matrix W_{n+1}, die implizit durch das gleiche Diskretisierungsschema gegeben sind, muß in jedem BDF-Integrationsschritt die Matrix M^k somit nur einmal berechnet und zerlegt werden.

Kapitel 2

Automatisches Differenzieren

2.1 Einführung

Viele numerische Anwendungen benötigen Jacobi-Matrizen, Hesse-Matrizen sowie Richtungsableitungen vektorwertiger Funktionen mit m Komponenten und n reellen oder komplexen Variablen. Dies gilt insbesondere in der nichtlinearen Optimierung, bei der Lösung von nichtlinearen Differential- und Integralgleichungen sowie bei Sensitivitätsanalysen. Als Beispiel betrachte man das Newton-Verfahren zur Lösung nichtlinearer Gleichungen:

$$0 = F(x) \quad \text{mit} \quad F : D_F \subset I\!R^n \longrightarrow I\!R^n$$

Ausgehend von einem Startwert $x^{(0)} \in D_F$ approximiert das Newton-Verfahren eine mögliche Lösung x^* mit $F(x^*) = 0$ durch das sukzessive Lösen der folgenden Gleichungssysteme:

$$J^{(k)} \cdot \left(x^{(k+1)} - x^{(k)}\right) = -F^{(k)} \quad k = 0, 1, 2, \ldots$$

mit

$$J^{(k)} := \frac{dF}{dx}\left(x^k\right), \quad F^{(k)} := F(x^k)$$

Hierbei bezeichnet $J^{(k)}$ die Jacobi-Matrix von F ausgewertet an der Stelle $x^{(k)}$.

In vielen Fällen sind die Funktionen, die ein Problem beschreiben, durch sequentielle Berechnungsvorschriften (Algorithmen) definiert, in denen viele Zwischenvariablen auftreten. Durch symbolische Elimination der Zwischenvariablen ist es zwar möglich, die m abhängigen Variablen explizit als Ausdrücke in den n unabhängigen Variablen darzustellen, doch das resultierende Ergebnis besteht dann meistens aus vielen unübersichtlichen algebraischer Formeln. Die Differentiation dieser Formeln verschärft das Problem des schnellen Anwachsens der Größe des mathematischen Ausdruckes sowie der wiederholten Berechnung gemeinsamer Teilausdrücke. SPEELPENNING (1980) konnte für Funktionen einer Veränderlichen ($n = 1$) oder skalarwertige Funktionen ($m = 1$) nachweisen, daß das effizienteste Verfahren für die Ableitungsberechnung direkt aus dem Quelltext der Funktionsberechnung erhalten werden kann. Dabei werden die einzelnen Instruktionen des Quelltextes unter Anwendung der Kettenegel 2.2.1 abgeleitet. Dieses Vorgehen wird als *Automatisches Differenzieren* (AD) oder auch als *Computational Differentiation* (CD) bezeichnet. Bemerkenswert ist hierbei die Tatsache, daß der Gradient einer skalarwertigen Funktion

im allgemeinen höchstens für den Faktor 5 der Operationen der Funktion selbst berechnet werden kann, unabhängig von der Anzahl n der unabhängigen Variablen. Folglich kann die zeilenweise Berechnung der Jacobi-Matrix einer vektorwertigen Funktion durch das sogenannte Rückwärtsverfahren in höchstens $5\,m$ mal sovielen Operationen erfolgen wie die Berechnung der zugrundeliegende Vektorfunktion.

Die folgenden Kapitel sollen dabei zeigen, daß für den allgemeinen Fall das alleinige Anwenden einer der beiden in Abschnitt 2.2 beschriebenen Verfahren nicht optimal ist. Da in dieser Arbeit die praktische Implementierung einen großen Platz einnimmt, ergeben sich aus obigen Ansätzen sofort Fragen nach dem Speicherplatzbedarf und dem arithmetischen Aufwand der benötigten Operationen.

2.1.1 Begriffsbestimmung

Unter Automatischem Differenzieren (AD) versteht man den Prozeß, Ableitungen von Funktionen mit Hilfe eines Computers zu berechnen, wobei die Funktionen durch ein Computerprogramm gegeben sind. Die klassischen AD-Implementierungen verwenden das Prinzip der Quelltext-Transformation. Hierbei wird mit Hilfe des Quelltextes, der die Funktion implementiert, ein weiterer Quelltext generiert, der die Ableitungen beschreibt. Man spricht hier vom *Ableiten von Routinen*. Die Ableitungen werden folglich nicht als explizite Funktionen generiert, sondern durch ein generiertes Programm berechnet. Die Tatsache, daß die Funktionen durch ein Computerprogramm gegeben sind, führt dazu, daß man mit AD nicht nur Funktionen ableiten kann, die als Formeln oder Aneinanderreihung von Formeln vorliegen, sondern sich auch allgemeiner definierte Funktionen differenzieren lassen. Diese Funktionen können beliebige Syntaxelemente höherer Programmiersprachen wie z.B. Verzweigungen (`if-else` Konstrukte) oder Iterationen (`for`- bzw. `while`-Schleifen) besitzen.

2.1.2 Automatische Berechnung eines mathematischen Ausdrucks

Genauso wie die Computeralgebra gründet sich das AD auf der Tatsache, daß sich die meisten in der Praxis auftretenden Funktionen als Komposition einer geringen Menge von Basisfunktionen, auch Elementarfunktionen oder Bibliotheksfunktionen genannt, darstellen lassen. Diese Basisfunktionen können unäre oder binäre arithmetische Operatoren (z.B. '+' und '$*$') und transzendente Funktionen (z.B. sin und arctan) sein. Dies trifft insbesondere für diejenigen Funktionen zu, die durch eine geschlossenen Funktion repräsentiert werden können oder mit einem Computerprogramm berechnet werden, das in einer höheren Programmiersprache (z.B. C, C++ oder Fortran77) geschrieben ist.

Im folgenden betrachten wir Funktionen F mit dem Vektor x der *unabhängigen* Variablen sowie dem Vektor y der *abhängigen* Variablen:

$$F : \begin{cases} \mathcal{D} \subset I\!R^n \longrightarrow I\!R^m \\ x = (x_1, x_2, \ldots, x_n) \in \mathcal{D} \longmapsto y = (y_1, y_2, \ldots, y_m) \end{cases} \tag{2.1}$$

Zunächst benötigen wir die Definition nach KEDEM (1980) einer *faktorisierbaren* Funktion:

Definition 2.1.1 *Sei $\mathcal{F}$ eine endliche Menge a priori definierter Basisfunktionen. Eine Funktion $f = f(x_1, x_2, \ldots, x_n) : I\!R^n \to I\!R$ nennt man faktorisierbar über $\mathcal{F}$, falls eine endliche Reihe von Funktionen $\phi_1, \phi_2, \ldots, \phi_l$ existiert mit:*

1. $\phi_i = x_i \quad \forall\ 1 \leq i \leq n$

2. $f = \phi_l$

3. *$\forall i$ mit $n < i \leq l$ gilt: ϕ_i ist entweder eine Komposition einer Basisfunktion $f^* \in \mathcal{F}$ mit einer Funktion ϕ_j $(j < i)$, oder ϕ_i ist eine konstante Funktion.*

Jede faktorisierbare Funktion läßt sich somit durch ein sequentielles Programm unter Verwendung von l $(l \geq n)$ Zwischenvariablen $v_1, v_2, \ldots, v_l$, der folgenden Form berechnen:

for $i := n + 1$ **up to** l **do**
    $v_i := \phi_i(U_i)$
od
$f := v_l$

Dabei wird für jeden Zwischenwert v_i $(1 \leq i \leq l)$ die Menge U_i unter Verwendung der Präzedenz-Relation '$\prec$' gesetzt durch:

$$U_i := \{\ v_j \mid j \ \prec\ i\} \tag{2.2}$$

Die in den folgenden Abschnitten verwendete Relation '$\prec$' hat dabei folgende Bedeutung:

$$j \ \prec\ i \Longleftrightarrow \text{ Knoten } v_i \text{ hängt direkt von Knoten } v_j \text{ ab .}$$

Die Zwischenwerte können dabei nach GRIEWANK & CHRISTIANSEN (1998) so geordnet werden, daß gilt:

v_{i-n}	$=$	x_i	$1 \leq i \leq n$
v_i	$=$	$\phi_i(U_i)$	$1 \leq i \leq l$
y_{m-i}	$=$	v_{l-i}	$m > i \geq 0$

Tabelle 2.1: Anordnung der Variablen bei der Funktionsberechnung

Diese Darstellung wird bei der algorithmischen Beschreibung der Verfahren des AD in Abschnitt 2.2 verwendet. Zur Vereinfachung dieser Darstellung kann man (o.B.d.A.) annehmen, daß alle Basisfunktionen ϕ_i skalarwertig sind und eine kleine Anzahl von Argumenten haben. Die Funktionswerte dieser Basisfunktionen werden dabei wieder als Argumente anderer Basisfunktionen verwendet und werden im folgenden als Zwischengrößen bezeichnet. Man kann im allgemeinen erwarten, daß die Anzahl m der abhängigen Variablen einer als Komposition darstellbaren Funktion viel kleiner ist als die Anzahl z der Zwischenvariablen. Daraus läßt sich nach GRIEWANK & UTKE (1995) folgern, daß die Komplexität für eine zusammengesetzte Funktion auf einem seriellen Computer ungefähr proportional zu z ist.

2.1.3 Definitionen aus der Graphentheorie

Für das Verständnis der Abschnitte 2.1.4 bis 2.2.7 benötigt man zunächst einige Definitionen aus der Graphentheorie, die aus KNUTH (1973) entnommen werden.

Definition 2.1.2 (Graph) *Ein Graph $G = (V, E)$ besteht aus einer endlichen Menge $V \neq \emptyset$ von Punkten (Knoten) sowie einer Menge $E \in V \times V$ von Linien (Kanten). Bestehen die Kanten aus geordneten Paaren (v_1, v_2) von Knoten, so spricht man von einem gerichteten Graphen, andernfalls von einem ungerichteten Graphen. Zwei Knoten v, w werden als adjazent bezeichnet, falls eine Kante e existiert mit $e = (v, w)$.*

Definition 2.1.3 (Pfad, Zyklus) *Sind v und v' Knoten und gilt $n \geq 0$, so nennt man $(v_0, v_1, \ldots, v_n)$ einen Pfad der Länge n falls, $v = v_0$, v_k adjazent zu v_{k+1} für $0 \leq k \leq n$ und $v_n = v'$. Unter einem Zyklus (Kreis) versteht man einen Pfad der Länge $n > 1$ der Form $(v = v_0, v_1, \ldots, v_n = v)$. Existiert in einem Graphen kein Zyklus, so spricht man von einem azyklischen Graphen, sonst von einem zyklischen Graphen. Man nennt G zusammenhängend falls gilt:*

$$\forall v_i, v_j \in V \ \exists \ \textit{Pfad } P \textit{ mit } P = (v_i = v_0, v_1, \ldots, v_n = v_j)$$

Definition 2.1.4 (Baum) *Ein zusammenhängender azyklischer Graph mit einem ausgezeichneten Knoten (Wurzel) wird als Baum bezeichnet.*

Definition 2.1.5 (Subgraph) *Ein Graph $G' = (V', E')$ ist ein Subgraph des Graphen $G = (V, E)$, wenn $V' \subset V$ und $E' \subset E$.*

Definition 2.1.6 (Clique) *Sei $G' = (V', E')$ ein Subgraph des Graphen $G = (V, E)$. Existiert zwischen jedem Paar unterschiedlicher Knoten von V' eine Kante, so bezeichnet man den Graphen G' als vollständigen Subgraphen oder Clique.*

Definition 2.1.7 (p-Färbung eines Graphen) *Eine p-Färbung eines Graphen G ist eine Abbildung $\phi : V \longrightarrow \{1, 2, \ldots, p\}$ mit $\phi(v_1) \neq \phi(v_2)$ genau dann, wenn die Knoten v_1 und v_2 nicht adjazent sind. Existiert eine p-Färbung des Graphen G, so bezeichnet man den Graphen auch als p-colorierbar. Das kleinste p, für das G p-colorierbar ist, nennt man auch chromatische Zahl $\chi(G)$. Eine p-Färbung von G nennt man optimal genau dann, wenn $p = \chi(G)$.*

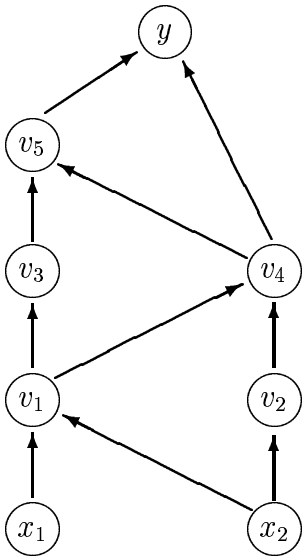
2.1.4 Darstellung eines Ausdruckes durch einen Graphen

Jede beliebige faktorisierbare Funktion f aus Abschnitt 2.1.2 läßt sich nach GRIEWANK & UTKE (1995) durch einen gerichteten, azyklischen Graphen $G = (V, E)$ darstellen. Die n unabhängigen Variablen $x_1, x_2, \ldots, x_n$, die m abhängigen Variablen $y_1, y_2, \ldots, y_m$ sowie die l Zwischengrößen $v_1, v_2, \ldots, v_l$ bilden dabei die Knoten von G. Diese Darstellung wird auch als *Kantorovitch-Graph* bezeichnet. Eine Kante e verläuft dabei von Knoten v_i nach

Knoten v_j genau dann, wenn der Wert von v_j direkt von v_i abhängt. D.h. für jede Elementarfunktion ϕ_i sind alle Argumentknoten aus U_i mit den Ergebnisknoten verbunden. Das Beispiel (2.3) aus GRIEWANK (1994) diene zur Veranschaulichung:

$$f : \begin{cases} \mathcal{D} \subset I\!R^2 \longrightarrow I\!R^1 \\ x = (x_1, x_2) \in \mathcal{D} \longmapsto f(x) = \dfrac{\cos(x_1 \cdot x_2) + x_1 \cdot x_2 - \exp(x_2)}{x_1 \cdot x_2 - \exp(x_2)} \end{cases} \tag{2.3}$$

Die Abbildung 2.1 veranschaulicht zusätzlich den zur Funktion (2.3) gehörenden Berechnungsgraphen.



$$\begin{aligned} v_1 &:= x_1 \cdot x_2 \\ v_2 &:= \exp(x_2) \\ v_3 &:= \cos(v_1) \\ v_4 &:= v_1 - v_2 \\ v_5 &:= v_3 + v_4 \\ y &:= v_5 / v_4 \end{aligned}$$

Abbildung 2.1: Graph-Repräsentation eines Berechnungsalgorithmus

Jeder Berechnungsgraph läßt sich nach GRIEWANK & UTKE (1995) ohne weiteres durch Vervielfachung zu einem *Wald* (eine Familie von M disjunkten Bäumen) erweitern. Das Ergebnis dieser Transformation ist in Abbildung 2.2 zu sehen.

Da die Funktion f skalarwertig ist, gilt $M = 1$. Der Wald besteht somit aus einem einzigen Baum. Dieser Baum hat 18 Knoten im Vergleich zu den 8 Knoten des Graphen in Abbildung 2.1. Die Anzahl der Vervielfältigungen jedes Knotens im Baum ist dabei gleich der Anzahl der verschiedenen direkten Pfade dieses Knotens zu dem Vaterknoten y. Der daraus folgende exponentiell wachsende Speicherplatz erschwert die explizite Darstellung der abhängigen Variablen als algebraische Ausdrücke durch die unabhängigen Variablen. Moderne Computeralgebrasystem wie beispielsweise MAPLE versuchen daher, die Vervielfachung gemeinsamer Teilausdrücke wie z.B. $(x_1 \cdot x_2)$ im obigen Beispiel zu vermeiden.

Die gewählte Reihenfolge für die Abarbeitung der Einzeloperationen eines mathematischen Ausdruckes spielt nach STOER & BULIRSCH (1971) auch in der *differentiellen Fehleranalyse* eines Algorithmus eine wichtige Rolle. Dabei wird untersucht, inwiefern sich die Eingangsfehler Δx von x und die im Laufe des Algorithmus auftretenden Rundungsfehler auf das Endresultat auswirken.

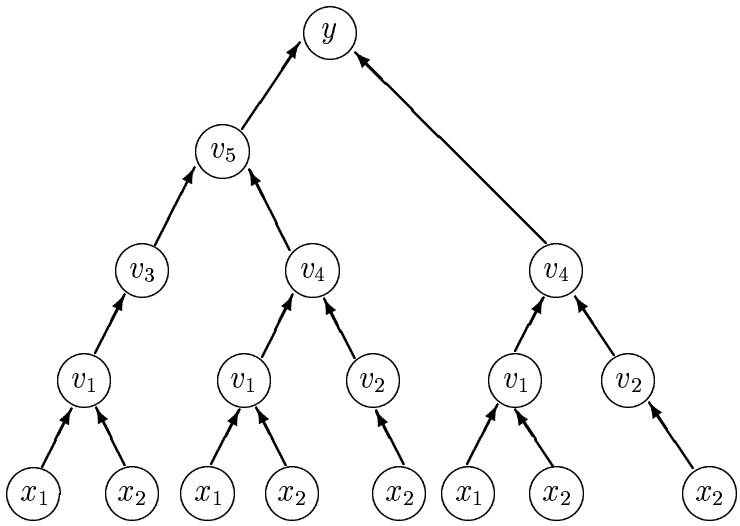


Abbildung 2.2: Vervielfachung der Zwischenknoten zur Generierung eines Baumes

2.2 Verfahren des AD

2.2.1 Vorbemerkungen

Im folgenden wird stets eine Funktion der Form

$$F : \begin{cases} \mathcal{D} \subset I\!R^n \longrightarrow I\!R^m \\ x = (x_1, x_2, \ldots, x_n) \in \mathcal{D} \longmapsto y = (y_1, y_2, \ldots, y_m) \end{cases} \tag{2.4}$$

betrachtet. Die Richtungsableitung von F nach gegebenen p Richtungen $\dot{x} \in I\!R^{n \times p}$ wird mit $\dot{y}$ bezeichnet[1], und wie folgt definiert:

$$\dot{y} := \left\langle \frac{dF}{dx}, \dot{x} \right\rangle = \frac{\partial F}{\partial x}\dot{x} = F'(x)\dot{x} \tag{2.5}$$

Hierbei ist $F'(x) \in I\!R^{m \times n}$ $(x \in \mathcal{D})$ die Jacobi-Matrix von F. In den Spalten von $\dot{x}$ stehen die einzelnen p Richtungen, nach denen abgeleitet wird. Setzt man $\dot{x}$ mit der Einheitsmatrix I_n gleich, so ist $\dot{y}$ gleich der Jacobi-Matrix von F.

Die zwei grundlegenden Verfahren des automatischen Differenzierens sind der Vorwärtsmodus (*forward mode*) und der Rückwärtsmodus (*backward mode*). Der grundlegende Unterschied beider Verfahren liegt darin, daß im Vorwärtsverfahren die partiellen Ableitungen jedes Zwischenwertes v_i nach den unabhängigen Variablen und im Rückwärtsverfahren die partiellen Ableitungen der abhängigen Variablen nach jedem Zwischenwert v_i berechnet werden.

Die Abschnitte 2.2.3 und 2.2.4 werden zeigen, daß das Vorwärtsverfahren für Funktionen vom Typ $F : I\!R \longrightarrow I\!R^m$ geeignet ist. Für skalare Funktionen vom Typ $f : I\!R^n \longrightarrow I\!R$, hingegen ist der Rückwärtsmodus optimal in Hinblick auf die Laufzeit.

[1] Aus Konsistenzgründen zur allgemeinen AD-Literatur wir hier die normalerweise Zeitableitungen bezeichnende Notation $\dot{y}$ für beliebige Richtungsableitungen verwendet.

2.2.2 Anwendung der Kettenregel

Das mathematische Grundprinzip der Automatischen Differentiation ist die wiederholte Anwendung der Kettenregel[2]. Beispielsweise findet man im Lehrbuch von HEUSER (1990):

Satz 2.2.1 (Kettenregel) *Die Funktion g sei auf dem Intervall I_g definiert, die Funktion f auf dem Intervall $I_f \supset g(I_g)$ erklärt; g sei in t, f in $g(t)$ differenzierbar. Dann ist $f \circ g$ in t differenzierbar, und es gilt für alle $t^* \in I_g$:*

$$\frac{\partial (f \circ g)(t^*)}{\partial t} = \left.\frac{\partial f(g(t))}{\partial t}\right|_{t=t^*} = \left(\frac{\partial f(s)}{\partial s} \mid_{s=g(t^*)}\right)\left(\frac{\partial g(t)}{\partial t} \mid_{t=t^*}\right). \tag{2.6}$$

Die Kettenregel wird jetzt auf die einzelnen Kompositionen der Elementaroperationen ϕ_i angewendet. Damit lassen sich die Ableitungen der gegebenen Funktion *exakt* berechnen. Dabei bedeutet *exakt* in diesem Zusammenhang, daß die berechneten Ableitungen bis auf Rundungsfehler, die bei den Elementaroperationen ϕ_i im Rahmen der Maschinengenauigkeit auftreten, mit den tatsächlichen Werten übereinstimmen. Die Ketten- bzw. Produktregel wird dabei im Vergleich zur symbolischen Differentiation nicht auf Formelteile angewendet, sondern auf die Werte der Ableitungen der Elementaroperationen ϕ_i, d.h. auf reelle Zahlen. Das Arbeiten mit Formeln wachsender Komplexität wird dadurch vermieden.

Ziel ist im folgenden Ableitungen der Funktion F nach den d Richtungen $p_1, p_2, \ldots, p_d$ mit $d \in \mathbb{N}$ und $p_i \in \mathbb{R}^n$, $1 \leq i \leq d$ automatisch zu berechnen.

Zunächst betrachtet man den Graphen $G = (V, E)$ aus Abschnitt 2.1.4, der den zu berechnenden mathematischen Ausdruck repräsentiert. Dieser Graph wird für die Berechnung der Ableitungen verwendet: Jedem Knoten $i \in V$ wird nun das Paar (v_i, v_i') zugeordnet. Dabei repräsentiert v_i den Wert der Zwischenvariable und v_i' ein dem Zwischenwert zugeordnetes Ableitungsobjekt der Dimension d. Das Ableitungsobjekt v_i' wird in den beiden Verfahren aus Abschnitt 2.2.3 bzw. Abschnitt 2.2.4 unterschiedlich interpretiert. Im Vorwärtsmodus repräsentiert v_i' die d Richtungsableitungen von v_i nach den Richtungen $p_1, p_2, \ldots, p_d$. Im Rückwärtsmodus repräsentiert v_i' die Ableitungen der unabhängigen Variablen y nach v_i. Jeder Kante $e \in E$ von i nach j wird nun die partielle Ableitung

$$a_{ij} = \frac{\partial v_i}{\partial v_j} \tag{2.7}$$

zugeordnet. Die partiellen Ableitungen a_{ij} gehen somit als reelle Zahlen in die Multiplikation, die aus der Kettenregel (2.2.1) hervorgeht, ein. Diese elementare Tatsache ist die Grundvoraussetzung für die Genauigkeit des AD, in der gleichen Größenordnung der des symbolischen Differenzierens (siehe Abschnitt 2.4) ist.

Mit Hilfe der Kettenregel läßt sich nun die Ableitung jedes beliebigen Zwischenwertes v_l nach einem beliebigen anderen Zwischenwert v_k berechnen. Sei dazu M_{kl} die Menge aller Pfade von k nach l. Es gilt somit nach mehrfacher Anwendung der Kettenregel:

$$\boxed{\frac{\partial v_l}{\partial v_k} = \sum_{P \in M_{kl}} \left(\prod_{e \in P} a_{ij} \right)} \tag{2.8}$$

[2] Aufgrund ihrer zentralen Bedeutung für das AD wird diese elementare Regel der Analysis hier explizit aufgeführt.

Der Graph G besteht nach Voraussetzung nur aus elementaren Funktionen und Operatoren. Somit lassen sich die partiellen Ableitungen a_{ij} leicht aus Tabellen, in denen die Ableitungsformeln gespeichert stehen, berechnen. Die Tabelle 2.2 faßt nach LOHNER (1993) die partiellen Ableitungen der grundlegenden binären Operatoren sowie die Ableitungen der mathematischen Standard-Funktionen, die bei allen gängigen Programmiersprachen als intrische Funktionen (Fortran77) bzw. Bibliotheksfunktionen (C, C++) implementiert werden, zusammen.

ϕ_i	$\frac{\partial \phi_i}{\partial v_j}$	ϕ_i	$\frac{\partial \phi_i}{\partial v_j}$	ϕ_i	$\frac{\partial \phi_i}{\partial v_j}$	ϕ_i	$\frac{\partial \phi_i}{\partial v_j}$
$v_j + v_r$	1	$v_l + v_j$	1	$v_j - v_r$	1	$v_l - v_j$	-1
$v_l \cdot v_j$	v_l	$v_j \cdot v_r$	v_r	$\frac{v_j}{v_r}$	$\frac{1}{v_r}$	$\frac{v_l}{v_j}$	$\frac{-\phi_i}{v_j}$
v_j^2	$2v_j$	$\sqrt{v_j}$	$\frac{1}{2\phi_i}$	e^{v_j}	ϕ_i	2^{v_j}	$\phi_i \cdot \ln 2$
10^{v_j}	$\phi_i \cdot \ln 10$	$\ln v_j$	$\frac{1}{v_j}$	$\log_2 v_j$	$\frac{1}{v_j \cdot \ln 2}$	$\log_{10} v_j$	$\frac{1}{v_j \cdot \ln 10}$
$\sin v_j$	$\cos v_j$	$\cos v_j$	$-\sin v_j$	$\tan v_j$	$1 + \phi_i^2$	$\cot v_j$	$-(1 + \phi_i^2)$
$\arcsin v_j$	$\frac{1}{\sqrt{1 - v_j^2}}$	$\arccos v_j$	$\frac{-1}{\sqrt{1 - v_j^2}}$	$\arctan v_j$	$\frac{1}{1 + v_j^2}$	$\operatorname{arccot} v_j$	$\frac{-1}{1 + v_j^2}$
$\sinh v_j$	$\cosh v_j$	$\cosh v_j$	$\sinh v_j$	$\tanh v_j$	$1 - \phi_i^2$	$\coth v_j$	$-(1 - \phi_i^2)$
$\operatorname{arsinh} v_j$	$\frac{1}{\sqrt{v_j^2 + 1}}$	$\operatorname{arcosh} v_j$	$\frac{-1}{\sqrt{v_j^2 - 1}}$	$\operatorname{artanh} v_j$	$\frac{1}{1 - v_j^2}$	$\operatorname{arcoth} v_j$	$\frac{-1}{1 - v_j^2}$

Tabelle 2.2: Tabelle mit Ableitungen der Elementarfunktionen

Komplexität der elementaren Ableitungen

Bei denjenigen Funktionen, die durch ein sequentielles Programm wie in Abschnitt 2.1.2 definiert sind, werden bei der Ableitungserzeugung mit AD die partiellen Ableitungen unter Verwendung der Präzedenz-Relation (2.2)

$$a_{ij} := \frac{\partial v_i}{\partial v_j} = \frac{\partial}{\partial v_j} \phi_i \left(U_i \right) \quad \text{mit} \quad j \prec i \tag{2.9}$$

der Zwischenvariablen v_i nach allen Vorgängervariablen v_j benötigt. Bei der Berechnung von zweiten Ableitungen werden hingegen die partiellen zweiten Ableitungen

$$a_{ijk} = \frac{\partial^2 v_i}{\partial v_j \partial v_k} = \frac{\partial^2}{\partial v_j \, \partial v_k} \phi_i \left(U_i \right) \quad \text{mit} \quad j \prec i \quad \text{und} \quad k \prec i \tag{2.10}$$

benötigt. Bei allen Programmen, die in einer höheren Programmiersprache codiert werden, sind die a_{ij} bzw. die a_{ijk} in der Regel wohldefiniert und einfach zu berechnen.

Um die Komplexität der beim AD durchgeführten Berechnungen zu bestimmen, bezeichnet $OPS(F)$ ein Maß für die Abschätzung des Aufwandes zur Auswertung einer Funktion F. Geht man von der Annahme aus, daß alle Elementaroperationen unendlich oft differenzierbar sind, so lassen sich folgende Aussagen über das AD ableiten:

- Da jede partielle Ableitung a_{ij} analytisch berechnet wird, ist jede berechnete Ableitung $F'(x)$ bis auf Rundungsfehler exakt. Es tritt somit kein Verfahrensfehler wie beim numerischen Differenzieren in Abschnitt 2.3 auf.
- Es gilt die auf einer sequentiellen Maschine erfüllte Additivitätsbedingung :

$$OPS(F) = \sum_{i=1}^{l} OPS\left(\phi_i\right) \tag{2.11}$$

- Die Operationen für alle Basisfunktionen und Elementaroperationen ϕ_i lassen sich nach oben durch eine Konstante q_i abschätzen:

$$OPS\left(\phi_i\left(U_i\right)\right) + OPS\left(\left(a_{ij}\right)_{j \prec i}\right) \leq q_i \cdot OPS\left(\phi_i\left(U_i\right)\right) \tag{2.12}$$

Der Aufwand zur Berechnung des Funktionswertes ϕ_i und aller partiellen Ableitungen a_{ij} mit $j \prec i$ ist dadurch nur ein konstantes, kleines Vielfaches q_i aufwendiger als die Berechnung von ϕ_i. Der Faktor q_i hängt in der Praxis von der exakten Definition von $OPS(F)$ und des konkreten Rechners, auf dem die Berechnung läuft, ab. Falls nur Multiplikationen betrachtet werden, gilt nach GRIEWANK (1994) für die obere Schranke $q := \max_{1 \leq i \leq l} q_i$ der Wert $q = 3$. Im folgenden wird für q der konservativere Wert $q = 5$ angenommen, der auch Speicherzugriffe und weitere Kosten berücksichtigt.

Jedem Knoten i des Graphen aus Abschnitt 2.2.2 wird neben dem Zwischenwert v_i auch ein Ableitungsobjekt v_i' der Größe d zugeordnet. Mit der Gleichung (2.11) und der Ungleichung (2.12) gilt folgende Abschätzung für die Berechnung des Gesamtaufwandes von F und der korrespondierenden Ableitung F':

$$\begin{aligned} OPS\left(F, F'\right) &= \sum_{i=1}^{l} d \cdot \left(OPS\left(\phi_i\right) + OPS\left(\left(a_{ij}\right)_{j \prec i}\right)\right) \leq d \cdot \sum_{i=1}^{l} q_i \cdot OPS\left(\phi_i\right) \\ &\leq q \cdot d \cdot \sum_{i=1}^{l} OPS\left(\phi_i\right) \leq 5 \cdot d \cdot \sum_{i=1}^{l} OPS\left(\phi_i\right) \\ &= 5 \cdot d \cdot OPS\left(F\right) \end{aligned} \tag{2.13}$$

2.2.3 Vorwärtsmodus

Das Grundprinzip des Vorwärtsmodus ist die Berechnung von partiellen Ableitungen des Vektors y der abhängigen Variablen nach dem Vektor x der unabhängigen Variablen durch

sukzessives Berechnen der partiellen Ableitungen aller benötigten Zwischenvariablen nach den unabhängigen Variablen x. Jede einzelne Zuweisung in der Berechnungssequenz aus Abschnitt 2.1.2 wird nach den Input-Größen x differenziert. Berechnet wird somit die Richtungsableitung $\dot{y} = F'(x) \cdot \dot{x} \in I\!R^m$ von F nach der Richtung $\dot{x} \in I\!R^n$. Die Input-Größen des Algorithmus sind somit $x, \dot{x} \in I\!R^n$. Jede elementare Zuweisung $v_i := \phi_i(U_i)$ der Berechnungsprozedur aus Abschnitt 2.1.2 impliziert die korrespondierende Differentiationsoperation:

$$\dot{v}_i := \sum_{v_j \in U_i} \frac{\partial \phi_i\,(U_i)}{\partial v_j} \dot{v}_j \qquad (2.14)$$

Unter Verwendung der faktorisierten Darstellung und der l Zwischenvariablen (siehe Abschnitt 2.1.2) läßt sich der Vorwärtsmodus algorithmisch somit wie folgt beschreiben:

Algorithmus 2.2.2 *Vorwärtsmodus*

1. $\{$*Initialisierung*$\}$

   for $i := 1$ **up to** n **do**

   $v_{i-n} := x_i$

   $\dot{v}_{i-n} := \dot{x}_i$

   od

2. $\{$*Ableitungsberechnung im Vorwärtslauf*$\}$

   for $i := 1$ **up to** l **do**

   $U_i := \{v_j \mid j \prec i\}$

   $v_i := \phi_i(U_i)$

   $\dot{v}_i := \sum_{U_i} \frac{\partial \phi_i\,(U_i)}{\partial v_j} \cdot \dot{v}_j$

   od

3. $\{$*Zuweisung der berechneten Ableitungen*$\}$

   for $i := m - 1$ **down to** 0 **do**

   $y_{m-i} := v_{l-i}$

   $\dot{y}_{m-i} := \dot{v}_{l-i}$

   od

Im obigen Algorithmus wird die Größe d für jedes Ableitungsobjekt auf Eins gesetzt, da nur eine Richtungsableitung berechnet wird. Bei der Berechnung der Jacobi-Matrix von F gilt $d = n$, da hier nach allen n Einheitsrichtungen abgeleitet wird. Bei der Berechnung der Ableitung nach den $d-$Richtungen $\dot{x}_1, \dot{x}_2, \ldots, \dot{x}_d \in I\!R^n$ werden anstelle *eines* Wertes $\dot{v}_i$ für jeden Zwischenwert die d Werte $\dot{v}_{i1}, \dot{v}_{i2}, \ldots, \dot{v}_{id} \in I\!R$ benötigt. In allen Teilen des obigen Algorithmus werden die d Werte aller Ableitungsobjekte $\dot{v}_i$ in einer Schleife der Länge d berechnet. Für Teil zwei ergeben sich folgende Anweisungen:

for $i := 1$ **up to** l **do**
  ...
  for $k := 1$ **up to** d **do**
    $\dot{v}_{ik} := \sum_{U_i} \frac{\partial \phi_i (U_i)}{\partial v_j} \cdot \dot{v}_{jk}$
  od
od

Komplexität des Vorwärtsmodus

Bei der Berechnung der Jacobi-Matrix von F wird jeder Zwischenvariablen v_i mit $i = 1-n, \ldots, 0, 1, \ldots, l$ ein Gradientenvektor

$$\dot{v_i} := \nabla v_i = (\dot{v}_{i1}, \ldots, \dot{v}_{in}) = (\partial v_i / \partial v_j) \quad \text{mit} \quad (j = 1, \ldots, n) \tag{2.15}$$

zugeordnet. Der Gradient des Vektors x der unabhängigen Variablen wird mit $\nabla x_j = \nabla v_{1-j} = e_j \in I\!R^n$ initialisiert. Hier bezeichnet e_j den j-ten kartesischen Einheitsvektor des $I\!R^n$. Berechnet werden also $(d = n)$ Richtungsableitungen. Mit Hilfe der Abschätzung (2.13) gilt für die Berechnung der Jacobi-Matrix einer Funktion $F : I\!R^n \longrightarrow I\!R^m$ mit dem Vorwärtsmodus:

$$OPS(F, F') \leq 5 \cdot n \cdot OPS(F) \tag{2.16}$$

2.2.4 Rückwärtsmodus

Wie Abschnitt 2.2.3 zeigt, lassen sich Jacobi-Matrizen mit dem Vorwärtsmodus in einer Zeitkomplexität proportional zur Anzahl n der unabhängigen Variablen bestimmen. In diesem Abschnitt wird ein Verfahren beschrieben, das die Ableitungen in einer Komplexität berechnet, die unabhängig von n ist. Der Gradient ∇f einer skalaren Funktion f läßt sich daher in einem konstanten Vielfachen der Berechnungszeit von f bestimmen.

Das Rückwärtsverfahren arbeitet im Prinzip analog zum Vorwärtsverfahren. Das Ableitungsobjekt v_i', das jedem Knoten v_i des Graphen aus Abschnitt 2.1.4 zugeordnet wird, nennt man *Adjungierte* (*Adjoint*). Analog gilt für die *Linearkombination* $\bar{x}$ des Gradienten einer vektorwertigen Funktion F, definiert durch:

$$\bar{x} := \nabla [\bar{y} \cdot F(x)] = \bar{y} \cdot \nabla F(x) = \bar{y} \cdot F'(x) \in I\!R^n \tag{2.17}$$

Dabei bezeichnet $\bar{y} \in I\!R^m$ einen festen Zeilenvektor, der analog zum Richtungsvektor $\dot{x}$ aus Abschnitt 2.2.3 zu verstehen ist. In Abschnitt 2.2.3 wurde $\dot{y} = F'(x)\dot{x} = \dot{F}(x, \dot{x}; y)$ als eine von x und $\dot{x}$ abhängige Funktion berechnet. Jetzt wird $\bar{x} = \bar{y} \cdot F'(x)$ als eine von x und $\bar{y}$ abhängige *Linearkombination* berechnet. Aus den Gleichungen (2.5) und (2.17) folgt die Identität:

$$\bar{y} \cdot \dot{y} = \bar{y} \cdot (F'(x) \cdot \dot{x}) = (\bar{y} \cdot F'(x)) \cdot \dot{x} = \bar{x} \cdot \dot{x}$$
$$\forall\, (\dot{x}, \dot{y}) \in I\!R^n \times I\!R^m \text{ mit } \dot{y} = F'(x) \cdot \dot{x} \tag{2.18}$$

Um die Vorgehensweise des Rückwärtsmodus formal herzuleiten, wird zunächst nach GRIEWANK & CHRISTIANSEN (1998) die Berechnungsvorschrift für das Vorwärtsverfahren in eine Matrix-Vektor Schreibweise umgeschrieben. Mit den elementaren partiellen Ableitungen a_{ij} aus Gleichung (2.7) wird für alle $i = 1, 2, \ldots, l-1, l$ gesetzt:

$$A_i = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & 1 & \cdots & 0 & \cdot & \cdot & \cdot & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \cdot & \cdot & \cdot & 0 \\ a_{i\,1-n} & a_{i\,2-n} & \cdot & a_{i\,i-1} & 0 & \cdot & \cdot & 0 \\ 0 & 0 & \cdot & \cdot & \cdot & 1 & \cdot & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \end{bmatrix} \in I\!R^{(n+l)\times(n+l)} \tag{2.19}$$

Die a_{ij} treten dabei in der $(n+i)$-ten Zeile von A_i auf. Die quadratischen Matrizen A_i sind untere Dreiecksmatrizen und können unter Verwendung der Einheitsmatrix I wie folgt zerlegt werden.

$$A_i = I + e_{n+i} \left(\nabla \phi_i(U_i) - e_{n+i}\right)^T \tag{2.20}$$

Hierbei bezeichnen die e_j die j-ten kartesischen Einheitsvektoren. Seien weiter

$$P_n := [I, 0, \ldots, 0] \in I\!R^{n\times(n+l)} \text{ und } Q_m := [0, \ldots, 0, I] \in I\!R^{m\times(n+l)}$$

diejenigen Matrizen, die multipliziert mit einem beliebigen Vektor der Größe $(n+l)$ die Projektion dieses Vektors auf seine ersten n bzw. letzten m Komponenten liefern. Mit Hilfe dieser Matrizen läßt sich die Berechnung von $\dot{y}$ aus $\dot{x}$ durch Gleichung (2.5) wie folgt definieren:

$$\boxed{\dot{y} = Q_m \cdot A_l \cdot A_{l-1} \cdot \ldots \cdot A_2 \cdot A_1 \cdot P_n^T \cdot \dot{x}} \tag{2.21}$$

Dabei entsprechen die einzelnen Multiplikationen den einzelnen Anweisungen im Algorithmus aus Abschnitt 2.2.3:

- Die Multiplikation mit P_n^T, die $\dot{x}$ in den $I\!R^{n+l}$ abbildet, korrespondiert zum Initialisierungsteil aus Teil 1.
- Die wiederholte Multiplikation mit den A_i generiert das Ableitungsobjekt $\dot{v}_i$ für alle Zwischenvariablen v_i aus Teil 2.
- Die abschließende Multiplikation mit Q_m extrahiert die letzten m Komponenten, die den Vektor $\dot{y}$ ergeben. Dies korrespondiert mit der Zuweisung der Ableitungen im dritten Teil des Algorithmus.

Der Vergleich mit Gleichung (2.5) ergibt somit eine multiplikative Darstellung der gesamten Jacobi-Matrix $F'(x)$:

$$F'(x) = Q_m \cdot A_l \cdot A_{l-1} \cdot \ldots \cdot A_2 \cdot A_1 \cdot P_n^T \in \mathbb{R}^{m \times n} \quad (2.22)$$

Die Berechnung von $\bar{x}$ aus Gleichung (2.17) erhält man folglich aus dem Transponieren der Gleichung (2.22):

$$\begin{aligned} \bar{x}^T &= \left[\bar{y} \cdot F'(x)\right]^T \\ &= F'(x)^T \cdot \bar{y}^T \\ &= \left(Q_m \cdot A_l \cdot A_{l-1} \cdot \ldots \cdot A_2 \cdot A_1 \cdot P_n^T\right)^T \cdot \bar{y}^T \\ &= P_n \cdot A_1^T \cdot A_2^T \cdot \ldots \cdot A_{l-1}^T \cdot A_l^T \cdot Q_m^T \cdot \bar{y}^T \end{aligned} \quad (2.23)$$

Dadurch erhält man eine explizite Darstellung der durch den Rückwärtsmodus berechneten Linearkombination $\bar{x}$.

Da nach Gleichung (2.20) gilt:

$$A_i^T = \left[I + e_{n+i} \cdot \left(\nabla \phi_i(U_i) - e_{n+i}\right)^T\right]^T = I + \left(\nabla \phi_i(U_i) - e_{n+i}\right) \cdot e_{n+i}^T \,, \quad (2.24)$$

läßt sich erkennen, daß die wiederholte Multiplikation eines gegebenen Vektors $(\bar{v}_j)_{i-n \le j \le l}$ mit den Matrizen A_i^T eine inkrementelle Operation darstellt:

Diejenigen $\bar{v}_j$ mit $j \not\prec i$ und $j \neq i$ bleiben gleich. Hingegen werden alle $\bar{v}_j$ $(j \prec i)$ mit $\bar{v}_i \cdot a_{ij}$ inkrementiert, und $\bar{v}_i$ wird auf Null gesetzt. Dadurch sind zum Schluß nur die ersten n Adjungierten mit Indizes $1 - n \le i \le 0$ von Null verschieden. Diese Werte werden durch die Projektion mit der Matrix P_n auf die einzelnen Komponenten $\bar{x}_i$ des Vektors $\bar{x}$ gesetzt.

Praktisch kann diese Vorgehensweise folgendermaßen implementiert werden: Zunächst werden in einem vorwärtsgerichteten Lauf die Funktionswerte v_i $(1 \le i \le l + m)$ der Zwischenvariablen berechnet. Dabei werden zusätzlich die partiellen Ableitungen a_{ij} geeignet gespeichert. Im darauffolgenden rückwärtsgerichteten Lauf werden die Adjungierten in umgekehrter Reihenfolge inkrementell berechnet.

Inkrementeller Modus des Rückwärtsverfahren

Nach GRIEWANK & UTKE (1995) läßt sich das Matrix-Vektor Produkt (2.23) als folgender Berechnungsalgorithmus für die Adjungierten formulieren:

Algorithmus 2.2.3 *Rückwärtsmodus*

1. {*Forward Sweep*}

   (a) {*Zuweisung des Vektors x an die Zwischenwerte*}

   for $i := 1$ **up to** n **do** $v_{i-n} := x_i$ **od**

   (b) {*Initialisierung der Adjungierten und Berechnung der Zwischenwerte*}

for $i := 1$ **up to** l **do**
$\bar{v}_i := 0$
$U_i := \{v_j \mid j \prec i\}$
$v_i := \phi_i(U_i)$
od

(c) {*Zuweisung der berechneten Funktionswerte*}

for $i := m-1$ **down to** 0 **do** $y_{m-i} := v_{l-i}$ **od**

2. {*Reverse Sweep*}

(a) {*Zuweisung der Komponenten von* $\bar{y}$ *an die Adjungierten* $\bar{v}_l, \ldots, \bar{v}_{l-m}$}

for $i := 0$ **up to** $m-1$ **do** $\bar{v}_{l-i} := \bar{y}_{m-i}$ **od**

(b) {*Inkrementelle Berechnung der Adjungierten im Rückwärtslauf*}

for $i := l$ **down to** 1 **do**
for *all* $j \in U_i$ **do**
$d := \bar{v}_i \cdot \dfrac{\partial}{\partial v_j}\phi_i(U_i)$
$\bar{v}_j := \bar{v}_j + d$
od
$\bar{v}_i := 0$
od

(c) {*Zuweisung der berechneten Richtungsableitungen*}

for $i := n$ **down to** 1 **do** $\bar{x}_i := \bar{v}_{i-n}$ **od**

Die Berechnung der Adjungierten erfolgt dabei in einem Rückwärtslauf. Nur dadurch ist garantiert, daß jedes $\bar{v}_j$ einen korrekten Wert zugewiesen bekommt, bevor $\bar{v}_j$ auf der rechten Seite einer Zuweisung steht.

Nichtinkrementeller Modus des Rückwärtsverfahren

Eine Modifikation des obigen Algorithmus besteht darin, anstelle der beiden ineinandergeschachtelten Schleifen zur Berechnung der Adjungierten $\bar{v}_j$, eine Schleife über j zu verwenden, in der in jedem Durchlauf über alle i, für die $i \succ j$ gilt aufsummiert wird:

$$\bar{v}_j := \sum_{i \succ j} \bar{v}_i \cdot \frac{\partial}{\partial v_j}\phi_i\left(U_i\right) \quad (2.25)$$

Dabei durchläuft die Schleife die Werte $j = l-m, \ldots, 1-n$ in dieser Reihenfolge. Diese Version wird nach Griewank & Christiansen (1998) auch als nichtinkrementelle Adjungierte-Rekursion bezeichnet.

Ein offensichtlicher Nachteil der nichtinkrementellen Version besteht darin, daß für jeden Zwischenwert v_j eine Liste von Knoten gespeichert werden muß, die alle Variablen i enthält, die von j abhängen. Solche vorwärts gerichteten Zeiger von einem Index j auf alle seine Nachfolger $i \succ j$ sind im allgemeinen nicht ohne größeren Aufwand zu erhalten. Wohingegen die rückwärts gerichteten Zeiger von einem Index i auf alle seine Vorgänger $j \prec i$ durch die Zerlegung von F in die elementaren Basisfunktionen ϕ_i bereits gegeben sind. Die nichtinkrementelle Version benötigt also globale Informationen, im Gegensatz zur inkrementellen Version, die die Berechnung der $\bar{v}_j$ allein durch Betrachtung aller Argumente $v_j \in U_i$ der Basisfunktion ϕ_i zu einem Iterationspunkt ausführen kann.

Komplexität des Rückwärtsmodus

Analog zu den Komplexitätsaussagen in Abschnitt 2.2.3 wird für jede der l Zwischenvariablen v_i ein Adjungierten-Vektor $\bar{v}_i = (\bar{v}_{i1}, \ldots, \bar{v}_{im})$ der Größe m mit Hilfe der Kettenregel berechnet. Folglich gilt für die Größe d aus Gleichung (2.13) der Wert $d = m$, und damit für die Berechnung der Jacobi-Matrix einer Funktion $F : I\!R^n \longrightarrow I\!R^m$ mit dem Rückwärtsmodus:

$$OPS(F, F') \leq 5 \cdot m \cdot OPS(F) \qquad (2.26)$$

Für den in der Praxis wichtigen Fall einer Funktion $f : I\!R^n \longrightarrow I\!R$ ist der Aufwand zur Berechnung von Funktionswert und Gradient höchstens das Fünffache des Aufwands der Funktionsauswertung, unabhängig von der Dimension n der abhängigen Variablen x der Funktion.

2.2.5 Beispiel für die automatische Ableitungsgenerierung

Als anschauliches Beispiel für die in Abschnitt 2.2.3 und 2.2.4 betrachteten Verfahren diene folgendes, aus der Praxis entnommene, Beispiel. Es handelt sich um die in 5.3 näher beschriebene Phosphin-Reaktion. Man betrachte im folgenden eine skalarwertige Funktion $F : I\!R^2 \to I\!R$, die die zeitliche Ableitung einer Stoffmenge, die bei einer chemischen Reaktion mit vier Spezies auftritt, beschreibt. Um F zu berechnen, benötigt man eine Reihe von Variablen: Die Aktivierungsenergie E_a, die universelle Gaskonstante R, den Frequenzfaktor k_{ref}, die Referenztemperatur T_{ref}, die Molmassen $M_1, \ldots, M_4$ der beteiligten Spezies, das Volumen V der Reaktionsmasse, die durchschnittlichen Dichte ϱ der Spezies, die Anfangsstoffmengen n_{a_1} und n_{a_4} der Spezies 1 und 4 sowie die Stoffmengen n_{e_2} und n_{e_4} der Spezies 2 und 4 des Zulaufs. Die beiden unabhängigen Variablen sind dabei durch $x_1 := T$ (Temperatur im Reaktor) und $x_2 := n_1$ (Stoffmenge von Spezies 1) definiert.

Die Berechnung von F erfolgt dabei durch folgende Vorschrift:

$$
\begin{array}{lcl}
n_3 & := & n_{a_1} - n_1 \\
n_2 & := & n_{e_2} - n_3 \\
n_4 & := & n_{a_4} + n_{e_4} \\
V & := & \left(\sum_{i=0}^{4} M_i \cdot n_i \right) / \varrho \\
F := \dfrac{\partial x_2}{\partial t} & = & -V \cdot k_{ref} \cdot \exp\left(-\dfrac{E_a}{R} \cdot \left(\dfrac{1}{x_1} - \dfrac{1}{T_{ref}} \right) \right) \cdot \dfrac{x_2}{V} \cdot \dfrac{n_2}{V}
\end{array}
\tag{2.27}
$$

Um F mit Hilfe der beiden Modi des Rückwärtsmodus automatisch nach den Input-Größen x_1 und x_2 abzuleiten, werden die einzelnen Adjungierten wie in Abbildung 2.3 berechnet. Die Abkürzungen $v \mathrel{+}= w$ anstelle von $v := v + w$, bzw. $v \mathrel{-}= w$ anstelle von $v := v - w$ werden analog zur Konvention in der Programmiersprache C verwendet. Die verwendeten temporären Adjungierten $\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_{23}, \bar{v}_{24}$ werden dabei im inkrementellen Modus mit Null initialisiert. Sowohl dem inkrementellen als auch dem nichtinkrementellen Rückwärtsmodus geht die Berechnung der Funktion F im sogenannten *Forward Sweep* voraus.

2.2.6 Höhere Ableitungen

In diesem Abschnitt wird ein Algorithmus nach CHRISTIANSON (1992) zur Berechnung des Vektor-Matrix-Produktes $H \cdot u$ einer skalarwertigen Funktion $f : \mathbb{R}^n \longrightarrow R$ angegeben. Wir bezeichnen mit $x = (x_1, \ldots, x_n)^T$ den Vektor der unabhängigen Variablen, mit $y = f(x)$ die abhängige Variable, mit $u = (u_1, \ldots, u_n)^T$ einen konstanten Spaltenvektor der Richtung, nach der abgeleitet wird, und mit $H = \nabla^2 f(x)$ wird die Hesse-Matrix der zweiten Ableitungen definiert durch

$$
H_f(x) = \nabla^2 f(x) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(x) \right) \quad (1 \leq i, j \leq n). \tag{2.28}
$$

Um eine effiziente Berechnung von $H = \nabla^2 f(x)$ zu erhalten, werden jetzt der Vorwärtsmodus aus Abschnitt 2.2.3 und der Rückwärtsmodus aus Abschnitt 2.2.4 miteinander kombiniert. Aufgrund der Übersichtlichkeit wird eine leicht modifizierte Notation des Vorwärtsmodus bei der Beschreibung der Gradienten, der Adjungierten und der Indizes der verwendeten Zwischenvariablen verwendet.

Zu Beginn wird der Vorwärtsmodus angewendet, um die Richtungsableitung $\dot{y} = u^T \nabla f$ der skalaren Funktion f zu berechnen. Die für die Berechnung von f benötigten Zwischenvariablen werden im folgenden mit $x_{n+1}, \ldots, x_m$ bezeichnet.

for $i := 1$ **up to** n **do**

  $w_i := u_i$

od

for $i := n + 1$ **up to** m **do**

  $U_i := \{x_j \mid j \prec i\}$

  $W_i := \{w_j \mid j \prec i\}$

Forward Sweep

v_{-1}	:=	$x_1 = T$
v_0	:=	$x_2 = n_1$
v_1	=	$n_{a_1} - v_0$
v_2	=	$n_{e_2} - v_1$
v_3	=	$n_{a_4} + n_{e_4}$
v_4	=	$M_1 * v_0$
v_5	=	$M_2 * v_2$
v_6	=	$M_3 * v_1$
v_7	=	$M_4 * v_3$
v_8	=	$v_4 + v_5$
v_9	=	$v_8 + v_6$
v_{10}	=	$v_9 + v_7$
v_{11}	=	v_{10}/ϱ
v_{12}	=	$1/v_{-1}$
v_{13}	=	$1/T_{ref}$
v_{14}	=	$v_{12} - v_{13}$
v_{15}	=	E_a/R
v_{16}	=	$-v_{15}$
v_{17}	=	$v_{16} * v_{14}$
v_{18}	=	$\exp(v_{17})$
v_{19}	=	$v_{18} * k_{ref}$
v_{20}	=	v_0/v_{11}
v_{21}	=	v_2/v_{11}
v_{22}	=	$v_{19} * v_{20}$
v_{23}	=	$v_{22} * v_{21}$
v_{24}	=	$v_{11} * v_{23}$
v_{25}	=	$-v_{24}$
y_1	=	v_{25}

inkrementeller Modus

$\bar{v}_{25}$	:=	$\bar{y}_1$
$\bar{v}_{24}$	$-=$	$\bar{v}_{25}$
$\bar{v}_{11}$	$+=$	$\bar{v}_{24} * v_{23}$
$\bar{v}_{23}$	$+=$	$\bar{v}_{24} * v_{11}$
$\bar{v}_{21}$	$+=$	$\bar{v}_{23} * v_{22}$
$\bar{v}_{22}$	$+=$	$\bar{v}_{23} * v_{21}$
$\bar{v}_{19}$	$+=$	$\bar{v}_{22} * v_{20}$
$\bar{v}_{20}$	$+=$	$\bar{v}_{22} * v_{19}$
$\bar{v}_2$	$+=$	$\bar{v}_{21}/v_{11}$
$\bar{v}_{11}$	$-=$	$\bar{v}_{21} * v_2/(v_{11} * v_{11})$
$\bar{v}_0$	$+=$	$\bar{v}_{20}/v_{11}$
$\bar{v}_{11}$	$-=$	$\bar{v}_{20} * v_0/(v_{11} * v_{11})$
$\bar{v}_{18}$	$+=$	$\bar{v}_{19} * k_{ref}$
$\bar{v}_{17}$	$+=$	$\bar{v}_{19} * \exp(v_{17})$
$\bar{v}_{16}$	$+=$	$\bar{v}_{17} * v_{14}$
$\bar{v}_{14}$	$+=$	$\bar{v}_{17} * v_{16}$
$\bar{v}_{15}$	$-=$	$\bar{v}_{16}$
$\bar{v}_{12}$	$+=$	$\bar{v}_{14}$
$\bar{v}_{13}$	$-=$	$\bar{v}_{14}$
$\bar{v}_{-1}$	$+=$	$\bar{v}_{12}/(v_{-1} * v_{-1})$
$\bar{v}_{10}$	$+=$	$\bar{v}_{11}/\varrho$
$\bar{v}_7$	$+=$	$\bar{v}_{10}$
$\bar{v}_9$	$+=$	$\bar{v}_{10}$
$\bar{v}_6$	$+=$	$\bar{v}_9$
$\bar{v}_8$	$+=$	$\bar{v}_9$
$\bar{v}_4$	$+=$	$\bar{v}_8$
$\bar{v}_5$	$+=$	$\bar{v}_8$
$\bar{v}_3$	$+=$	$\bar{v}_7 * M_4$
$\bar{v}_1$	$+=$	$\bar{v}_6 * M_3$
$\bar{v}_2$	$+=$	$\bar{v}_5 * M_2$
$\bar{v}_0$	$+=$	$\bar{v}_4 * M_1$
$\bar{v}_1$	$-=$	$\bar{v}_2$
$\bar{v}_0$	$-=$	$\bar{v}_1$
$\bar{x}_2$	=	$\bar{v}_0$
$\bar{x}_1$	=	$\bar{v}_{-1}$

nichtinkrementeller Modus

$\bar{v}_{25}$	:=	$\bar{y}_1$
$\bar{v}_{24}$	=	$-\bar{v}_{25}$
$\bar{v}_{23}$	=	$\bar{v}_{24} * v_{11}$
$\bar{v}_{22}$	=	$\bar{v}_{23} * v_{21}$
$\bar{v}_{21}$	=	$\bar{v}_{23} * v_{22}$
$\bar{v}_{20}$	=	$\bar{v}_{22} * v_{19}$
$\bar{v}_{19}$	=	$\bar{v}_{22} * v_{20}$
$\bar{v}_{18}$	=	$\bar{v}_{19} * k_{ref}$
$\bar{v}_{17}$	=	$\bar{v}_{18} * \exp(v_{17})$
$\bar{v}_{16}$	=	$\bar{v}_{17} * v_{14}$
$\bar{v}_{15}$	=	$-\bar{v}_{16}$
$\bar{v}_{14}$	=	$\bar{v}_{17} * v_{16}$
$\bar{v}_{13}$	=	$-\bar{v}_{14}$
$\bar{v}_{12}$	=	$\bar{v}_{14}$
$\bar{v}_{11}$	=	$\bar{v}_{24} * v_{23}$
		$-\bar{v}_{21} * v_2/(v_{11} * v_{11})$
		$-\bar{v}_{20} * v_0/(v_{11} * v_{11})$
$\bar{v}_{10}$	=	$\bar{v}_{11}/\varrho$
$\bar{v}_9$	=	$\bar{v}_{10}$
$\bar{v}_8$	=	$\bar{v}_9$
$\bar{v}_7$	=	$\bar{v}_{10}$
$\bar{v}_6$	=	$\bar{v}_9$
$\bar{v}_5$	=	$\bar{v}_8$
$\bar{v}_4$	=	$\bar{v}_8$
$\bar{v}_3$	=	$\bar{v}_7 * M_4$
$\bar{v}_2$	=	$\bar{v}_{21}/v_{11} + \bar{v}_5 * M_2$
$\bar{v}_1$	=	$\bar{v}_6 * M_3 - \bar{v}_2$
$\bar{v}_0$	=	$\bar{v}_{20}/v_{11}$
		$+\bar{v}_4 * M_1 - \bar{v}_1$
$\bar{v}_{-1}$	=	$-\bar{v}_{12}/(v_{-1} * v_{-1})$
$\bar{x}_2$	=	$\bar{v}_0$
$\bar{x}_1$	=	$\bar{v}_{-1}$

Abbildung 2.3: Inkrementeller und nichtinkrementeller Rückwärtsmodus bei der Berechnung der rechten Seite einer DAE.

$x_i := \phi_i(U_i)$

$w_i := g_i(U_i, W_i)$

od

$y := x_m = \phi_m(U_m) = f(x_1, \ldots, x_n),\ \dot{y} = w_m = u^T \nabla f(x_1, \ldots, x_n)$

mit

$$g_i(x, w) := \sum_{U_i} \frac{\partial \phi_i\,(U_i)}{\partial x_j} \cdot w_j.$$

Nun wird der Rückwärtsmodus auf das obige Berechnungsschema angewendet, um die zweiten Richtungsableitungen $\nabla \left(u^T \nabla f\right) = \left(\nabla \nabla^T f\right) u = (Hf)\, u$ zu berechnen. Dafür werden die Größen $\partial w_m / \partial x_j$ und $\partial w_m / \partial w_j$ benötigt. Unter Verwendung der Adjungierten $\bar{x}_j$ $(1 \leq j \leq m)$ gilt:

$$\partial w_m / \partial w_j = \bar{x}_j \tag{2.29}$$

Beweis. (durch vollständige Induktion über j)

Induktionsanfang:	$j = m : \partial w_m / \partial w_m = 1 = \bar{x}_m.$
Induktionsannahme:	Die Behauptung gelte für ein beliebiges $j < m$
Induktionsschritt:	$m, m-1, \ldots, j+1 \longrightarrow j$

$$\frac{\partial g_i}{\partial w_j} = \frac{\partial}{\partial w_j} \left(\sum_{U_i} \frac{\partial \phi_i\,(U_i)}{\partial x_j} \cdot w_j \right) = \frac{\partial \phi_i\,(U_i)}{\partial x_j}, \tag{2.30}$$

somit gilt:

$$\frac{\partial w_m}{\partial w_j} = \sum_{i \succ j} \frac{\partial w_m}{\partial w_i} \frac{\partial g_i}{\partial w_j} = \sum_{i \succ j} \bar{x}_i \frac{\partial \phi_i\,(U_i)}{\partial x_j} = \bar{x}_j. \tag{2.31}$$

■

Sei nun $\bar{w}_j := \partial w_m / \partial x_j$ für $1 \leq j \leq m$. Da keines der ϕ_i und g_i die Variable x_m als Argument besitzt, gilt $\bar{w}_m = 0$, und die Anwendung der Kettenregel ergibt:

$$\bar{w}_j = \frac{\partial w_m}{\partial x_j} = \sum_{i \succ j} \left(\bar{x}_i \frac{\partial g_i}{\partial x_j} + \bar{w}_i \frac{\partial \phi_i}{\partial x_j} \right) = \sum_{i \succ j} \left(\bar{x}_i \sum_{k \prec i} w_k \frac{\partial^2 \phi_i}{\partial x_j \partial x_k} + \bar{w}_i \frac{\partial \phi_i}{\partial x_j} \right). \tag{2.32}$$

Das Anwenden der Gleichungen (2.29) und (2.32) ermöglicht somit das Aufstellen des Algorithmus zur Berechnung von $(Hf)\, u$.

Algorithmus 2.2.4 *Berechnung der Hesse-Matrix mit Vorwärts- und Rückwärtsmodus*

1. {*Initialisierung des Forward Sweep*}

   for $i := 1$ **up to** n **do**

   $w_i := u_i$

   od

2. {*Forward Sweep*}

for $i := n+1$ **up to** m **do**
$U_i := \{x_j \mid j \prec i\}$
$x_i := \phi_i(U_i)$
$\bar{x}_i = 0$
$w_i = 0$
$w_i := \sum_{U_i} \frac{\partial \phi_i(U_i)}{\partial x_j} \cdot x_j$
$\bar{w}_i := 0$
od
$x_m = f(x_1, \ldots, x_n), \quad w_m = (u^T \cdot \nabla f)(x_1, \ldots, x_n)$

3. {*Initialisierung des Reverse Sweep*}

$\bar{x}_m := 1$
for $i := 1$ **up to** n **do**
$\bar{x}_i := 0$
$\bar{w}_i := 0$
od

4. {*Reverse Sweep*}

$\bar{x}_m := 1$
for $i := m$ **down to** $n+1$ **do**
for *all* $j \in U_i$ **do**
$\bar{x}_j := \bar{x}_j + \bar{x}_i \frac{\partial \phi_i(U_i)}{\partial x_j}$
$\bar{w}_j := \bar{w}_j + \bar{x}_i \frac{\partial \phi_i(U_i)}{\partial x_j}$
for *all* $k \in U_i$ **do**
$\bar{w}_j := \bar{w}_j + \bar{x}_i w_k \frac{\partial^2 \phi_i(U_i)}{\partial x_j \partial x_k}$
od
od
od

5. {*Output*}

$(\bar{x}_1, \ldots, \bar{x}_n)^T = \nabla f(x_1, \ldots, x_n), \quad (\bar{w}_1, \ldots, \bar{w}_n) = \nabla(u^T \cdot \nabla f)(x_1, \ldots, x_n).$

Bei der Berechnung von $H \cdot u$ mit dem obigen Algorithmus stellt man interessanterweise fest, daß seine Laufzeit unabhängig von der Anzahl n der abhängigen Variablen ist. Es gilt:

$$\frac{OPS\left(f, \nabla f, \nabla^2 f\right)}{OPS\left(f\right)} = \mathcal{O}\left(1\right). \tag{2.33}$$

Setzt man $u = e_i$, d.h. den i-ten Einheitsvektor, so erhält man die i-te Zeile der Hesse-Matrix H von f. Das n-malige Ausführen des obigen Algorithmus ermöglicht somit die Berechnung von H. Dabei können die n Reihen von H unabhängig voneinander ermittelt werden und bieten sich daher für eine parallele Berechnung auf n Prozessoren an. Weiteres über die Möglichkeiten der Parallelisierung findet man in CHRISTIANSON (1992).

Für die Komplexität der Berechnung von H gilt daher:

$$OPS\left(f, \nabla f, \nabla^2 f\right) = \mathcal{O}\left(n\right) \cdot OPS\left(f\right). \tag{2.34}$$

Wendet man hingegen den Vorwärtsmodus zweimal an, um $H \cdot u$ zu ermitteln, so ist die Komplexität nach Abschätzung (2.16) durch $\mathcal{O}\left(n^2\right) \cdot OPS\left(f\right)$ gegeben.

Da in der Praxis für die Basisfunktionen ϕ_i nur elementare Operationen bzw. mathematische Bibliotheksfunktionen verwendet werden, läßt sich für die Gleichung (2.34) nach LOHNER (1993) die folgende Approximation machen:

$$\frac{OPS\left(f, \nabla f, \nabla^2 f\right)}{OPS\left(f\right)} \approx 11 \cdot n \tag{2.35}$$

Im folgenden wird ein Beispiel nach CHRISTIANSON (1992) betrachtet, um anzudeuten, daß der obige Algorithmus bis auf einen konstanten Faktor optimal ist. Die Funktion $f = f(x_1, \ldots, x_n)$ sei definiert durch:

$$f(x) = \left(x^T x\right)^2$$

Dann gilt

$$\nabla f(x) = 4(x^T x)x \quad \text{und} \quad Hf(x) = 4(x^T x)I_n + 8xx^T.$$

Bezeichne a die Kosten für eine Addition und t für eine Multiplikation, so ist der Aufwand für f gleich $(n+1)\cdot t + (n-1)\cdot a$. Die Berechnung von ∇f benötigt zusätzlich $2(n+1)\cdot t + 4n \cdot a$ Operationen. Der zusätzliche Aufwand zur Evaluation von Hf ist dabei mindestens $\frac{1}{2} \cdot n(n+1) \cdot t$.

2.2.7 Ausnutzung der Dünnbesetztheit von Jacobi-Matrizen

In den vorherigen Abschnitten wurde gezeigt, daß die Komplexität der Berechnung der Jacobi-Matrix J linear mit n (Vorwärtsmodus) oder m (Rückwärtsmodus) im Verhältnis zur Funktionsauswertung ansteigt. Falls das Dünnbesetztheitsmuster (*sparsity pattern*) von J a priori bekannt ist, läßt sich diese Struktur zur effizienteren Berechnung von J ausnutzen.

Nach CURTIS ET AL. (1974) läßt sich erkennen, daß zwei Spalten einer $(m \times n)$-Jacobi-Matrix $J_j(x) = J(x)e_j$ und $J_k(x) = J(x)e_k$ mit $x \in \mathbb{R}^n$, die *strukturell orthogonal*[3] zueinander sind, simultan aus der Berechnung einer einzigen Richtungsableitung $J_{j,k} := J(x)(e_j + e_k)$ ermittelt werden können. Die Berechnung von J_j aus $J_{j,k}$ ist dann trivial. Sei $\mathcal{N}_j$ die Menge der Zeilenindizes aller Nichtnullelemente von J_j, so gilt für die einzelnen Elemente $(J_j)_i$ $(i = 1, \ldots, m)$ des Spaltenvektors J_j:

$$(J_j)_i(x) = \begin{cases} (J_{j,k})_i(x), & \text{falls } i \in \mathcal{N}_j \\ 0, & \text{sonst.} \end{cases}$$

Nach diesem Prinzip läßt sich unter Verwendung des Dünnbesetztheitsmusters aus J eine $(m \times p)$-Matrix P mit $p \leq n$ konstruieren. Jede Spalte von P repräsentiert dabei eine Gruppe von Spalten, die paarweise strukturell orthogonal zueinander sind. Die n Spalten von J werden in p Gruppen $G_1, G_2, \ldots, G_p$ aufgeteilt, so daß jede Spalte in nur einer Gruppe ist, und die einzelnen Spalten einer Gruppe strukturell orthogonal zueinander sind. Diese Vorgehensweise führt somit zu einem Partitionierungsproblem für die n Spalten von J. Um den Rechenaufwand zu minimieren, sollte dabei die Anzahl p der Gruppen möglichst niedrig sein.

Das Problem der Partitionierung der Spalten ist äquivalent zum Problem der Färbung eines Graphen $G(V, E)$. Die Knoten V des Graphen sind dabei die Spalten von J. Ein Tupel $(v_j, v_k) \in V \times V$ gehört dabei zu E genau dann, wenn die Spalten J_j und J_k von J nicht strukturell orthogonal zueinander sind. Das Auffinden einer minimalen p-Färbung und somit der chromatischen Zahl $\chi(G)$ aus Abschnitt (2.1.3) ist bekanntlich ein NP-schweres Problem. Heuristische Ansätze für dieses Problem, die ursprünglich bei der Berechnung mit Differenzenquotienten (siehe Abschnitt 2.3) angewendet wurden, finden sich bei CURTIS ET AL. (1974) sowie bei NEWSAM & RAMSDELL (1983).

Mit den Definitionen aus Abschnitt 2.1.4 gilt weiter:

Korollar 2.2.5 *Die Größe jeder Clique G' von G ist eine untere Schranke für die chromatische Zahl χ eines Graphen G. Für das obige Spaltenpartitionierungsproblem gilt daher im speziellen: Sei J eine $m \times n$-Matrix und $\tilde{n}_i$ $(i = 1, \ldots, m)$ die Anzahl der Nichtnullelemente der Zeile i, dann gilt:*

$$\max_{1 \leq i \leq m} \tilde{n}_i \leq \chi(G(J)) \tag{2.36}$$

Beweis. Sei J eine $m \times n$-Matrix. Ist $G' = (V', E')$ eine Clique von $G(J)$, dann sind trivialerweise alle Knoten von V' paarweise adjazent und $\chi(G(J))$ ist mindestens so groß wie die Anzahl der Knoten in V'. Durch die Teilmengen V' mit $V_i = \{J_j : J_{ij} \neq 0\}$ werden Cliquen in $G(J)$ erzeugt, wobei jede Clique G_i über $\tilde{n}$ $(i = 1, \ldots, m)$ Knoten verfügt. ∎

2.3 Das Numerische Differenzieren

In vielen numerischen Programmen wird zur Berechnung von Richtungsableitungen der Differenzenquotient verwendet. Man unterscheidet zwischen dem einseitigen und dem zweiseitigen Differenzenquotienten.

[3] Zwei Vektoren $v, w \neq 0$ aus dem $\mathbb{R}^n$ werden als strukturell orthogonal bezeichnet, falls $v_i \cdot w_i = 0$ für alle $i = 1, \ldots, n$.

Die Richtungsableitung von F nach $x \in R^n$ in der Richtung $\nu \in I\!R^n$ ergibt sich mittels des einseitigen Differentenquotienten zu:

$$\frac{\partial F}{\partial x}(x) \cdot \nu = \lim_{h \to 0} \frac{F(x + h \cdot \nu) - F(x)}{h} \tag{2.37}$$

Mittels des zweiseitigen Differenzenquotienten zu:

$$\frac{\partial F}{\partial x}(x) \cdot \nu = \lim_{h \to 0} \frac{F(x + h \cdot \nu) - F(x - h \cdot \nu)}{2 \cdot h} \tag{2.38}$$

Da man auf einem Rechner h nicht beliebig klein machen kann, ist diese Approximation immer fehlerbehaftet. Die Schrittweite h ist durch die Maschinengenauigkeit ε immer nach unten beschränkt.

Die Komplexität der Berechnung einer Richtungsableitung mit dem Differenzenquotient ist aufgrund der Gleichungen (2.37) und (2.38) trivialerweise durch $2 \cdot OPS(F)$ beim einseitigen Differenzenquotienten bzw. $3 \cdot OPS(F)$ beim zweiseitigen Differenzenquotienten gegeben, falls die Funktion F auch berechnet werden muß. Die Komplexität ist im Gegensatz zum AD unabhängig von der zu berechnenden Funktion F und der Richtung nach der abgeleitet wird.

2.3.1 Fehleranalyse

Eine kleine Schrittweite h wird benötigt, um den Abbruchfehler bei der Berechnung von Ableitungen durch Differenzenquotienten zu minimieren. Doch bei Werten von h, die sehr nah an der Maschinengenauigkeit ε liegen, kann die Subtraktion von zwei fast gleichen Gleitkommazahlen zu signifikanten Auslöschungsfehlern führen. Die daraus resultierenden Rundungsfehler im Zähler von Gleichung (2.37) können sehr groß sein und führen somit zu falschen Ableitungswerten. Das Numerische Differenzieren ist nach STOER & BULIRSCH (1971) wenig stabil und ist nach GRIEWANK & CHRISTIANSEN (1998) bei dritten oder höheren Ableitungen aufgrund der auftretenden Rundungsfehler praktisch nutzlos.

Sei $gl(F(x))$ die Darstellung von $F(x)$ in Gleitkommaarithmetik, so existiert nach STOER & BULIRSCH (1971) ein $0 < \delta < \varepsilon$ mit:

$$|gl(F(x)) - F(x))| \leq \delta. \tag{2.39}$$

Nach dem Satz von Taylor gilt für den eindimensionalen Fall:

$$\begin{aligned} F(x + \nu \cdot h) &= \sum_{k=0}^{\infty} \frac{F^k(x)}{k!} (\nu h)^k \\ &= F(x) + (\nu h) F'(x) + \frac{(\nu h)^2}{2} F''(x) + \frac{(\nu h)^3}{6} F'''(x) + \mathcal{O}\left(h^4\right) \end{aligned} \tag{2.40}$$

$$\begin{aligned} F(x-\nu\cdot h) &= F(x+(-\nu\cdot h)) = \sum_{k=0}^{\infty}\frac{F^k(x)}{k!}(-\nu h)^k \\ &= F(x)-\nu hF'(x)+\frac{(\nu h)^2}{2}F''(x)-\frac{(\nu h)^3}{6}F'''(x)+\mathcal{O}\left(h^4\right) \end{aligned} \tag{2.41}$$

Damit gilt für den einseitigen Differenzenquotienten (2.37):

$$\begin{aligned} F_{D_1} &:= \frac{1}{h}\left(F(x+\nu\cdot h)-F(x)\right) = F'(x)\nu+\frac{h}{2}F''(x)\nu^2+\mathcal{O}\left(h^2\right) \\ &= F'(x)+\mathcal{O}\left(h\right) \end{aligned} \tag{2.42}$$

Somit folgt für den Verfahrensfehler Δ_F als Summe von Rundungsfehler Δ_{F_1} und Abbruchfehler Δ_{F_2} mit der Ungleichung (2.39) und der Abschätzung $K \geq \left|F''(x)\nu^2\right|$:

$$\begin{aligned} \Delta_F &= \left|\frac{1}{h}\left(gl(F(x))-gl(F(x+h\cdot\nu))\right)-F'(x)\cdot\nu\right| \leq \Delta_{F_1}+\Delta_{F_2} \\ &\leq \underbrace{\frac{2\delta}{h}+K\cdot h}_{:=s_1(h)} \end{aligned} \tag{2.43}$$

Die obere Schranke $s_1(h)$ erreicht ihr Minimum bei $h=\sqrt{\frac{2\delta}{K}}$. Somit ergibt sich für den Verfahrensfehler Δ_F beim einseitigen Differenzenquotienten:

$$\Delta_F \leq \frac{2\delta}{\sqrt{\frac{2\delta}{K}}}+K\cdot\sqrt{\frac{2\delta}{K}} = \sqrt{\frac{K}{2\delta}}\left(2\delta+2\delta\right) = 4\delta\cdot\sqrt{\frac{K}{2\delta}} = 2\sqrt{2K}\cdot\sqrt{\delta}. \tag{2.44}$$

Selbst bei einer optimalen Wahl von ε verliert man aufgrund des Faktors $\sqrt{\delta}$ für $K \approx 1$ ungefähr die Hälfte der signifikanten Stellen.

Analog gilt für den zweiseitigen Differenzenquotienten (2.38):

$$\begin{aligned} F_{D_2} &:= \frac{1}{2h}\left(F(x+\nu\cdot h)-F(x-\nu\cdot h)\right) = F'(x)\nu+\frac{h^2}{6}F'''(x)\nu^2+\mathcal{O}\left(h^4\right) \\ &= F'(x)\nu+\mathcal{O}\left(h^2\right) \end{aligned} \tag{2.45}$$

Für den Verfahrensfehler Δ_F mit $K \geq \left|F'''(x)\nu^2\right|$ folgt:

$$\begin{aligned} \Delta_F &= \left|\frac{1}{h}\left(gl(F(x+h\cdot\nu))-gl(F(x-h\cdot\nu))\right)-F'(x)\cdot\nu\right| \leq \Delta_{F_1}+\Delta_{F_2} \\ &\leq \underbrace{\frac{2\delta}{h}+K\cdot h^2}_{:=s_2(h)} \end{aligned} \tag{2.46}$$

Die obere Schranke $s_2(h)$ erreicht ihr Minimum bei $h=\left(\frac{\delta}{K}\right)^{\frac{1}{3}}$. Es ergibt sich für den Verfahrensfehler Δ_F beim zweiseitigen Differenzenquotienten

$$\Delta_F \leq \frac{2\delta}{\left(\frac{\delta}{K}\right)^{\frac{1}{3}}} + K \cdot \left(\frac{\delta}{K}\right)^{\frac{2}{3}} = \left(\frac{\delta}{K}\right)^{-\frac{1}{3}} \cdot (2\delta + \delta) = K^{\frac{1}{3}} \cdot \frac{3\delta}{\delta^{\frac{1}{3}}} = 3K^{\frac{1}{3}} \cdot \delta^{\frac{2}{3}} \quad (2.47)$$

Da $\delta^{\frac{2}{3}} < \delta^{\frac{1}{2}}$ für $0 < \delta < 1$, ist der Gesamtfehler beim zweiseitigen Differenzenquotienten somit kleiner als beim einseitigen. Anschaulich zeigen dies die Abbildung 2.4 und 2.5 aus Abschnitt 2.3.2.

2.3.2 Gesamtfehler beim Differenzenquotienten

Nach Abschnitt 2.3.1 kann man h nicht beliebig klein machen, um den Gesamtfehler zu minimieren. Im folgenden wird anhand eines praktischen Beispiels untersucht, inwieweit sich die Genauigkeit des Numerischen Differenzierens in Abhängigkeit der Schrittweite h verhält.

Als Testfunktion wurde eine Funktion $F : U \subset \mathbb{R}^2 \longrightarrow \mathbb{R}^3$ mit:

$$F(x_1, x_2) := \begin{pmatrix} x_1^2 \cdot x_2 + 0.1 \cdot \exp(2x_1 + x_2) \\ x_1 \cdot x_2 + \sin x_1^2 \\ \sqrt{x_1}/x_2 \end{pmatrix}, \quad x_1, x_2 > 0$$

ausgewählt. Ausgewertet wird der Funktionswert und die erste Ableitung dabei an der Stelle $(x_1, x_2)^T = (2.0, 3.0)^T$. Die Maschinengenauigkeit des Rechners (AMD-K6/233 unter Linux 2.0.35) betrug dabei $\varepsilon = 2.2204460 \cdot 10^{-16}$. Die Abbildungen 2.4 und 2.5 zeigen die Genauigkeit der Ableitungen für die Richtungen $(2.3, 3.3)^T$ bei der Berechnung mit Hilfe des einseitigen bzw. zweiseitigen Differenzenquotienten in Abhängigkeit der Schrittweite h. Für den Gesamtfehler $e(h)$ und für die Schrittweite h wird die logarithmische Skala verwendet. Die numerisch berechneten Werte werden dabei mit den analytisch berechneten, korrekten Ableitungswerten verglichen. Das für die Berechnung der Testwerte verwendete C++-Programm folgt anschließend als Listing. Folgende Schlußfolgerungen können aus dem Test gezogen werden:

- Der Verlauf der Fehlerkurven ist unabhängig von der gewählten Funktion bzw. den Richtungen, nach denen abgeleitet wird.
- Der optimale Wert für h liegt bei allen drei Funktionen bei $\approx 10^{-8}$ beim einseitigen Differenzenquotienten bzw. bei $\approx 10^{-6}$ beim zweiseitigen Differenzenquotienten.
- Die Genauigkeit der berechneten Ableitungen ist bei der Verwendung des zweiseitigen Differenzenquotienten deutlich höher als beim einseitigen. Sie liegt bei einer optimaler Wahl von h bei $\approx 10^{-12}$ merklich höher als beim einseitigen Differenzenquotienten mit $\approx 10^{-8}$.
- Vom Iterationsbeginn ($h = 1$) bis zum optimalen Wert nimmt der absolute Fehler linear ab. Hier wirkt der systematische mit fallendem h kleiner werdende Abbruchfehler.

- Das Rauschen der Werte ab dem optimalen Wert für h und die Zunahme des absoluten Fehlers läßt sich mit dem zufällig auftretenden Rundungsfehler erklären.
- Die Tests bestätigen die beiden oberen Schranken (2.44) und (2.47) für die untersuchten Varianten.

Die beiden Abbildungen 2.4 und 2.5 zeigen darüber hinaus, daß die Wahl *eines* h für eine vektorwertige Funktion bei stark unterschiedlichen höheren Ableitungen zu unterschiedlichen Genauigkeiten in den einzelnen Komponenten der Ableitung führt.

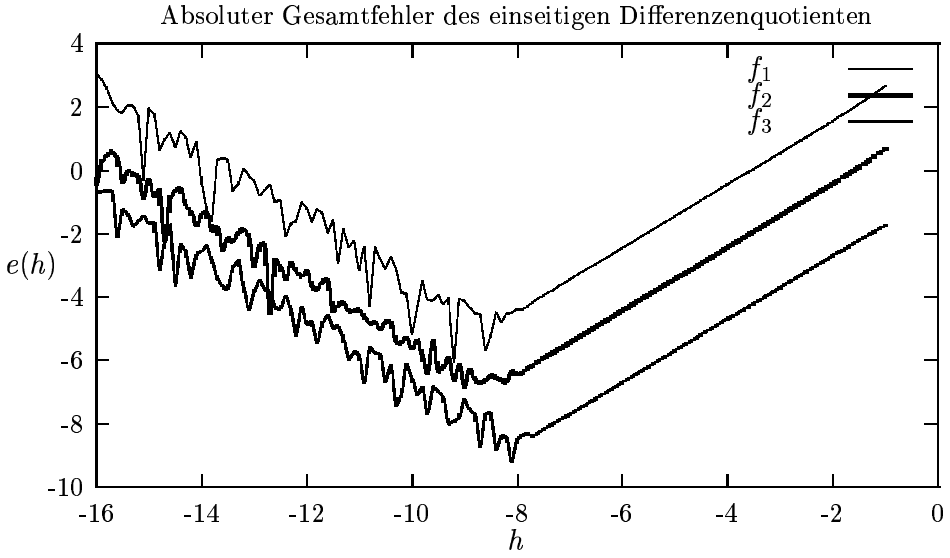


Abbildung 2.4: Gesamtfehler beim einseitigen Differenzenquotient

Die Berechnung von Richtungsableitungen mit Numerischer Differentiation hat den Vorteil, daß man die abzuleitende Funktion als *Black-Box* betrachten kann. Dies bedeutet, daß ein Parsen und eine Zerlegung des mathematischen Ausdrucks in elementare Operationen nicht benötigt wird, sondern nur ein weiterer Funktionsaufruf an einer Störstelle $x + h$. Diese Methode wird weiterhin eine Berechtigung haben, wenn die Exaktheit der berechneten Ableitungen, im Vergleich zum Aufwand für das AD, eine untergeordnete Rolle spielt. Vor allem Programme in C und C++ mit ihren vielfältigen Möglichkeiten, selbstdefinierte Datentypen in die Funktionsauswertung einfließen zu lassen, können mitunter nur schwer für die Verwendung von AD mit Quelltexttransformation umgeschrieben werden.

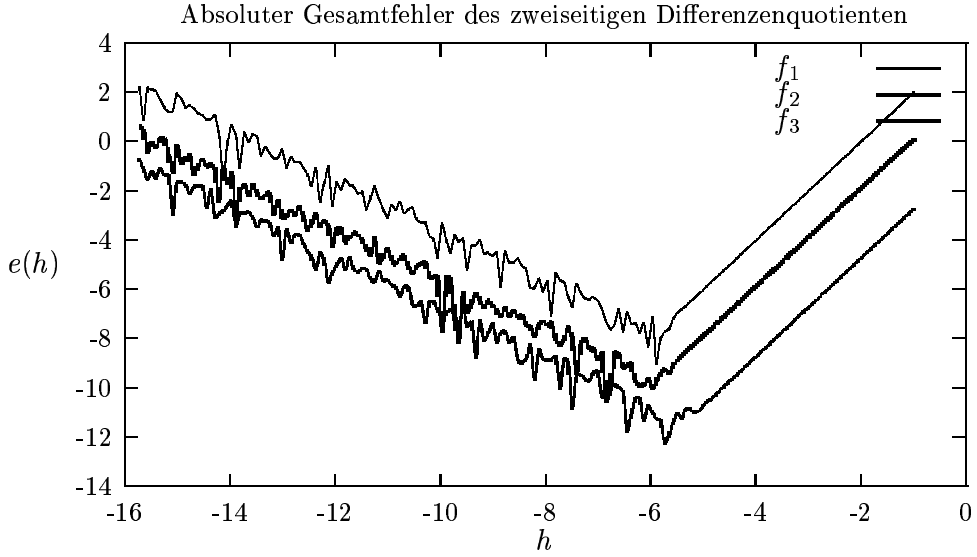


Abbildung 2.5: Gesamtfehler beim zweiseitigen Differenzenquotient

Listing zur Berechnung des Verfahrensfehler bei Numerischen Differenzen

```
// Rundungsfehler beim Numerischen Differenzieren mit Differenzenquotienten

#include <math.h>                 // Trigonometrische Funktionen
#include <stdio.h>                // Ausgabe in Datei
#include <float.h>                // Bestimmung der Maschinengenauigkeit
#define FORMAT "%25.22f "         // Format der Float Ausgabe
#define STEPSIZE 0.08             // Logarithmische Schrittweite
#define N_X 2                     // Anzahl der unabhaengigen Variablen
#define N_F 3                     // Anzahl der abhaengigen Variablen

double fcn1( double *x ){ return x[0]*x[0]*x[1] + 0.1 * exp(2* x[0] + x[1] ); }
double fcn2( double *x ){ return x[0] * x[1] + sin( x[0] * x[0] ); }
double fcn3( double *x ){ return sqrt( x[0] ) / x[1]; }

// Ableitungen
double dfcn1( double *x, double *d ){
       return ( 2*x[0]*x[1] + 2*0.1*exp( 2*x[0]+x[1] ) ) * d[0] + ( x[0]*x[0]
                            + 0.1 * exp( 2*x[0] + x[1] ) ) * d[1];
}
double dfcn2( double *x, double *d ){
    return (x[1] + 2*x[0] * cos( x[0] * x[0] ) ) * d[0] + ( x[0] ) *  d[1] ;
}
double dfcn3( double *x, double *d ) {
  return (1 / ( 2* sqrt( x[0] ) * x[1] ) ) * d[0]
                - ( sqrt( x[0] ) / ( x[1] * x[1] ) ) * d[1];
}

// Array von Funktionspointern aller Testfunktionen
double (*fcn[])( double *x ) = { fcn1, fcn2, fcn3 };
double (*dfcn[])( double *x, double *d ) = { dfcn1, dfcn2, dfcn3 };

// einseitige und zweiseitige Differenzenquotientenberechnung
double DiffQuot( double *x, double (*fcn) ( double* ), double h, double *d, int mode ){
     double f_arg1[N_X], f_arg2[N_X], one_side, two_side;
     for( int i = 0; i < N_X; i++ )
          f_arg1[i] = x[i] + h*d[i], f_arg2[i] = x[i] - h*d[i];
```

```
     one_side = ( fcn( f_arg1 ) - fcn( x ) ) / h ;             // einseitiger DQ
     two_side = ( fcn( f_arg1 ) - fcn( f_arg2 ) ) / ( 2 * h ); // zweiseitiger DQ
     return  mode == 0  ? one_side : two_side;
}

int main( int argc, char **argv ){
     FILE *fp;
     double x[N_X], d[N_X], f_d_val[N_F], f_d_val_c[N_F], error[N_F];
     register double h = 1.0, step = 1.0, eps = DBL_EPSILON;

     if( !( fp = fopen( argv[1], "w" ) ) )
          exit( -1 );
     x[0] = 2.0, x[1] = 3.0, d[0] = 2.3, d[1] = 3.3;
     do{    // Iteration mit Schrittweite h
          h = pow( 10, -step );
          fprintf( fp, FORMAT, log10( h ) );
          for( register int i = 0; i < N_F; i++ ){
            f_d_val[i] = DiffQuot( x, fcn[i], h, d, 1 );
            f_d_val_c[i] = dfcn[i]( x, d );
            error[i] = fabs( f_d_val[i] - f_d_val_c[i] );
            fprintf( fp, FORMAT, log10( error[i] ));
          }
          fputc( '\n', fp );
          step += STEPSIZE;
     }
     while( h >= eps );
     if( fclose( fp ) )
          exit( -1 );
     return 0;
}
```

2.4 Weitere Formen der Ableitungsbestimmung

Im Bereich des *Symbolischen Differenzierens* existieren einige Software-Pakete, die aus einem gegebenen algebraischen Ausdruck die Ableitung berechnen können. Beispiele hierfür sind MAPLE, MATHEMATICA, AXIOM, MACSYMA und DERIVE. Dabei wird durch Anwendung der Differentiationsregeln ein expliziter Ausdruck für die Ableitungen generiert. Diese Ausdrücke werden jedoch sehr schnell sehr groß.

Das grundsätzliche Problem beim Symbolischen Differenzieren liegt darin, daß jede binäre Operation (außer '+' und '−') bei der Ableitungsgenerierung zu einem größeren Term anwächst. Dies führt bei der Generierung des Ableitungsterms pro Ableitungsordnung zu einer Vervielfachung der Ausdrucksgröße und vor allem bei Problemen höherer Dimension, wie sie in der Praxis auftreten, zu einem hohen Maß an Speicherbedarf und Laufzeit. Weitere Nachteile liegen darin, daß obige Software-Pakete meistens nicht in der Lage sind, prozedural definierte Funktionen sowie typische Quelltext-Konstrukte wie beispielsweise `if-then-else, for,` oder `while` zu behandeln.

Eine weitere Möglichkeit, Ableitungen in einem Programm bereitzustellen, ist das Aufstellen von Ableitungsformeln und die Generierung des korrespondierenden Quelltexte *von Hand*. Zwangsläufig führt dieses Verfahren zu einem hohen Aufwand in puncto Programmieraufwand, Modifikation sowie Verwaltung des Quelltextes. Außerdem müssen schon bei kleinsten Veränderungen des abzuleitenden Programms die Ableitungen neu ermittelt und implementiert werden. Für laufzeitkritische Abschnitte eines Programms bietet dieses flexible Verfahren aber eine Möglichkeit, strukturelle Gegebenheiten am effizientesten

auszunutzen. Bei der Codierung der Ableitungen können dabei die Prinzipien des AD angewendet werden.

2.5 AD Implementierungen

2.5.1 ADIFOR

ADIFOR ist ein general-purpose System zum automatischen Differenzieren von Funktionen, die in Form eines Quelltextes in der Programmiersprache Fortran77[1] spezifiziert werden. Das Programm-Paket wurde am Aragonne National Laboratory[2] (Mathematics and Computer Science Division) und an der Rice University[3], Houston (Center for Research on Parallel Computation) entwickelt. Als wichtige Koordinatoren in einem größeren Entwicklungsteam sind Christian Bischof und Alan Carle zu nennen. Der Quelltext des abzuleitenden Programms darf dabei aus allen Syntaxelementen bestehen, wie z.B. Schleifen und Iterationen, die in der Programmiersprache Fortran77 vorhanden sind. Das System basiert auf dem Prinzip der *Quelltexttransformation*. Um die Ableitung einer durch ein Computerprogramm definierten Funktion zu berechnen, wird der zugrundeliegende Quelltext entsprechend modifiziert. Durch den Aufruf neu generierter Unterprogramme oder neu generierter Schleifen wird dann die Ableitung der betrachteten Funktion berechnet. Mit Hilfe des von ADIFOR erzeugten Programmes lassen sich mit einem einzigen Funktionsaufruf beliebig viele Richtungsableitungen berechnen. Seien im folgenden n die Anzahl der unabhängigen Variablen, m die Anzahl der abhängigen Variablen sowie p die Anzahl der Richtungen, die berechnet werden.

Die prinzipielle Vorgehensweise bei der Ableitungserzeugung mit ADIFOR ist nach Bischof et al. (1992) gegeben durch:

- Spezifikation der Benutzereingabe:
  - Fortran77 Subroutine `ffcn`, in der die abzuleitende Funktion implementiert ist.
  - Alle Subroutines, die direkt oder indirekt von `ffcn` aufgerufen werden.
  - Namen der Vektoren der unabhängigen (`x`) und abhängigen Variablen (`y`).[1]
  - Maximale Anzahl der Richtungen (`max_p`), nach denen abgeleitet werden soll.
- Analyse der Quelltexte durch den in ADIFOR integrierten Parser:
  - Aufstellen des Aufrufgraphen der einzelnen Subroutines.
  - Ermittlung des Abhängigkeitsgraphen aus Abschnitt 2.1.4, der die inner- und zwischenprozeduralen Abhängigkeiten aller Variablen repräsentiert.
  - Bestimmung der *aktiven* Variablen. Eine skalare Variable `v` der Berechnungsprozedur wird dabei als aktiv bezeichnet, wenn sie direkt oder indirekt von mindestens einer Komponente von `x` abhängt und wenn mindestens eine Komponente von `y` existiert, die direkt oder indirekt von `v` abhängt.

[1] Es existiert auch ein Paket, das Programme in der Programmiersprache C ableitet, ADIC genannt.
[2] http://www-unix.mcs.anl.gov/autodiff/ADIFOR/
[3] http://www.cs.rice.edu/~adifor/
[1] An die Vektoren sind dabei keine Restriktionen gestellt. Bei beiden Vektoren kann es sich sowohl um lokale Variablen, Subroutine-Parameter als auch Common-Block Variablen handeln.

  - Zerlegung aller Zuweisungen der betrachteten Subroutines in elementare binäre Zuweisungen durch die Verwendung zusätzlicher Zwischenvariablen.

- Erzeugen des Ableitungsprogramms:
  - Für alle aktiven Variablen werden Anweisungen zur Speicherplatzallokation von Vektoren der Größe `max_p` generiert.[2]
  - Der originale Quelltext wird nun nach den Prinzipien des AD um Ableitungsanweisungen erweitert. Die einzelnen Zuweisungen werden mit Hilfe des effektiveren Rückwärtsmodus abgeleitet. Die Berechnung über das ganze Programm erfolgt im Vorwärtsmodus (siehe dazu die Abschnitte 2.2.3 und 2.2.4).
  - Die Parameterlisten aller Subroutines werden dabei erweitert mit:
    * Einer `INTEGER` Variablen `p`, die die Anzahl der Richtungen angibt.
    * Einer Richtungsmatrix `g_x`, auch *Seed-Matrix* genannt, mit den einzelnen Richtungen als Zeilen.
    * Einer Ableitungsmatrix `g_y`, in deren Spalten die berechneten Richtungsableitungen abgespeichert werden.
    * Den *führenden Dimensionen* (*Leading Dimensions*) der obigen Matrizen.

Der von ADIFOR erzeugte Quelltext berechnet somit die transponierte Richtungsmatrix:

$$\mathtt{g\_y} = (\frac{dy}{dx} \cdot \mathtt{g\_x}^T)^T \in I\!R^{p \times m}$$

Besteht die Seed-Matrix `g_x` beispielsweise aus den n Einheitsrichtungen e_i $(1 \leq i \leq n)$ in den Zeilen, so wird die komplette Jacobi-Matrix berechnet. Die meiste Zeit des generierten Programms wird in den Schleifen der einzelnen Gradientenberechnungen (siehe Gleichung (2.14)) verbraucht. Diese Schleifen laufen von 1 bis p. Die Laufzeit und der Speicherplatz für das erzeugte Ableitungsprogramm ist somit proportional zu p. Dies entspricht auch der theoretischen Abschätzung (2.13).

Behandlung dünnbesetzter Matrizen

Bei einer konkreten Ableitungserzeugung kann sowohl die Jacobi-Matrix J als auch die Richtungsmatrix `g_x` dünn besetzt (*sparse*) sein. Um dies zu einer Optimierung der Laufzeit auszunutzen, lassen sich Modifikationen bei der Anwendung von ADIFOR machen.

Für dünnbesetzte Seed-Matrizen läßt sich die SparsLinC-Bibliothek (siehe BISCHOF ET AL. (1995)) verwenden. Hierbei werden keine *vollen* Richtungen durch `g_x` an die zu berechnende Subroutine übergeben, sondern für jede Richtung $d_i \in I\!R^n$ wird ein Dünnbesetztheitsmuster (*Sparsity Pattern*) mit Hilfe zweier Arrays (*Index Array* und *Value Array*) spezifiziert. Neben der in Fortran77 üblichen Konvention der Speicherung der Indizes aller Nichtnullelemente im Index Array und der korrespondierenden Werte im Value Array, wird bei der *SparsLinC Bibliothek* auch eine zweite strukturausnutzende Abspeicherung

[2] Da die Programmiersprache Fortran77 keine dynamische Speicherallokation besitzt, kann die Größe dieser Vektoren nicht erst zur Laufzeit bestimmt werden, sondern muß bereits zur Übersetzungszeit bekannt sein.

verwendet (vergleiche BISCHOF ET AL. (1995)). Das Index Array ist dabei doppelt indiziert. Jeder Index besteht aus einem 2-Tupel von Indizes, die die sogenannte *Range* von Nichtnulleinträgen spezifizieren. Das Value Array beinhaltet die korrespondierenden Werte, die durch das erste Array festgelegt werden.

Betrachtet wird als Beispiel folgender Vektor:

$$d = (0.0, 1.0, 2.0, 0.0, 0.0, 0.0, 4.0, 0.0, 4.0, 2.0, 4.0, 5.0, 0.0, 0.0)^T \in I\!R^{14}$$

Die beiden zugehörigen Arrays sind somit:

Index Array:	(2,3)	(7,7)	(9,12)

Value Array:	1.0	2.0	4.0	4.0	2.0	4.0	5.0

Diese Abspeicherung ist in puncto Speicherbedarf effizienter als die Variante mit dem einfach-indizierten Integer Array, falls der dünnbesetzte Vektor eine große Anzahl langer, aufeinanderfolgender von Null verschiedener Einträge besitzt. Die Abspeicherung in einer der beiden Varianten wird bei der *SparsLinC* in Abhängigkeit von der Anzahl der Nichtnulleinträge eines Vektors automatisch durchgeführt.

Die theoretischen Überlegungen aus Abschnitt 2.2.7 können nun genutzt werden, um die Dünnbesetztheit der Jacobi-Matrix J bei der Ableitungserzeugung effizient auszunutzen. Anstelle einer kompletten Einheitsmatrix werden nun strukturell orthogonale Spalten von J durch geeignete Wahl der Seed-Matrix mit Hilfe einer einzigen Richtung berechnet.

Behandlung von nichtdifferenzierbaren Funktionen

In der Praxis können Situationen auftreten, in denen man auf nichtdifferenzierbare Funktionen trifft. Zum Beispiel ist die Ableitung von `sqrt(x)` an der Stelle `x = 0` nicht definiert, obwohl die Funktion selbst an dieser Stelle definiert ist. ADIFOR verwendet hierbei das *ADIntrinsics 1.0 Exception Handling*, um solche und ähnlich auftretende Ausnahmefälle bei der Ableitungserzeugung abzufangen.

Fazit und Beurteilung

Mit Hilfe von ADIFOR lassen sich ohne Modifikation des eigenen Quelltextes Richtungsableitungen von beliebigen Fortran77-Programmen sehr effizient und einfach berechnen. Aufgrund des verwendeten Richtungsableitungskonzepts können auch einzelne Richtungsableitungen ermittelt werden, ohne die Berechnung der kompletten Jacobi-Matrix. Durch die Berücksichtigung der Matrix-Vektor-Produktbildung während der Ableitungsberechnung ist der Aufwand wesentlich kleiner, als wenn man erst die komplette Jacobi-Matrix berechnet und sie anschließend mit einem Richtungsvektor multipliziert.

Das wiederholte Anwenden von ADIFOR in der vorliegenden Version auf die jeweils neu erzeugten Subroutines ermöglicht die Berechnung höherer Ableitungen mit wachsenden Parameterlisten. Dieses Problem wurde bei der Erstellung der Versuchsplanungssoftware VPLAN98 dadurch umgangen, daß die Initialisierung und die Aufrufe von ADIFOR für die einzelnen Richtungsableitungen automatisch von einem Treibermodul (DOIT) durchgeführt werden. Werden erste Ableitungen mit Hilfe der *SparsLinC* Bibliothek erzeugt,

so beinhaltet der generierte Quelltext zusätzlich Aufrufe von Bibliotheksfunktionen, zu denen kein Quelltext vorhanden ist. Die wiederholte Anwendung von ADIFOR auf diese Quelltexte ist daher nicht möglich.

Da in der Versuchsplanung (siehe Abschnitt 1.3.4) auch zweite Ableitungen der Modellfunktionen berechnet werden müssen, kann durch die Verwendung von ADIFOR die Dünnbesetztheit der Richtungsmatrizen nicht effizient ausgenutzt werden. Dies ist vor allem bei den Ableitungen nach den Steuergrößen q (siehe Abschnitt 3.1.3) von großem Nachteil, da hier in der Regel sehr große Einheitsmatrizen als Seed-Matrizen verwendet werden. Deshalb wurde in dieser Arbeit in Kapitel 4 ein anderer Ansatz gewählt, um die Laufzeit der generierten Richtungsableitungen zu optimieren.

2.5.2 ADOL–C

Die meisten modernen Programmiersprachen wie C++, Ada, Pascal-XSC oder Fortran90 unterstützen die Möglichkeit, nicht nur Funktionen, sondern auch Operatoren zu überladen.[3] Man spricht dabei von *Überladen*, wenn man zu einer bereits definierten Funktion (Operator) eine neue Funktion (Operator) mit gleichem Namen aber unterschiedlicher Parameterliste oder Rückgabewert implementiert. Dies bedeutet, daß die Bedeutung von arithmetischen Operationen ('+', '$*$') vom Programmierer umdefiniert werden kann. Damit läßt sich im folgenden mit einer Operation die Multiplikation und deren Ableitung berechnen.

In einer objektorientierten Programmiersprache wie C++ lassen sich benutzereigene Datentypen in Form von Klassen definieren. Beispielsweise läßt sich eine Klasse `DoubleDiff` implementieren, die einen Fließkommawert und dessen erste und zweite Ableitung repräsentiert:

```
class DoubleDiff{
   double v;
   double dv;
   double d2v;
};
```

Die Multiplikation und die gleichzeitige Ableitung der Multiplikation zweier Instanzen der Klasse `DoubleDiff` kann nun wie folgt realisiert werden, indem der '$*$'-Operator überladen wird.

```
DoubleDiff operator* ( DoubleDiff x, DoubleDiff y ){
   DoubleDiff z;
   z.v = x.v * y.v;
   z.dv = x.dv * y.v + x.v * y.dv;
   z.d2v = x.d2v * y.v + x.dv * y.dv + x.dv * y.dv + x.v * y.d2v;
   return z;
}
```

[3] Operator Overloading

Das Software-Tool ADOL-C[4] (GRIEWANK ET AL. (1996)) zur automatischen Differentiation von Funktionen, die durch Quelltexte spezifiziert sind, verwendet das Prinzip des Operator-Overloadings. Für den abzuleitenden Bereich eines Programms wird eine Mitschrift (*Tape*) erstellt. Die in das Differenzieren involvierten Variablen müssen dabei von einem durch ADOL-C neu definiertem Datentyp sein. Die Mitschrift ist somit die Grundlage für die Ableitungsberechnung. Ein Interpreter kann in einem Rückwärtslauf das Tape lesen und die Ableitungen im Rückwärtsverfahren berechnen. Für Funktionen, die mit Verzweigungen (`if-then-else`) arbeiten, ändert sich der Berechnungsgraph in Abhängigkeit der Eingabedaten, und das Tape muß bei jedem Lauf neu angelegt werden. Dies macht sich natürlich negativ in der Laufzeit bemerkbar. Mit ADOL-C lassen sich sehr einfach Ableitungen beliebiger Ordnung berechnen. Hierfür lassen sich die Treiber-Routinen, wie z.B. `gradient(...)`, `hessian(..)`, `jac_vect(...)` und `hess_vec` verwenden, mit denen man die jeweiligen Ableitungen berechnen kann. Mit Hilfe der Routinen `forward(...)` und `reverse(..)` läßt sich sowohl der Vorwärts- als auch der Rückwärtsmodus bis zu einer beliebigen Ordnung für die Berechnung der Ableitungen verwenden.

[4] Automatic Differentiation with Operator Overloading in C

Kapitel 3

Praktische Realisierung

In diesem Kapitel werden die theoretischen Ergebnisse der Kapitel 1 und 2 benutzt und auf eine konkrete Problemklasse angewendet. Zuerst wird die Modellierung eines Reaktionssystems in einem Rührkessel beschrieben. Danach werden die für die Optimierung benötigten Richtungsableitungen aufgestellt.

3.1 Modellierung des chemischen Reaktionssystems

Im folgenden Abschnitt werden die reaktionskinetischen Betrachtungen aus Abschnitt 1.2 verwendet, um ein beliebiges instationäres homogenes chemisches Reaktionssystem in einem Rührkessel durch ein Differentiell-Algebraisches Gleichungssystem der Form

$$\begin{aligned} \dot{y}(t) &= f(t, y(t), z(t), p, q) \\ 0 &= g(t, y(t), z(t), p, q) \end{aligned} \tag{3.1}$$

aus Abschnitt 1.2 mathematisch zu beschreiben.

Jedes chemische Reaktionssystem läßt sich allgemein durch folgendes Schema beschreiben (vgl. z.B. BARROW (1973)):

$$\begin{array}{ccccccccccc} \alpha_{11}S_1 & + & \cdots & + & \alpha_{1n}S_n & \longrightarrow & \beta_{11}S_1 & + & \cdots & + & \beta_{1n}S_n \\ \alpha_{21}S_1 & + & \cdots & + & \alpha_{2n}S_n & \longrightarrow & \beta_{21}S_1 & + & \cdots & + & \beta_{2n}S_n \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \alpha_{m1}S_1 & + & \cdots & + & \alpha_{mn}S_n & \longrightarrow & \beta_{m1}S_1 & + & \cdots & + & \beta_{mn}S_n \end{array} \tag{3.2}$$

Hierbei ist m die Anzahl der Gleichungen, $S_1, \ldots, S_n$ sind die an den Reaktionen beteiligten Spezies sowie α_{ij} und β_{ij} die reellwertigen positiven *stöchiometrischen Koeffizienten* der Edukte und Produkte. Ist Spezies S_j nicht Edukt bzw. Produkt der i-ten Reaktionsgleichung, wird α_{ij} bzw. β_{ij} auf Null gesetzt. Handelt es sich bei einer Reaktion um eine chemische Gleichgewichtsreaktion, so wird diese im Schema (3.2) durch zwei Gleichungen modelliert. Die erste Gleichung beschreibt die Hinreaktion, die zweite die Rückreaktion.

Dabei werden aus Edukten der Hinreaktion Produkte der Rückreaktion und vice versa. Die stöchiometrischen Koeffizienten bleiben dabei gleich. Um daraus das DAE-System aufzustellen, benötigt man die *Stöchiometriematrix* $\mathcal{N}$ sowie die sich daraus ergebende *Bilanzmatrix* $\mathcal{E}$ (siehe Abschnitt 3.1.2). Die Stöchiometriematrix beschreibt den qualitativen und quantitativen Verlauf einer chemischen Reaktion, und ist durch das Reaktionsschema (3.2) gegeben. Aus der Matrix $\mathcal{N}$ lassen sich die Differentialgleichungen und aus der Matrix $\mathcal{E}$ die algebraischen Gleichungen des DAE-Systems (3.1) bestimmen. Die Anzahl der Reaktionsgleichungen entspricht dabei der Zeilenanzahl m von $\mathcal{N}$, die Anzahl der Spezies n addiert mit der Anzahl der Lösungsmittel n_l ist gleich der Spaltenanzahl n' von $\mathcal{N}$. Jede Zeile der Matrix $\mathcal{N}$ entspricht einer Reaktionsgleichung. Dabei gilt für die Einträge von $\mathcal{N}$:

$$\mathcal{N}(i,j) = \begin{cases} -\alpha_{ij}, & \text{falls Spezies } j \text{ Edukt in der } i\text{-ten Gleichung ist} \\ \beta_{ij}, & \text{falls Spezies } j \text{ Produkt in der } i\text{-ten Gleichung ist} \\ 0, & \text{sonst,} \end{cases}$$

$$\text{wobei } (i,j) \in \{1,\ldots,m\} \times \{1,\ldots,n,\ldots,n',\}\,.$$

Als Beispiel betrachte man eine chemische Reaktion zur Herstellung von Urethan aus Isocyanat (siehe auch Abschnitt 3.3). Dabei handelt es sich um ein Reaktionssystem mit fünf Spezies, einem Lösungsmittel, zwei einfachen Reaktionen und einer Gleichgewichtsreaktion, das in symbolischer Form folgendermaßen dargestellt werden kann:

$$\begin{array}{rcl} A + B & \longrightarrow & C \\ A + C & \rightleftharpoons & D \\ 3A & \longrightarrow & E \end{array} \tag{3.3}$$

Dann hat die dazugehörige Stöchiometriematrix $\mathcal{N}$ folgende Gestalt:

$$\mathcal{N} = \begin{pmatrix} -1 & -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & -1 & 1 & 0 & 0 \\ 1 & 0 & 1 & -1 & 0 & 0 \\ -3 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.4}$$

3.1.1 Aufstellen des Differentialgleichungssystems

In den vorherigen Abschnitten sowie im grundlegenden Abschnitt 1.2 über Reaktionskinetik wurden die chemischen und verfahrenstechnischen Grundlagen behandelt, die für das Aufstellen der Modellgleichungen notwendig sind. In den Modellgleichungen werden dabei die rechten Seiten des DAE-Systems (3.1) spezifiziert.

Betrachtet wird im folgenden ein chemischer Reaktionsprozeß, der sich nach dem Schema (3.2) beschreiben läßt. Die Reaktionen finden in einem Rührreaktor, wie er in Abschnitt 1.2.4 beschrieben ist, statt. Der Reaktor habe l Zuläufe, mit denen die Reaktionen in ihrem Reaktionsverhalten verändert werden können. Die Stöchiometriematrix $\mathcal{N} = (\nu_{ij}) \in \mathbb{R}^{m \times n}$ wird wie oben beschrieben aufgestellt. Im Modell auftretendende physikalische und chemische Größen werden in Tabelle 3.1 zusammengefaßt.

$r_i(t)$	Reaktionsgeschwindigkeit der Reaktion i	$[mol/min/m^3]$
$k_i(t)$	Reaktionsgeschwindigkeitskonstante der Reaktion i	
k_{c_i}	Gleichgewichtskonstante der i-ten Reaktion	
T_{ref_i}	Referenztemperatur der i-ten Reaktion	$[K]$
T_{g_i}	Temperatur auf die sich die Konstante k_{c_i} bezieht	$[K]$
ΔH_i	Reaktionsenthalpie der i-ten Reaktion	$[J/mol]$
α_{ij}	Reaktionsordnung für die j-te Spezies der i-ten Reaktion	
E_{a_i}	Aktivierungsenergie der i-ten Reaktion	$[kJ/mol]$
k_{ref_i}	Frequenzfaktor der i-ten Reaktion	
n_{a_j}	Anfangsstoffmenge der Spezies j im Reaktor	$[mol]$
M_j	Molmasse der j-ten Spezies	$[kg/mol]$
ϱ_j	Dichte der j-ten Spezies	$[kg/m^3]$
$V(t)$	Gesamtvolumen im Reaktor	$[m^3]$
$M_{ges}(t)$	Gesamtmasse im Reaktor	$[kg]$
R	Universelle Gaskonstante (= 8.314)	$[J/mol/K]$

Tabelle 3.1: Nomenklatur für die Modellgenerierung

Zunächst werden folgende Indexmengen für alle $1 \leq i \leq m$ definiert:

$$E_i := \left\{ j \in \{1, \ldots, n\} \mid \nu_{ij} < 0 \right\} \text{ (Menge der Edukte der Gleichung } i \text{)}$$
$$P_i := \left\{ j \in \{1, \ldots, n\} \mid \nu_{ij} > 0 \right\} \text{ (Menge der Produkte der Gleichung } i \text{)}$$

Im isothermen Fall wird für die Temperatur $T(t)$ im Reaktor der Wert $T_{in}(t)$ des Kühlmediums angenommen. Im nicht-isothermen Fall ergibt sich $T(t)$ bei den in dieser Arbeit betrachteten Prozessen aus der Differentialgleichung (3.28). Für die Reaktionsgeschwindigkeitskonstanten k_i $(1 \leq i \leq m)$ wird der Arrhenius-Ansatz gewählt, und es gilt nach Gleichung (1.11) abhängig vom Typ der Reaktion:

- Falls die Reaktion i eine Hinreaktion ist:

$$k_i(t) = k_{ref_i} \cdot \exp\left(-\frac{E_{a_i}}{R} \left(\frac{1}{T(t)} - \frac{1}{T_{ref_i}} \right) \right) \tag{3.5}$$

- Falls die Reaktion i eine Rückreaktion ist [1]:

$$k_i(t) = k_{i-1}(t) \Bigg/ \left(k_{c_{i-1}} \cdot \exp\left(-\frac{\Delta H_i}{R} \left(\frac{1}{T(t)} - \frac{1}{T_{g_i}} \right) \right) \right) \tag{3.6}$$

Das Volumen V und die Gesamtmasse M_{ges} zum Zeitpunkt t berechnen sich nach folgenden Formeln:

$$V(t) = \sum_{j=1}^{n} \frac{n_j(t) \cdot M_j}{\varrho_j}, \quad M_{ges}(t) = \sum_{j=1}^{n} n_j(t) \cdot M_j \tag{3.7}$$

Die Reaktionsgeschwindigkeit r_i ergibt sich durch:

$$r_i(t) = k_i(t) \cdot \prod_{j \in E_i} \left(\frac{n_j(t)}{V(t)} \right)^{\alpha_{ij}}, \quad (1 \leq i \leq m) \tag{3.8}$$

[1] Hierbei bezeichne k_{i-1} die zu k_i korrespondierende Hinreaktion.

Unter Verwendung der gesamten Zulaufrate $dn_{e_j}(t)$ der Spezies j, definiert in Tabelle 3.2, wir nun das folgende Differentialgleichungssystem aufgestellt.

$$\frac{d}{dt}n_j(t) = V(t) \cdot \sum_{i=1}^{m} (\nu_{ij} \cdot r_i(t)) + dn_{e_j}(t), \quad (1 \leq j \leq n) \quad (3.9)$$

3.1.2 Erweiterung mit algebraischen Gleichungen

Im vorherigen Abschnitt wurde für die Stoffmenge jeder Spezies n_j eine Differentialgleichung formuliert. Da in der numerischen Praxis bei der Integration des Differentialgleichungssystems (3.9) der zusätzlich geltende Massenerhaltungssatz (1.17) für jede Gleichung aufgrund numerischer Ungenauigkeiten (*Drift*) verletzt sein kann, werden zusätzliche algebraische Bedingungen, aus dem Massenerhaltungssatz resultierend, für jede Reaktionsgleichung formuliert.

Da die einzelnen Zeilen von $\mathcal{N}$ linear abhängig sein können, kann das resultierende DAE-System überbestimmt sein. Deswegen wird im weiteren eine vollrangige Teilmatrix von $\mathcal{N}$, mit Rang r bestimmt, aus der die Differentialgleichungen für r Spezies aufgestellt werden. Die entsprechenden Stoffmengen werden als *differentielle Zustandsvariablen (DV)* bezeichnet. Die restlichen $(n-r)$ Stoffmengen werden durch die obigen algebraische Gleichungen bestimmt und werden deshalb auch als *algebraische Zustandsvariablen (AV)* bezeichnet.

Desweiteren wird ein Verfahren hergeleitet, bei dem die rechten Seiten aller betrachteten Gleichungen zur Bestimmung der AV ausschließlich aus DV zuzüglich der Einheitsmatrix für die AV bestehen.

Dies hat den Vorteil, daß sich die AV direkt aus den DV ermitteln lassen, ohne bei der Integration berücksichtigt zu werden. Sei nun $r(t) = (r_1(t), r_2(t), \ldots, r_m(t))^T \in \mathbb{R}^m$ der Vektor der Reaktionsgeschwindigkeiten, $\mathcal{N}$ die Stöchiometriematrix sowie $n(t) = (n_1(t), n_2(t), \ldots, n_n(t))^T$ $(t \in \mathbb{R})$ der zeitabhängige Vektor der Stoffmengen der beteiligten Spezies. Es gilt aufgrund reaktionskinetischer Betrachtungen folgendes Differentialgleichungssystem:

$$\dot{n}(t) = V(t) \cdot \mathcal{N}^T \cdot r(t) \quad (3.10)$$

Sei t_0 Anfangswert des obigen Systems, so gilt nach dem Hauptsatz der Differential- und Integralrechnung:

$$\begin{aligned} n(t) &= n(t_0) + \int_{t_0}^{t} \dot{n}(\tau)\, d\tau = n(t_0) + \int_{t_0}^{t} \mathcal{N}^T \cdot r(\tau) V(\tau)\, d\tau \\ &= n(t_0) + \mathcal{N}^T \underbrace{\int_{t_0}^{t} r(\tau) V(\tau)\, d\tau}_{:=\chi(t)} = n(t_0) + \mathcal{N}^T \chi(t) \end{aligned}$$

Mit $\chi(t) = (\chi_1(t), \chi_2(t), \ldots, \chi_m(t))$. Die Komponentenfunktion $\chi_i(t)$ wird auch als der *Fortschrittsgrad* der Reaktion i bezeichnet. Für die Molzahländerungen Δn aller Spezies gilt somit:

$$\Delta n := n(t) - n(t_0) = n(t_0) + \mathcal{N}^T \chi - n(t_0) = \mathcal{N}^T \chi \quad (3.11)$$

Der Massenerhaltungssatz (1.17) garantiert dazu die Existenz einer vollrangigen Matrix $\mathcal{E} \in \mathbb{R}^{(n-r)\times n}$ mit $r = \text{Rang}(\mathcal{N})$, für die gelten muß:

$$\mathcal{E} \cdot \Delta n = 0 \tag{3.12}$$

Bei der unten angegebenen praktischen Berechnung der Bilanzmatrix wird jetzt eine Matrix $\mathcal{E}$ konstruiert mit

$$\boxed{\mathcal{E} \cdot \mathcal{N}^T = 0, \quad \mathcal{E} \text{ vollrangig.} \qquad (3.13)}$$

Die Matrix $\mathcal{E}$ ist dabei im allgemeinen nicht eindeutig bestimmt. Da somit

$$\mathcal{E} \cdot \Delta n = \mathcal{E} \cdot \mathcal{N}^T \cdot \chi = 0 \cdot \chi = 0$$

gilt, erfüllt die im allgemeinen nichteindeutige Matrix $\mathcal{E}$ die in Gleichung (3.12) geforderte Bedingung.

Im folgenden wird ein Verfahren zur Berechnung einer Matrix $\mathcal{E}$ angegeben, das die Gleichung (3.13) erfüllt und im Rahmen eines Software-Praktikums von FANTANA & GUTFLEISCH (1998) entwickelt und implementiert wurde. Dafür werden die folgenden Sätze aus der Linearen Algebra verwendet (vergleiche FISCHER (1989)).

Satz 3.1.1 (Gaußsches Eliminationsverfahren) *Jede $(m \times n)$-Matrix $\mathcal{A}$ läßt sich mit endlich vielen elementaren Zeilenumformungen in eine Matrix $\mathcal{A}^*$ in Zeilenstufenform überführen. Die beiden durch die Matrizen $\mathcal{A}$ und $\mathcal{A}^*$ beschriebenen Gleichungssysteme haben dabei die gleiche Lösungsmenge. Der Rang von $\mathcal{A}$ ist dabei gleich der Anzahl der Pivotelemente von $\mathcal{A}^*$.*

Satz 3.1.2 *Die sich aus den Pivotelementen von $\mathcal{A}^*$ ergebenen Spalten sind linear unabhängig. Die Spalten mit denselben Indizes aus $\mathcal{A}$ sind somit auch linear unabhängig und bilden dadurch eine vollrangige Submatrix von $\mathcal{A}$.*

Sei nun $\mathcal{N}^*$ die nach Satz 3.1.1 zu $\mathcal{N}$ korrespondierende Matrix in Zeilenstufenform mit

$$r := \text{Rang}(\mathcal{N}^*) = \text{Rang}(\mathcal{N}) \in \mathbb{N}.$$

$\mathcal{N}^*$ läßt sich aus $\mathcal{N}$ nach Satz 3.1.1 durch $t \in \mathbb{N}$ Zeilenumformungen bestimmen. Unter Verwendung von t Elementarmatrizen $\mathcal{G}_i$ gilt:

$$\mathcal{N}^* = \mathcal{N} \cdot \underbrace{\prod_{i=1}^{t} \mathcal{G}_i}_{:=\mathcal{G}}, \quad \mathcal{G} \text{ invertierbar} \tag{3.14}$$

Sei $\tilde{\mathcal{N}}^*$ diejenige Matrix, bei der auf der Hauptdiagonalen die Pivotelemente von $\mathcal{N}^*$ stehen und die durch sukzessive Multiplikation mit $s \in \mathbb{N}$ Permutationsmatrizen $\mathcal{P}_i$ (vergleiche FISCHER (1989)) aus $\mathcal{N}^*$ entsteht. Es gilt also:

$$\tilde{\mathcal{N}}^* = \mathcal{N}^* \cdot \underbrace{\prod_{i=1}^{s} \mathcal{P}_i}_{:=\mathcal{P}} \quad \text{mit} \quad \mathcal{P}^T = \mathcal{P}^{-1} \tag{3.15}$$

Damit läßt sich Gleichung (3.13) wie folgt umformen:

$$\begin{aligned}
& \mathcal{E} \cdot \mathcal{N}^T = 0 \\
\Longleftrightarrow \quad & \left(\mathcal{E} \cdot \mathcal{N}^T\right)^T = \mathcal{N} \cdot \mathcal{E}^T = 0 \\
\Longleftrightarrow \quad & \mathcal{G}^{-1} \cdot \mathcal{N}^* \cdot \mathcal{E}^T = 0 \\
\Longleftrightarrow \quad & \mathcal{N}^* \cdot \mathcal{E}^T = 0 \\
\Longleftrightarrow \quad & \mathcal{N}^* \cdot \mathcal{I} \cdot \mathcal{E}^T = \mathcal{N}^* \cdot \left(\mathcal{P} \cdot \mathcal{P}^{-1}\right) \cdot \mathcal{E}^T = \mathcal{N}^* \cdot \left(\mathcal{P} \cdot \mathcal{P}^T\right) \cdot \mathcal{E}^T = 0 \\
\Longleftrightarrow \quad & (\mathcal{N}^* \cdot \mathcal{P}) \cdot \left(\mathcal{P}^T \cdot \mathcal{E}^T\right) = 0
\end{aligned} \tag{3.16}$$

Da die Matrix $\tilde{\mathcal{N}}^*$ in Zeilenstufenform ist, hat sie folgende Gestalt:

$$\tilde{\mathcal{N}}^* = \begin{pmatrix}
* & * & \cdots & * & \cdots & * \\
0 & * & \cdots & * & \cdots & * \\
\vdots & \ddots & \ddots & \vdots & & \vdots \\
0 & \cdots & 0 & * & \cdots & * \\
0 & \cdots & 0 & 0 & \cdots & 0 \\
\vdots & & \vdots & \vdots & & \vdots \\
0 & \cdots & 0 & 0 & \cdots & 0
\end{pmatrix} \tag{3.17}$$

Elementare Zeilenumformungen ermöglichen folgende Darstellung mit Hilfe einer regulären Matrix $\mathcal{D} \in \mathbb{R}^{r \times w}$:

$$\mathcal{D} \cdot \tilde{\mathcal{N}}^* = \left(\begin{array}{cccc|ccc}
1 & 0 & \cdots & 0 & l_{11} & \cdots & l_{1w} \\
0 & 1 & \cdots & 0 & l_{21} & \cdots & l_{2w} \\
\vdots & \ddots & \ddots & \vdots & \vdots & & \vdots \\
0 & \cdots & 0 & 1 & l_{r1} & \cdots & l_{rw} \\
\hline
0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & & \vdots & \vdots & \vdots & \cdots & \vdots \\
0 & \cdots & 0 & 0 & 0 & \cdots & 0
\end{array}\right) =: \begin{pmatrix} \mathcal{I}_r & \mathcal{S}_w \\ 0 & 0 \end{pmatrix} \tag{3.18}$$

mit $W = n - r$.

Für die durch $\mathcal{C} := \begin{pmatrix} -\mathcal{S}_w \\ \mathcal{I}_w \end{pmatrix}$ definierte Matrix mit Rang $(\mathcal{C}) = w$ gilt:

$$\begin{pmatrix} \mathcal{I}_r & \mathcal{S}_w \\ 0 & 0 \end{pmatrix} \cdot \mathcal{C} = \begin{pmatrix} \mathcal{I}_r & \mathcal{S}_w \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} -\mathcal{S}_w \\ \mathcal{I}_w \end{pmatrix} = \begin{pmatrix} -\mathcal{I}_r \cdot \mathcal{S}_w + \mathcal{S}_w \cdot \mathcal{I}_w \\ 0 \end{pmatrix} = 0 \tag{3.19}$$

sowie

$$\text{Rang}\,(\mathcal{N}) + \text{Rang}\,(\mathcal{C}) = r + w = n. \tag{3.20}$$

Setzt man nun $\mathcal{E} := \mathcal{C}^T \cdot \mathcal{P}^T$, so gilt:

$$\begin{aligned}
& \mathcal{E}^T = \left(\mathcal{C}^T \cdot \mathcal{P}^T\right)^T = \mathcal{P} \cdot \mathcal{C} \\
\Longleftrightarrow \quad & \mathcal{C} = \mathcal{P}^{-1} \cdot \mathcal{E}^T = \mathcal{P}^T \cdot \mathcal{E}^{\mathcal{T}}
\end{aligned} \tag{3.21}$$

Unter Verwendung der Gleichungen (3.16), (3.19) und (3.21) sowie der Regularität von D gilt:

$$\begin{aligned}
& \begin{pmatrix} \mathcal{I}_r & \mathcal{S} \\ 0 & 0 \end{pmatrix} \cdot \mathcal{C} = 0 \\
\Longleftrightarrow \quad & \left(\mathcal{D} \cdot \tilde{\mathcal{N}}^*\right) \cdot \mathcal{C} = 0 \\
\Longleftrightarrow \quad & \mathcal{D} \cdot (\mathcal{N}^* \cdot \mathcal{P}) \cdot \left(\mathcal{P}^T \cdot \mathcal{E}^T\right) = 0 \\
\Longleftrightarrow \quad & (\mathcal{N}^* \cdot \mathcal{P}) \cdot \left(\mathcal{P}^T \cdot \mathcal{E}^T\right) \\
\Longleftrightarrow \quad & \mathcal{E} \cdot \mathcal{N}^T = 0
\end{aligned}$$

Somit erfüllt das gewählte $\mathcal{E}$ Gleichung (3.13) und hat mit $\text{Rang}(\mathcal{E}) = \text{Rang}(\mathcal{C}) = w = n-r$ auch den gewünschten vollen Rang.

Bestimmung der algebraischen Zustandsvariablen

Um nun die algebraischen Variablen durch die differentiellen Zustandsvariablen zu bestimmen, sei p_i der Pivotspaltenindex der Zeile i von $\mathcal{E}$ und $P := \{p_1, \ldots, p_r\}$ $(i = 1, \ldots, w)$. Zusätzlich definiert man die Indexmenge

$$I_{DV} := \{1, \ldots, n\} \setminus \{p_1, \ldots, p_w\} \quad \text{mit } |I_{DV}| = \text{Rang}(\mathcal{N}) = r$$

der differentiellen Zustandsvariablen. Die vollrangige Bilanzmatrix $\mathcal{E}$ wird gerade so erzeugt, daß in den Spalten der algebraischen Variablen die Einheitsmatrix steht. Deshalb lassen sich die algebraischen Gleichungen auflösen, und die algebraischen Zustandsvariablen können direkt durch die r differentiellen Zustandsvariablen ausgedrückt werden. Die Molzahländerung Δn_j der differentiellen Variablen definiert man unter Berücksichtigung der modellierten Zuläufe und mit Hilfe der Größen aus Tabelle 3.2 durch:

$$\Delta n_j := n_j - n_{a_j} - \sum_{k=1}^{l} n_{e_{k,j}}, \quad j \in I_{DV} \tag{3.22}$$

Für $i = 1, \ldots, w$ gilt daher für die algebraischen Variablen:

$$n_j = n_{a_j} + \sum_{k=1}^{l} n_{e_{k,j}} - \sum_{\substack{s=1 \\ s \notin P}}^{n} \varepsilon_{js} \Delta n_s \quad \text{mit } j := p_i \tag{3.23}$$

3.1.3 Verfahrenstechnische Aspekte der Modellierung

In der DAE (3.1) sind zusätzliche, vom Experimentator wählbare Größen, *Steuergrößen* genannt, enthalten. Durch diese Steuergrößen q läßt sich der Reaktionsverlauf des Systems beeinflussen. Der Vektor q kann aus zeitlich festen Versuchsplanungsgrößen wie z.B.

den Anfangsstoffmengen na_i von Spezies und Lösungsmitteln oder der Anfangstemperatur $T_0 = T(t_0)$ bestehen. Zusätzlich können durch zeitabhängige Funktionen $u(t)$, auch *Steuerfunktionen* genannt, die Prozeßeigenschaften beeinflußt werden.

Bei den in dieser Arbeit betrachteten Prozessen werden als weitere verfahrenstechnische Komponenten z.B. Reaktoreinträge (Zuflüsse) $feed_i(t)$ sowie die Kühl- und Heiztemperatur $T_{in}(t)$ als Steuerfunktion modelliert. Hier tritt das Problem auf, den unendlichdimensionalen Funktionenraum zu diskretisieren.

Im folgenden wird der *direkte Ansatz* für optimale Steuerungsprobleme gewählt: Jede Steuerfunktion $u : [t_0, t_{end}] \longrightarrow \mathbb{R}$ wird durch einen Satz weiterer, endlich vieler, Steuergrößen q spezifiziert. Die Funktion u wird mit Hilfe endlich vieler Größen parametrisiert. Hierbei wird ein Gitter aus m Teilintervallen festgelegt:

$$t_0 = \tau_0 < \tau_1 < \cdots < \tau_m = t_{end}$$

Die Steuerfunktion $u(t)$ wird nun auf

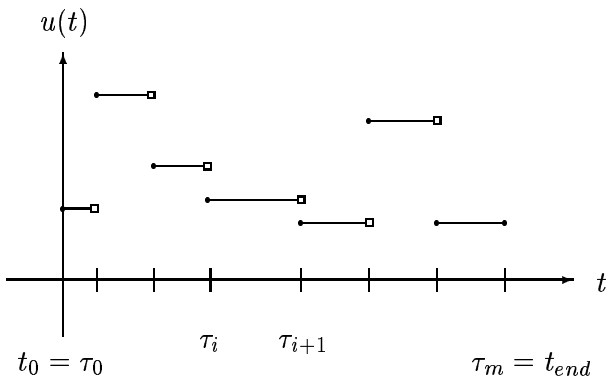
$$I_i := \begin{cases} [\tau_i, \tau_{i+1}[, & \text{falls } i = 0, \ldots, m-2 \\ [\tau_i, \tau_{i+1}], & \text{falls } i = m-1 \end{cases}$$

durch m lokale Ansatzfunktionen

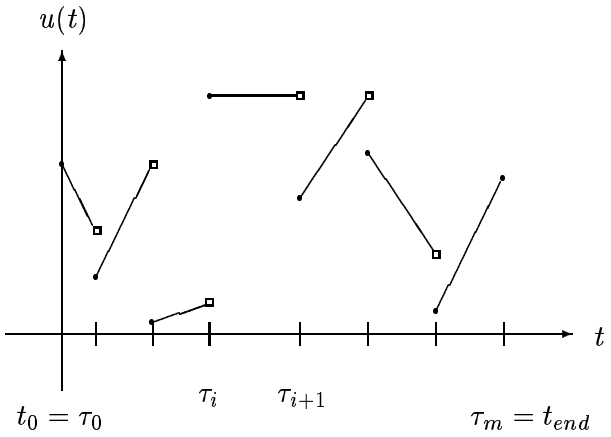
$$u(t) := u_i\,(t, q)\,, \quad t \in I_i, \quad i = 0, \ldots, m-1$$

ersetzt. Hier bestimmt die Anzahl der Freiheitsgrade die Anzahl der Steuergrößen q der lokalen Funktionen u_i. Typische Beispiele für die u_i sind stückweise Polynome, die im weiteren näher bestimmt werden.

1. u_i ist stückweise konstant: $u(t) = u_i(t, q) = q_{i_0+i}$



2. u_i ist stückweise linear: $u(t) = u_i(t, q) = q_{(i_0+2i)} + (t - \tau_i) \cdot q_{(i_0+2i+1)}$



Wird von den Steuerfunktionen $u(t)$ im zweiten Fall Stetigkeit verlangt, müssen zusätzliche Nebenbedingungen für alle $1 \leq i \leq m$ formuliert werden:

$$\lim_{t \to \tau_{i+1}} u_i(t,q) = q_{(i_0+2i)} + (\tau_{i+1} - \tau_i) \cdot q_{(i_0+2i+1)} \stackrel{!}{=} q_{(i_0+2(i+1))} = u_{i+1}(\tau_{i+1}, q)$$

Die Behandlung von vorgegebenen Schranken an die einzelnen Steuerfunktionen u_i wird durch Nebenbedingungen an die Parametrisierungsvariablen q_{i_0+2i} und q_{i_0+2i+1} der u_i modelliert.

Wir betrachten nun folgende verfahrenstechnische Komponenten, die hier als stückweise lineare stetige Steuerfunktionen modelliert:

Temperaturprofil des Kühlmediums

Die Innentemperatur T_{in} des Rührreaktors läßt sich vom Experimentator beliebig in gegebenen Grenzen einstellen. Zum Beispiel kann die Temperatur T_{in} nach dem sogenannten Prinzip des *Kühlungsdreiecks* (Abbildung 3.1) eingestellt werden. Zu Beginn ($t = t_0$) wird die Temperatur auf Raumtemperatur ($T_{in} = 293K$) gesetzt. Im Laufe der Reaktion wird zum Zeitpunkt $t = 8h$ die Reaktionstemperatur $T_{in} = 472K$ erreicht. Danach wird in einem Abkühlungsprozeß T_{in} im Laufe der restlichen Reaktionszeit auf Raumtemperatur gekühlt. Eine lineare Abnahme der Temperatur wird nicht zugelassen, da nachts keine Reaktion stattfinden soll. Dies erklärt die auftretenden Plateaus.

Zuflüsse (Zuläufe) am Reaktor

An der Reaktorwand können Zulaufhähne angebracht sein, mit denen die an der Reaktion beteiligten Spezies zeitlich abhängig dem Reaktor zugeführt werden.

Folgende Größen werden zur Beschreibung eines Reaktors mit mehreren Zuläufen[2] in Tabelle 3.2 eingeführt.

[2]Um mehrere Zuläufe unterscheiden zu können spezifiziert der Index k ($1 \leq k \leq l$) bei diesen Größen den jeweiligen Zulauf.

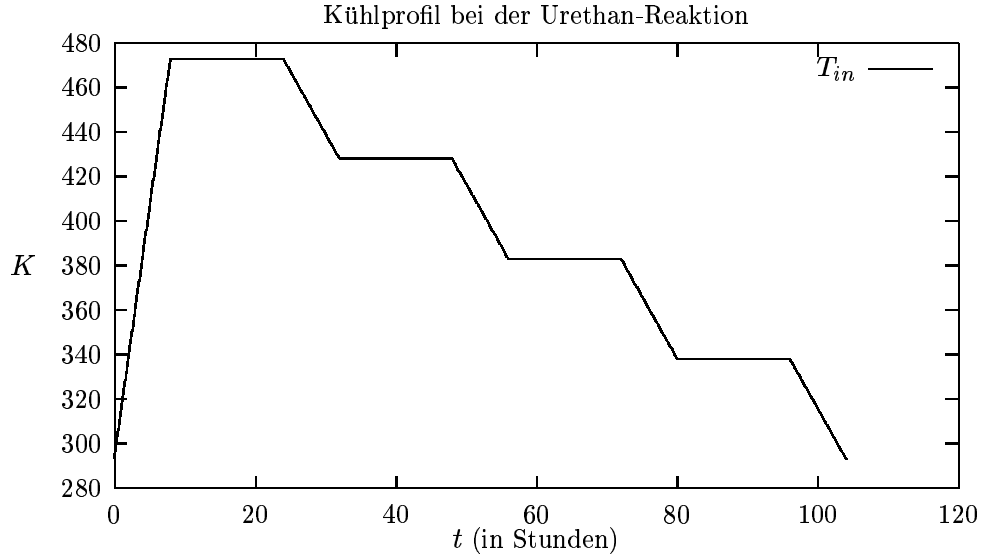


Abbildung 3.1: Beispiel für die Modellierung eines Temperaturprofils

$M_{e_k}(t)$	zugeflossene Masse zum Zeitpunkt t	$[kg]$
$dM_{e_k}(t)$	Massenzulaufrate des Zulaufes zum Zeitpunkt t	$[kg/min]$
$Mp_{e_k,j}$	Massenprozent der Spezies j an der Gesamtmasse des Zulaufes	
$feed_k(t)$	Anteil des bereits zugeflossenen Zulaufinhaltes am gesamten Zulauf zum Zeitpunkt t	
$dfeed_k(t)$	zeitliche Ableitung von $feed(t)$	
$na_{e_k,j}$	Anfangsstoffmenge der Spezies j im Zulauf	$[mol]$
$n_{e_k,j}(t)$	zugeflossene Stoffmenge der Spezies j zum Zeitpunkt t	$[mol]$
$dn_{e_j}(t)$	gesamte Zulaufrate der Spezies j	

Tabelle 3.2: Nomenklatur für die Modellierung von Reaktorzuläufen

Die gesamte Zulaufrate einer Spezies ergibt sich durch:

$$dn_{e_j}(t) = \sum_{k=1}^{l} na_{e_k,j} \cdot dfeed_k(t), \tag{3.24}$$

Jeder Zufluß wird nun durch eine Steuerfunktion $u(t)$ beschrieben. Dabei lassen sich zwei verschiedene Modi unterscheiden.

1. Der Zulauf k wird durch Angabe von $feed_k(t)$ modelliert. Dabei muß gelten:

$$feed_k(t_0) = 0, \quad feed_k(t_{end}) = 1.$$

Dies bedeutet, daß zu Beginn nichts, am Ende der gesamte Zulaufinhalt zugeflossen ist. Für jede Spezies j, die im Zulauf k vorhanden ist gilt:

$$n_{e_k,j}(t) = na_{e_k,j} \cdot feed_k(t) \tag{3.25}$$

2. Durch Angabe der Massenzulaufrate $dM_{e_k}(t)$ wird der Zulauf modelliert. Die bereits zugeflossene Masse $M_{e_k}(t)$ des Zulaufes k ist somit durch

$$M_{e_k}(t) = \int_{t_0}^{t} dM_{e_k}(\tau)\ d\tau$$

gegeben. Für jede Spezies j, die im Zulauf k vorhanden ist, gilt:

$$n_{e_{k,j}}(t) = \frac{M_{e_k}(t) \cdot Mp_{e_{k,j}}}{M_j}, \tag{3.26}$$

wobei $M_{e_k}(t)$ der zusätzlichen Differentialgleichung

$$\dot{M}_{e_k}(t) = dM_{e_k}(t),\ M_{e_k}(t_0) = 0 \tag{3.27}$$

genügen muß.

Typische Zulaufprofile werden in Abbildung 3.2 dargestellt.

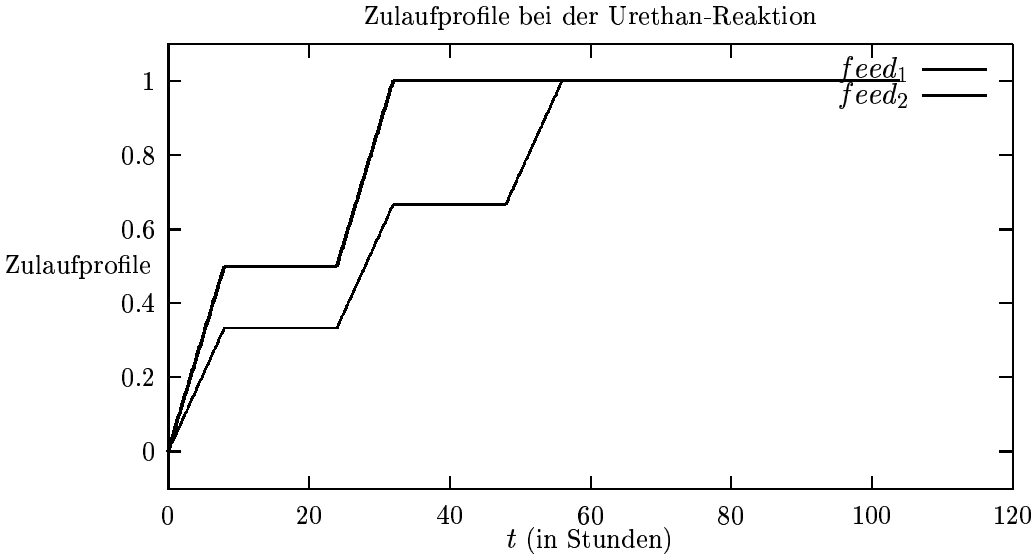


Abbildung 3.2: Beispiel für die Modellierung eines Zulaufprofils

3.1.4 Behandlung der Wärmebilanz

Bei komplexeren chemischen Prozessen[3] ist eine isotherme Reaktionsführung nicht oder nur sehr schwer möglich. Um dieser Tatsache bei der Experimentsimulation und Versuchsplanung gerecht zu werden, wird hier eine zusätzliche Differentialgleichung für die Wärmebilanz im instationären Rührkessel mitberücksichtigt. Die benötigten Größen werden in Tabelle 3.3 aufgelistet.

[3]Z.B. bei der in Abschnitt 5.3 beschriebenen Phosphin-Reaktion

T	absolute Temperatur im Reaktionsvolumen V	$[K]$
T_{in}	absolute Temperatur des Kühlmediums	$[K]$
T_{e_k}	absolute Temperatur des Eintrags k	$[K]$
c_p	spezifische Wärme der Reaktionslösung im Reaktionsvolumen	$[J/kg/K]$
d	Durchmesser des Reaktors	[m]
h	Reaktorfüllhöhe	[m]
A_w	Wärmeaustauschfläche zwischen Reaktionsmasse und Reaktor	$[m^2]$
T_w	mittlere Kühlwassertemperatur des Kühlmediums	[K]
k_d	Wärmedurchgangskoeffizient	$[J/m^2/min/K]$
v_k	Volumenstrom des Kühlmediums	$[m^3/min]$
ϱ_k	Dichte des Kühlmediums	$[kg/m^3]$
$c_{p,k}$	spezifische Wärme des Kühlmediums	$[J/kg/K]$

Tabelle 3.3: Nomenklatur für die Wärmebilanz in einem Rührkessel

Mit der Berechnung folgender Zwischenwerte

$$
\begin{aligned}
h &= \frac{4 \cdot V}{d^2 \cdot \pi \cdot 1000} \\
A_w &= \pi \cdot d \cdot h + \frac{\pi \cdot d^2}{4} = \frac{4 \cdot V}{d \cdot 1000} + \frac{\pi \cdot d^2}{4} \\
T_w(T(t)) &= \frac{k_d \cdot A_w \cdot T(t) + (2 v_k \cdot \varrho_k \cdot c_{p,k}) \cdot T_{in}(t)}{k_d \cdot A_w + 2 v_k \cdot \varrho_k \cdot c_{p,k}} \\
dM_{e_k} &= df eed_k(t) \cdot \sum_{j=1}^{n} \left(na_{e_{k,j}} \cdot M_j \right), \quad k = 1, \ldots, l \\
w_1 &:= V(t) \cdot \sum_{i=1}^{m} r_i(t) \cdot \Delta H_i && \text{(Reaktion)} \\
w_2 &:= \sum_{k=1}^{l} (dM_{e_k} \cdot c_p \cdot (T_{e_k} - T(t))) && \text{(Zuläufe)} \\
w_3 &:= k_d \cdot A_w \cdot (T_w - T(t)) && \text{(Kühlung)}
\end{aligned}
$$

kann die Differentialgleichung für die Wärmebilanz nach Kud (1997) aufgestellt werden.[4]

$$\boxed{\frac{d}{dt} T(t) = \frac{w_1 + w_2 + w_3}{c_p \cdot M_{ges}(n(t))} \quad (3.28)}$$

3.1.5 Parameter, Steuergrößen und Zustandsvariablen

Bei einigen in den Gleichungen (3.5) bis (3.9) und (3.22) bis (3.28) auftretenden Größen kann es sich neben Konstanten und den oben beschriebenen Steuergrößen q und den Zustandsvariablen x auch um sogenannte Parameter p handeln. Es handelt sich hierbei fast immer um Natur- oder Gerätekonstanten im allgemeinen sowie um kinetische Reaktionsgrößen im speziellen. Bei den hier betrachteten Prozessen sind typische Parameter

[4] Dabei wird die Annahme, daß c_p unabhängig von der Zusammensetzung der Reaktionslösung ist, gemacht.

Frequenzfaktoren k_{ref_i}, Reaktionsordnungen α_{ij}, Aktivierungsenergien E_{a_i} sowie Reaktionsenthalpien ΔH_i. Diese Parameter werden in der in Abschnitt 1.30 beschriebenen Parameterschätzung mit dem Programm *PARFIT* näherungsweise bestimmt.

3.2 Berechnung der Ableitungen

Um das DAE-System (1.2) integrieren und das Optimierungsproblem aus Abschnitt 1.3.3 lösen zu können, werden die ersten und zweiten Ableitungen der rechten Seiten der Modellgleichungen des DAE-Systems nach den Zustandsvariablen x, den Steuergrößen q sowie den Parametern p benötigt. Um die Prinzipien des automatischen Differenzierens anzuwenden, erfolgt nun die Berechnung der Ableitungen aller betrachteten Gleichungen und Zuweisungen nach den Variablen x, p und q. Da man a priori nicht immer weiß, ob eine Variable Parameter, Steuergröße oder Konstante ist, müssen bei der automatischen Ableitungserzeugung alle sinnvollen Fälle betrachtet werden. Um die Ableitungen der einzelnen Gleichungen nach den Vektoren x, p, q mit Hilfe eines Verfahrens aus Abschnitt 2.2 berechnen zu können, werden in den beiden nächsten Abschnitten die Terme für die ersten und zweiten Ableitungen der Modellgleichungen analytisch notiert.[5]

3.2.1 Erste Richtungsableitungen

Für jede Größe v, die von einem der drei Vektoren x, p, q abhängt, bezeichne $\dot{v}$ in den nachfolgenden Gleichungen den Gradientenvektor (siehe Gleichung (2.15)) der Größe v nach dem abzuleitenden Vektor x, p oder q. Hängt eine nicht konstante Größe auf der rechten Seite der Gleichung nicht vom Vektor der abhängigen x, p oder q ab, so ist der Gradientenvektor $\dot{v}$ dieser Größe gleich Null. Alle Gradientenvektoren in den folgenden Zuweisungen, denen kein Wert explizit zugeordnet wird, hängen direkt von einem der drei Vektoren x, p oder q ab, und die Werte dieser Gradientenvektoren ergeben sich direkt aus den einzelnen Richtungen nach denen abgeleitet wird (z.B. $\dot{na}_{1,2} = q(1), \ddot{na}_{1,2} = q(1,1)$). Die Berechnung der ersten Ableitungen der rechten Seiten des Differentialgleichungssystems (3.9) nach den Versuchsplanungsgrößen x, p und q ergibt sich somit aus folgenden Gleichungen in dieser Reihenfolge:

Die Ableitungen der zugeflossenen Stoffmenge $n_{e_{k,j}}$ von Spezies j im k-ten Zulauf erhält man durch:

$$\dot{n}_{e_{k,j}} = \dot{na}_{e_{k,j}} \cdot feed_k + na_{e_{k,j}} \cdot \dot{feed}_k, \quad 1 \leq k \leq l, 1 \leq j \leq n \tag{3.29}$$

Für die Molzahländerung der differentiellen Zustandsvariablen gilt unter Verwendung der Indexmenge I_{DV} aus Abschnitt 3.1.2 aller differentiellen Zustandsvariablen:

$$\Delta\dot{n}_j = \dot{n}_j - \dot{n}_{a_j} - \sum_{k=1}^{l} \dot{n}_{e_{k,j}}, \quad j \in I_{DV} \tag{3.30}$$

[5] Das Mitführen des Arguments t wird bei allen von t abhängigen Größen übersichtlichkeitshalber weggelassen.

Die Ableitungen der Stoffmengen, die keine differentiellen Variablen sind (siehe Abschnitt 3.1.2), ergeben sich durch:

$$\dot{n}_j = \dot{n}_{a_j} + \sum_{k=1}^{l} \dot{n}_{e_{k,j}} - \sum_{\substack{s=1 \\ s \notin P}}^{n} \varepsilon_{js} \dot{\Delta n}_s \quad \text{mit } j := p_i, \quad i \in P \tag{3.31}$$

V und M_{ges} nach den Versuchsplanungsgrößen differenziert ergibt:

$$\dot{V} = \sum_{j=1}^{n} \dot{n}_j \cdot \frac{M_j}{\varrho_j}, \quad \dot{M}_{ges} = \sum_{j=1}^{n} \dot{n}_j \cdot M_j \tag{3.32}$$

Für die beiden Zuweisungen (3.5) und (3.6) der Reaktionsgeschwindigkeitskonstanten k_i $(1 \leq i \leq m)$ gelten mit den Hilfvariablen s_1, s_2, definiert durch

$$s_1 := -\frac{E_{a_i}}{R}\left(\frac{1}{T} - \frac{1}{T_{ref_i}}\right), \quad s_2 := -\frac{\Delta H_i}{R}\left(\frac{1}{T} - \frac{1}{T_{g_i}}\right) \tag{3.33}$$

folgende Terme:

$$\dot{k}_i = \dot{k}_{ref_i} \cdot \frac{k_i}{k_{ref_i}} + \dot{E}_{a_i} \cdot \frac{s_1 \cdot k_i}{E_a} + \dot{T} \cdot \frac{E_{a_i} \cdot k_i}{R \cdot T^2}, \tag{3.34}$$

sowie

$$\dot{k}_i = \dot{k}_{i-1} \cdot \frac{k_i}{k_{i-1}} - \dot{k}_{c_{i-1}} \cdot \frac{k_i}{k_{c_{i-1}}} - \dot{\Delta H}_{i-1} \cdot \frac{k_i \cdot s_2}{\Delta H_{i-1}} - \dot{T} \cdot \frac{k_i \cdot \Delta H_{i-1}}{R \cdot T^2}. \tag{3.35}$$

Die Ableitungen der Reaktionsgeschwindigkeiten r_i $(1 \leq i \leq m)$ aus Gleichung (3.8) nach den Versuchsplanungsgrößen ergeben sich mit Hilfe der Indexmengen aus Abschnitt 3.1.1 und

$$S := \sum_{j \in E_i} \alpha_{ij} \tag{3.36}$$

als

$$\begin{aligned} \dot{r}_i \;=\; & \dot{k}_i \cdot \frac{r_i}{k_i} - \dot{V} \cdot \frac{S \cdot r_i}{V} + k \cdot V^{-S} \cdot \underbrace{\sum_{j \in E_i} \left(\dot{n}_j \cdot \alpha_{ij} \cdot n_j^{(\alpha_{ij}-1)} \prod_{\substack{k \in E_i \\ k \neq j}} n_k^{\alpha_{ik}} \right)}_{:= s_3} \\ + \; & \underbrace{k \cdot V^{-S} \underbrace{\sum_{j \in E_i} \left(\dot{\alpha}_{ij} \cdot \log(n_j) \cdot n_j^{\alpha_{ij}} \cdot \prod_{\substack{k \in E_i \\ k \neq j}} n_k^{\alpha_{ij}} \right)}_{:= s_4} + r_i \cdot \log(V) \cdot \underbrace{\sum_{j \in E_i} \dot{\alpha}_{ij}}_{:= s_6}}_{:= s_5}. \end{aligned} \tag{3.37}$$

Für den Vektor f der rechten Seiten des Differentialgleichungssystems (3.9), definiert durch

$$f(t) := \left(\frac{d}{dt} n_{i_1}(t), \ldots, \frac{d}{dt} n_{i_r}(t) \right)^T, \quad i_1, \ldots, i_r \in I_{DV}, \tag{3.38}$$

gilt:

$$\dot{f}_j = \dot{V} \cdot \sum_{i=1}^{m} \nu_{ij} \cdot r_i + V \cdot \sum_{i=1}^{m} \nu_{ij} \cdot \dot{r}_i + \dot{dn}_{e_j} \tag{3.39}$$

Für die Wärmebilanzgleichung bei nicht-isothermen Reaktionen gilt mit $f_1 := \frac{dT}{dt}$:

$$\begin{aligned} \dot{f}_1 &= \frac{1}{c_p \cdot M_{ges}} \cdot \left[\frac{\dot{V} \cdot w_1}{V} + V \cdot \sum_{i=1}^{m} \left(\dot{r}_i \cdot \Delta H_i + \dot{\Delta H}_i \cdot r_i \right) \right. \\ &+ c_p \cdot \sum_{k=1}^{l} \left(\dot{dM}_{e_k} \cdot (T_{e_k} - T) + \dot{T}_{e_k} \cdot dM_{e_k} - \dot{T} \cdot dM_{e_k} \right) \qquad (3.40) \\ &+ \left. \frac{\dot{A}_w \cdot w_3}{A_w} + s_1 \cdot \left(\dot{T}_w - \dot{T} \right) \right] - \frac{\dot{M}_{ges} \cdot f_1}{M_{ges}} \qquad (3.41) \end{aligned}$$

mit folgenden Zwischenwerten:

$$\begin{aligned} \dot{dM}_{e_k} &= \dot{dfeed} \cdot \sum_{j=1}^{n} \left(na_{e_{k,j}} \cdot M_j \right) + dfeed \cdot \sum_{j=1}^{n} \dot{na}_{e_{k,j}} \cdot M_j, \quad k = 1, \ldots, l \\ \dot{T}_w &= \frac{\dot{T} \cdot s_1 + \dot{T}_{in} \cdot T}{s_1 + s_2} + \frac{\dot{A}_w \cdot k_d \cdot T \cdot (s_1 + s_2) - (s_1 \cdot T + s_2 \cdot T_{in}) \cdot \dot{A}_w \cdot k_d}{(s_1 + s_2)^2} \\ \dot{A}_w &= \frac{4 \cdot \dot{V}}{d \cdot 1000} \\ s_1 &:= k_d \cdot A_w \\ s_2 &:= 2v_k \cdot \varrho_k \cdot c_{p,k} \end{aligned}$$

3.2.2 Zweite Richtungsableitungen

Im Rahmen der Versuchsplanung werden die folgenden zweiten Ableitungen der Modellgleichungen bzw. Meßfunktionen nach den Zustandsvariablen x, den Parametern p sowie den Steuergrößen q benötigt[6]:

$$\frac{\partial^2 f}{\partial x \partial x}, \frac{\partial^2 f}{\partial p \partial x}, \frac{\partial^2 f}{\partial x \partial q}, \frac{\partial^2 f}{\partial p \partial q}$$

Es läßt sich erkennen, daß die zweite Ableitung nie nach den Parametern p erfolgt. Deshalb müssen auch bei den folgenden zweiten Ableitungen die partiellen Ableitungen nach Größen, die nur als Parameter auftreten, nicht betrachtet werden.

Ausgehend von den Termen der ersten Ableitung, werden nun die Terme der einzelnen Zuweisungen zur Berechnung der zweiten Ableitungen angegeben. Um die Ableitungen notieren zu können, verwendet man symbolisch u_1 für den Vektor, nach dem bei der ersten Ableitung differenziert wird, und u_2 für den Vektor, nach dem die zweite Ableitung bestimmt werden soll (beispielsweise $u_1 = p,\ u_2 = q$). Von jeder relevanten Zwischengröße v müssen daher drei verschiedene Ableitungsvektoren bestimmt werden:

$$\dot{v} := \frac{dv}{du_1}, \quad \tilde{v} := \frac{dv}{du_2}, \quad \ddot{v} := \frac{d^2 v}{du_1 du_2}$$

[6] Im folgenden steht die Funktion f für alle betrachteten Gleichungen.

Es werden nur partielle zweite Ableitungen angegeben, die von Null verschieden sind. Die Berechnung der Ableitungen vom Typ $\tilde{v}$ werden wie in Abschnitt 3.2.1 berechnet.

Für die zweiten Ableitungen der Gleichungen (3.5) und (3.6) gilt für $i = 1, \ldots, m$ mit den Hilfsvariablen s_1 und s_2 aus Gleichung (3.33) und S aus Gleichung (3.36):

$$\ddot{k}_i = \frac{\tilde{k}_i \cdot \dot{k}_{ref_i}}{k_{ref_i}} + \frac{\tilde{k}_i \cdot \dot{E}_{a_i} \cdot s_1}{E_{a_i}} + \frac{\tilde{T} \cdot \dot{E}_{a_i} \cdot k_i}{R \cdot T^2} + \frac{\tilde{k}_i \cdot \dot{T} \cdot E_{a_i}}{R \cdot T^2} - 2 \cdot \frac{\dot{T} \cdot \tilde{T} \cdot E_{a_i} \cdot k_i}{R \cdot T^3} \tag{3.42}$$

und für die Rückreaktion

$$\begin{aligned} \ddot{k}_i &= \frac{1}{k_{i-1}} \left(k_i \cdot \left(\frac{\ddot{k}_{i-1} - \tilde{k}_{i-1} \cdot \dot{k}_{i-1}}{k_{i-1}} \right) + \dot{k}_{i-1} \cdot \tilde{k}_i \right) - \frac{\dot{k}_{c_{i-1}} \cdot \tilde{k}_i}{k_{c_{i-1}}} \\ &- \frac{\Delta \dot{H}_{i-1} \cdot \tilde{k}_i \cdot s_2}{\Delta H_{i-1}} + \frac{\tilde{T} \cdot \Delta \dot{H}_i \cdot k_i}{R \cdot T^2} - \frac{\ddot{T} \cdot k_i \cdot \Delta H_{i-1}}{R \cdot T^2} \\ &- 2 \cdot \frac{T \cdot \tilde{T} \cdot \dot{T} \cdot k_i \cdot \Delta H_{i-1}}{R \cdot T^3} + \frac{\dot{T} \cdot \tilde{k}_i \cdot \Delta H_{i-1}}{R \cdot T^2} \end{aligned} \tag{3.43}$$

Der Term für die zweiten Ableitungen der Reaktionsgeschwindigkeit ergibt sich für alle $i = 1, \ldots, m$ mit den Indexmengen aus Abschnitt 3.1.1:

$$\begin{aligned} \ddot{r}_i &= \frac{1}{k_i} \left(\ddot{k}_i \cdot r_i + \dot{k}_i \cdot \tilde{r}_i - \frac{1}{k_i} \left(\dot{k}_i \cdot r_i \cdot \tilde{k}_i \right) \right) - \frac{S \cdot r_i}{V} \left(\frac{\ddot{V} - \dot{V} \cdot \tilde{V}}{V} + \frac{\dot{V} \cdot \tilde{r}_i}{r_i} \right) \\ &+ \tilde{k}_i \cdot V^{-S} \cdot s_3 - S \cdot \tilde{V} \cdot V^{(-S-1)} \cdot k_i \cdot s_3 \\ &+ k \cdot V^{-S} \cdot \sum_{j \in E_i} \left[\alpha_{ij} \cdot (\alpha_{ij} - 1) \cdot n_j^{(\alpha_{ij}-2)} \cdot \prod_{\substack{k \in E_i \\ k \neq j}} (n_k^{\alpha_{ik}}) \cdot \dot{n}_j \cdot \tilde{n}_j \right. \\ &\qquad + \alpha_{ij} \cdot n_j^{(\alpha_{ij}-1)} \prod_{\substack{k \in E_i \\ k \neq j}} (n_k^{\alpha_{ik}}) \cdot \ddot{n}_j \\ &\qquad \left. + \alpha_{ij} \cdot n_j^{(\alpha_{ij}-1)} \cdot \dot{n}_j \sum_{\substack{k \in E_i \\ k \neq j}} \left(\alpha_{ik} \cdot n_k^{(\alpha_{ik}-1)} \cdot \tilde{n}_k \prod_{\substack{l \in E_i \\ l \neq j, l \neq k}} n_l^{\alpha_{il}} \right) \right] \\ &+ \tilde{k}_i \cdot V^{-S} \cdot s_4 - \frac{\tilde{V} \cdot S \cdot s_5}{V} \\ &+ k \cdot V^{-S} \cdot \sum_{j \in E_i} \left[n_j^{(\alpha_{ij}-1)} \cdot \dot{\alpha_{ij}} \cdot \tilde{n}_j \prod_{\substack{k \in E_i \\ k \neq j}} (n_k^{\alpha_{ik}}) \cdot (1 + \log(n_j) \cdot \alpha_{ij}) \right. \\ &\qquad \left. + \log(n_j) \cdot n_j^{\alpha_{ij}} \cdot \dot{\alpha_{ij}} \sum_{\substack{k \in E_i \\ k \neq j}} \left(\alpha_{ik} \cdot n_k^{(\alpha_{ik}-1)} \cdot \tilde{n}_k \prod_{\substack{l \in E_i \\ l \neq j, l \neq k}} n_l^{\alpha_{il}} \right) \right] \\ &+ s_6 \cdot \left(\tilde{r}_i \cdot \log(V) + \frac{r_i \cdot \tilde{V}}{V} \right) \end{aligned} \tag{3.44}$$

Die zweiten Ableitungen der rechten Seite f (definiert durch Gleichung (3.38)) ergeben sich somit zu:

$$\begin{aligned}\ddot{f}_j &= \ddot{V}\cdot\sum_{i=1}^{m}\nu_{ij}\cdot r_i + \dot{V}\cdot\sum_{i=1}^{m}\nu_{ij}\cdot\tilde{r}_i + \tilde{V}\cdot\sum_{i=1}^{m}\nu_{ij}\cdot\dot{r}_i + V\cdot\sum_{i=1}^{m}\nu_{ij}\cdot\ddot{r}_i \\ &= \sum_{i=1}^{m}\nu_{ij}\left(\ddot{V}\cdot r_i + \dot{V}\cdot\tilde{r}_i + \tilde{V}\cdot\dot{r}_i + V\cdot\ddot{r}_i\right) \end{aligned} \tag{3.45}$$

Für die Berechnung der zweiten Ableitungen der Wärmebilanzgleichung (3.28) gilt:

$$\begin{aligned}\ddot{f}_1 &= \frac{1}{c_p\cdot M_{ges}}\cdot\Bigg[-\frac{\dot{V}\cdot\tilde{V}\cdot w_1}{V} + \sum_{i=0}^{m}\left(\Delta H_i\left(\ddot{r}_i\cdot V\cdot + r_i\cdot\tilde{V}\right) + \Delta\dot{H}_i\left(\tilde{r}_i\cdot V + r_i\cdot\tilde{V}\right)\right) \\ &\quad + c_p\cdot\sum_{k=1}^{l}\left(d\dot{M}_{e_k}\left(\tilde{T}_{e_k}-\tilde{T}\right) + d\dot{M}_{e_k}\cdot\dot{T}_{e_k} + dM_{e_k}\cdot\tilde{T}_{e_k} - \dot{T}\cdot d\dot{M}_{e_k}\right) \\ &\quad - \frac{1}{A_w^2}\cdot\left(k_d\left(\tilde{A}_w\cdot(T_w-T) + \dot{A}_w\left(\tilde{T}_w-\tilde{T}\right)\right)\dot{A}_w - \tilde{A}_w\cdot A_w\cdot w_3\right) \\ &\quad + k_d\cdot\tilde{A}_w\left(\dot{T}_w-\dot{T}\right) + s_1\cdot\ddot{T}_w\Bigg] \\ &\quad - \frac{\tilde{M}_{ges}}{M_{ges}}\left(\dot{f}_1 + \frac{\dot{M}_{ges}\cdot f_1}{M_{ges}}\right) - \frac{\dot{M}_{ges}\cdot\tilde{f}_1\cdot M_{ges} - \dot{M}_{ges}\cdot f_1\cdot\tilde{M}_{ges}}{M_{ges}^2}\end{aligned} \tag{3.46}$$

mit

$$\begin{aligned}\ddot{T}_w &= \frac{1}{(s_1+s_1)^2}\cdot\left[\left(\dot{T}\cdot k_d\cdot\tilde{A}_w + \dot{T}_{in}\cdot\tilde{T}\right)\cdot(s_1+s_2) - \left(\dot{T}\cdot s_1 + \dot{T}_{in}\cdot T\right)\cdot k_d\cdot\tilde{A}_w\right] \\ &\quad + \frac{1}{(s_2+s_2)^3}\cdot\Big[\Big(\tilde{T}\cdot\dot{A}_w\cdot k_d\cdot(s_1+s_2) + \dot{A}_w\cdot k_d^2\cdot T\cdot\tilde{A}_w \\ &\qquad - \left(k_d\cdot\tilde{A}_w\cdot T + \tilde{T}\cdot s_1\right)\cdot\dot{A}_w\cdot k_d\Big)\cdot(s_1+s_2) \\ &\qquad - \left(\dot{A}_w\cdot k_d\cdot T\cdot(s_1+s_2) - (s_1\cdot T + s_2\cdot T_{in})\right)\cdot 2\cdot k_d\cdot\tilde{A}_w\Big].\end{aligned}$$

Die benötigten Zwischenwerte $\tilde{f}_1$, $\tilde{V}$, $\tilde{M}_{ges}$, $\tilde{T}_w$ und $\tilde{A}_w$ berechnen sich analog wie in Abschnitt 3.2.1 beschrieben. Statt nach dem Vektor u_1 wird nach dem Vektor u_2 abgeleitet. Die Werte für $\tilde{T}$ und $\tilde{T}_{e_k}$ ergeben sich direkt aus den Richtungen, nach denen die zweite Ableitung erfolgt.

Kapitel 4

Implementierung

Zentraler Bestandteil dieser Diplomarbeit ist die Implementierung eines Modellgenerators für die in Abschnitt 3.1 beschriebenen chemischen Prozesse. Zu diesem Zweck wurde eine graphische Benutzeroberfläche implementiert, mit der das chemische Reaktionssystem spezifiziert werden kann. Anschließend werden alle Information über das Modell in einer *datenbankähnlichen* Datenstruktur abgespeichert. Danach werden die Modellgleichungen, welche die rechten Seiten der DAE (3.1) beschreiben, automatisch in Form eines Quelltextes in der Sprache Fortran77 generiert. Optional lassen sich auch alle erzeugten Quelltexte (Modellgleichungen und Ableitungen) in der Programmiersprache C durch das vorliegende Programm erzeugen. Die Anwendung der erzeugten Quelltexte in der am IWR implementierten Versuchsplanungssoftware, sowie ein Überblick über das Versuchsplanungsprogramm VPLAN98 wird in Abschnitt 4.2 beschrieben.

Im zweiten Teil der Implementierung werden die in Kapitel 2 erläuterten Prinzipien des Automatischen Differenzierens verwendet, um die ersten und zweiten Ableitungen der Modellgleichungen nach den Zustandsvariablen, Parametern und Steuergrößen zu berechnen. Um die Abhängigkeiten zwischen den einzelnen Variablen abzuspeichern, wird der in Abschnitt 2.1.4 beschriebene Abängigkeitsgraph für jede erzeugte Funktion (Modellgleichung oder Meßfunktion) aufgestellt. Dabei wird eine modifizierte Variante des Vorwärtsmodus aus Abschnitt 2.2.3 verwendet, indem die auftretenden Strukturen in den Gleichungen ausgenutzt werden, um einen möglichst effizienten Ableitungsquelltext zu erzeugen. Der Quelltext der erzeugten Ableitungen wird dabei aus dem Quelltext für die Modellgleichungen durch Hinzufügen von zusätzlichen Anweisungen für die Ableitungsberechnung generiert.

4.1 Allgemeines zur Implementierung

Das im Rahmen dieser Diplomarbeit entwickelte Programm *MGAD* ist in der objektorientierten Programmiersprache Java geschrieben. Es wird dabei die Version 1.1.6 des Java Development Kit (JDK) verwendet. Für die Erstellung der graphischen Benutzeroberfläche wird die *SWING 1.1* API[1] der Firma Sun Microsystems benutzt. Entwickelt wurde das Programm MGAD auf einem 586 AMD-K6 unter Linux 2.0.35. Der vollständige, ca.

[1] Application Programming Interface

12000 Zeilen umfassende Quelltext, wird im Rahmen dieser Diplomarbeit nur in Auszügen angegeben.

Die Entscheidung für die Programmiersprache Java basiert auf den nachfolgenden Überlegungen:

- Das rein objektorientierte Konzept der Sprache unterstützt die Modellierung eines komplexen Systems erheblich durch die Verwendung von eigenen Datentypen in Form von Klassen und den darauf operierenden Methoden. Die Vermischung von prozeduraler und objektorientierter Programmierung wie in anderen Sprachen (z.B. C++) ist in Java somit nicht möglich.

- Da die Interaktion des Programms mit der Versuchsplanungsssoftware VPLAN98 in Form einer *Client-Server Architektur* realisiert wird, muß eine Klassenbibliothek vorhanden sein, die netzwerkfähig ist. Dies ist mit den Standardklassen in Java gegeben.

- Die automatische Speicherverwaltung von dynamischen Objekten (*Garbage Collection*) und die fehlende Möglichkeit, auf Adressen von Objekten zuzugreifen, verhindert das Auftreten der häufigsten Laufzeitfehler, wie sie vor allem in C++-Programmen auftreten können. Dies ermöglicht eine weniger umfangreiche Fehlersuche und sehr robuste Programme.

- Java stellt mächtige Klassen zur dynamischen Verarbeitung von Zeichenketten (z.B. *StringBuffer*) zur Verfügung, die zur Erzeugung von umfangreichen Quelltexten unerläßlich sind.

- Durch die Verwendung der standardisierten SWING-API konnte die in Abschnitt 4.3.1 beschriebene graphische Benutzeroberfläche einfach und modular in Java programmiert werden.

- Java ist *architekturneutral*, d.h. es ist auf verschiedenen Systemen mit unterschiedlichen Prozessoren und Betriebssystemarchitekturen lauffähig. Der generierte Java-Bytecode kann auf jedem Prozessor, der einen Java-fähigen Browser bzw. die virtuelle Maschine von Java im allgemeinen unterstützt, ohne vorherige Neuübersetzung ausgeführt werden. Das vorliegende Programm wurde sowohl auf den Betriebssystemen UNIX (Linux 2.0.35, IRIX 6.3, AIX 3.4) als auch unter Windows (Windows 95 und Windows NT) eingesetzt. Das Programm ist sowohl als Java-Applikation als auch mit einem Browser (Netscape oder MS Internet Explorer) verwendbar.

- Die *Multithreading* Fähigkeit von Java erlaubt im Gegensatz zu Sprachen wie C oder C++, die nur die gleichzeitige Ausführung von einzelnen Programmen erlauben, die parallele Ausführung von mehreren einzelnen Programmschritten oder zusammenhängenden Prozessen.

- Der scheinbare Nachteil, daß Java eine Interpreter-Sprache ist, hat einen geringen Einfluß auf die Eignung für das vorliegende Programm, da hier durch die Sprache Java Quelltexte in einer kompilierten, maschinennahen Sprache wie Fortran77 oder C erzeugt werden. Die Laufzeit der erzeugten Ableitungsroutinen ist folglich von der Wahl der Programmiersprache zur Generierung der Ableitungsroutinen unabhängig.

4.2 Überblick über die Gesamtarchitektur von VPLAN98

Das in Abschnitt 1.3.5 angeführte Verfahren zur Berechnung der Nominaltrajektorie sowie der ersten und zweiten Ableitungen der Nominaltrajektorie des DAE-Systems (1.2) sind in dem BDF-Verfahren DAESOL (siehe BAUER (1996)) implementiert. Die effizienten Berechnungen der ersten und zweiten Ableitungen der Modellfunktionen f und g erfolgen durch den in Kapitel 4.3 beschriebenen und in dieser Arbeit implementierten Modellgenerator.

Einen Überblick über die bisher besprochenen Verfahren, ihre Implementierung durch Software-Pakete sowie die dafür benötigten Ableitungen gibt Abbildung 4.1. Die Suffixe vor den Namen der Routinen spezifizieren dabei die jeweiligen partiellen Ableitungen. Beispielsweise gilt:

$$\texttt{ffcn} \simeq f, \quad \texttt{x\_ffcn} \simeq \frac{\partial f}{\partial x} \quad \text{und} \quad \texttt{q\_p\_gfcn} \simeq \frac{\partial^2 g}{\partial q \partial p} \tag{4.1}$$

Mit der Versuchsplanungssoftware VPLAN98 lassen sich im einzelnen folgende Berechnungen durchführen:

- **Simulation**
  Die Auswertung und Bewertung von vom Benutzer spezifizierten Versuchsplänen. Es werden die Kovarianzmatrix, die Gütekriterien, die Konfidenzintervalle und die Kosten der Experimente berechnet. Zusätzlich wird überprüft, ob die an den Prozeß gestellten Nebenbedingungen erfüllt sind.

- **Versuchsplanung**
  Ausgehend von einem initialen Versuchsplan wird für einen spezifizierten Versuchsraum ein optimaler Versuchsplan erstellt. Informationen durch bereits durchgeführte Versuche können bei der Optimierung berücksichtigt werden. Die Lösung des Versuchsplanungsproblems erfolgt durch den SQP-Löser SNOPT.

- **Parameterschätzung**
  Mit dem Programm-Paket PARFIT lassen sich die unbekannten Parameter schätzen, so daß die experimentell erzeugten Meßdaten vom Modell optimal angepaßt werden.

- **Integration**
  Mit Hilfe des Integrators DAESOL kann das aufgestellte DAE-System effizient gelöst werden.

Bei der Durchführung dieser Berechnungen werden dabei die in Abbildung 4.1 dargestellten und in dieser Arbeit implementierten Ableitungsroutinen benötigt.

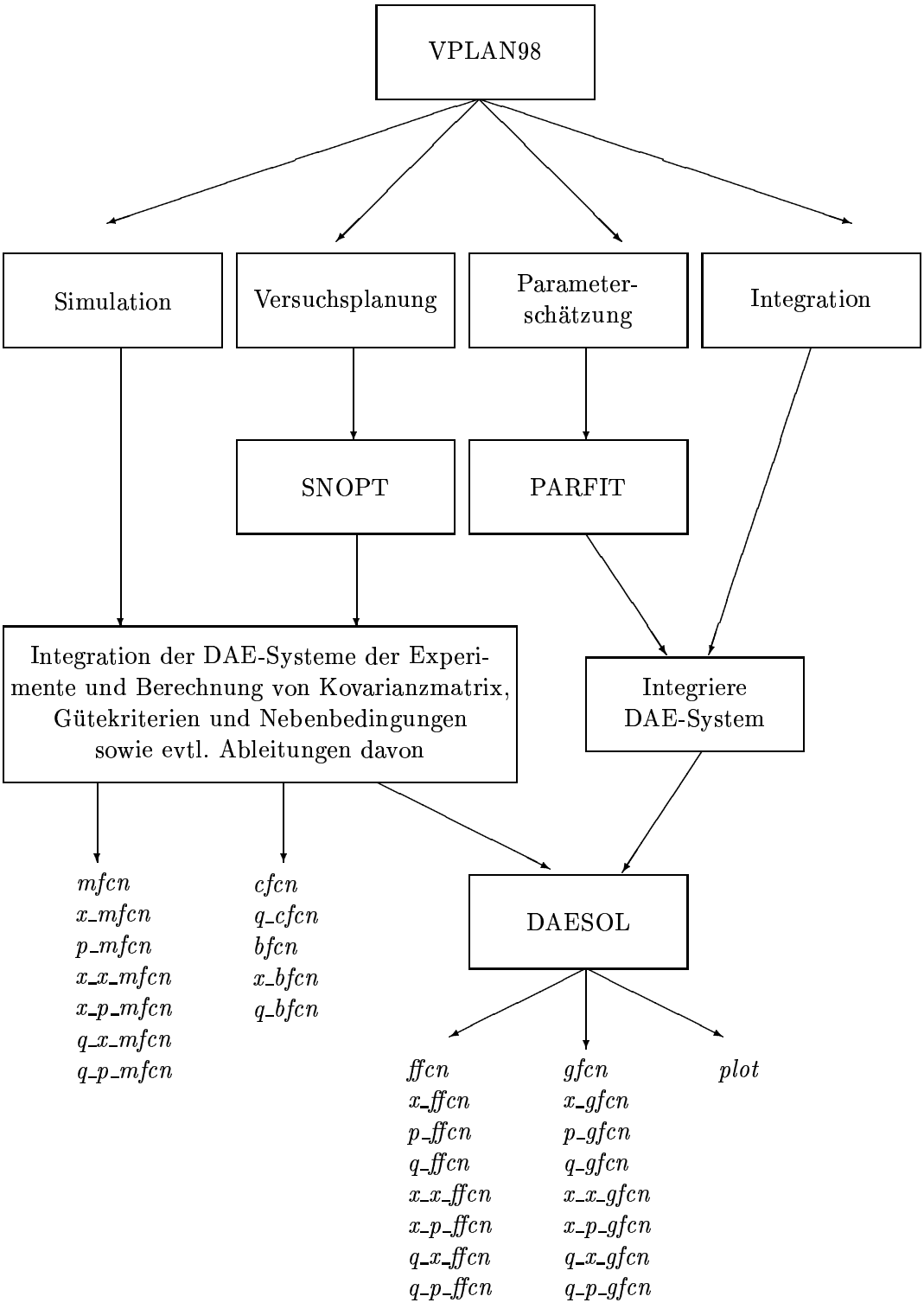


Abbildung 4.1: Aufbau der Gesamtarchitektur

4.3 Modellgenerator

4.3.1 Graphische Benutzeroberfläche

Um ein chemisches Reaktionssystem zu spezifizieren, müssen vom Benutzer eine Reihe von Informationen festgelegt werden. Dafür wurde im Rahmen der vorliegenden Diplomarbeit ein GUI[2] entwickelt, mit der die relevanten Größen für die Erstellung des chemischen Reaktionssystems eingegeben werden können. Die Programmierung erfolgt mit Hilfe der Graphikbibliothek SWING. Zusätzlich lassen sich bestimmte Größen als Parameter oder Steuergrößen festlegen.

Einstellung grundlegender Werte

Mit Hilfe dieser Startmaske werden die Anzahl der beteiligten Spezies, der Lösungsmittel und der Einträge des chemischen Reaktionssystems eingestellt. Die verschiedenen Ableitungsmöglichkeiten lassen sich hier auch spezifizieren.

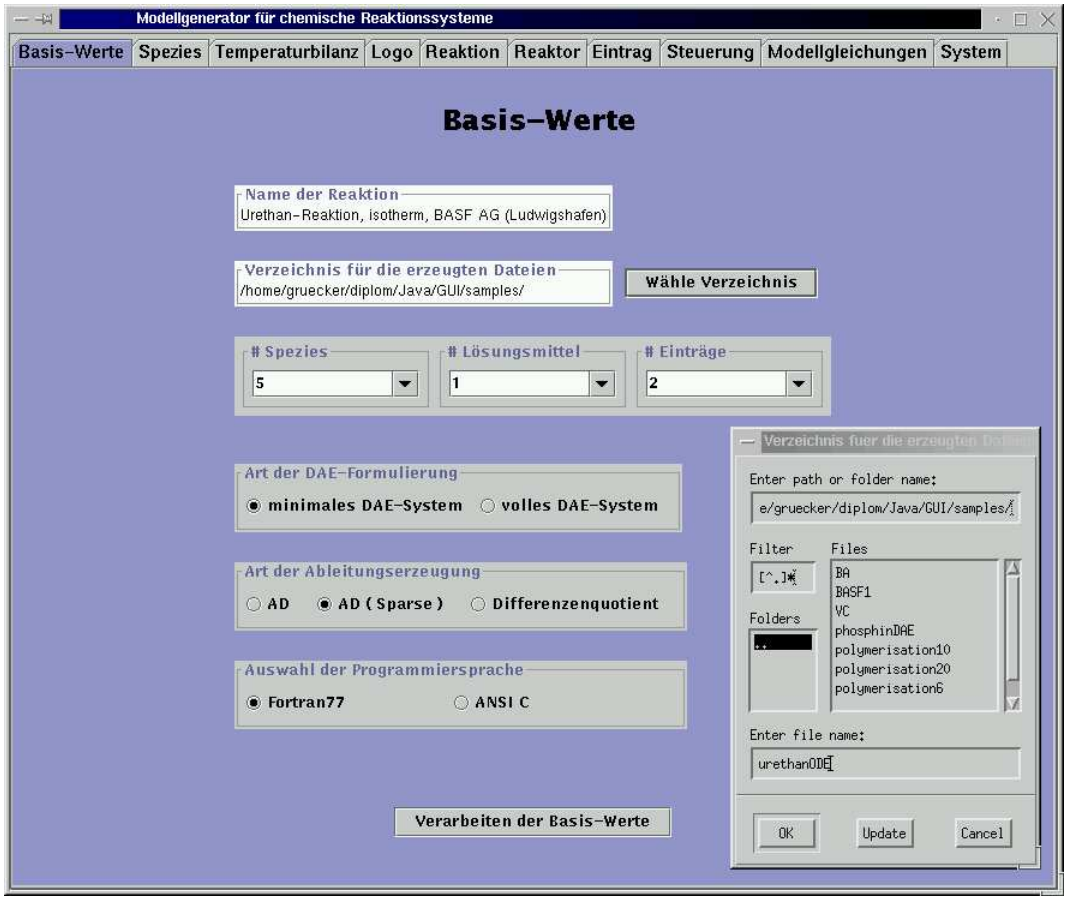


Abbildung 4.2: Eingabemaske Basiswerte

[2] Graphical User Interface

Spezies und Lösungsmittel

Für alle Spezies und Lösungsmittel müssen die stoffspezifischen Eigenschaften (Name, Molmasse, Dichte) bekannt sein. Dabei handelt es sich ausschließlich um Konstanten.

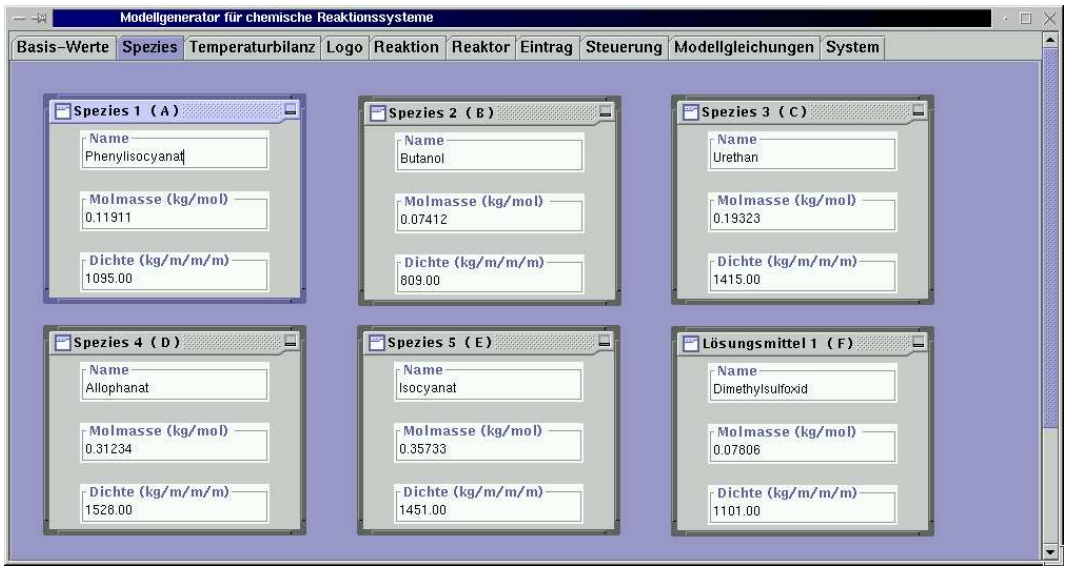


Abbildung 4.3: Eingabemaske Eigenschaften der Spezies

Temperaturbilanz

Bei nichtisothermen Reaktionsführungen wird mit folgender Eingabemaske die Temperaturbilanz spezifiziert.

Abbildung 4.4: Eingabemaske Temperaturbilanz im Reaktor

Spezifikation der Stöchiometrie der chemischen Reaktionen

Mit Hilfe eines Gleichungsgenerators lassen sich die einzelnen chemischen Reaktionen in einer symbolischen Notation (siehe Reaktionsschema (3.3)) eingeben. Für jede Reaktion werden in einem eigenen Fenster die reaktionsspezifischen Konstanten und Parameter eingelesen.

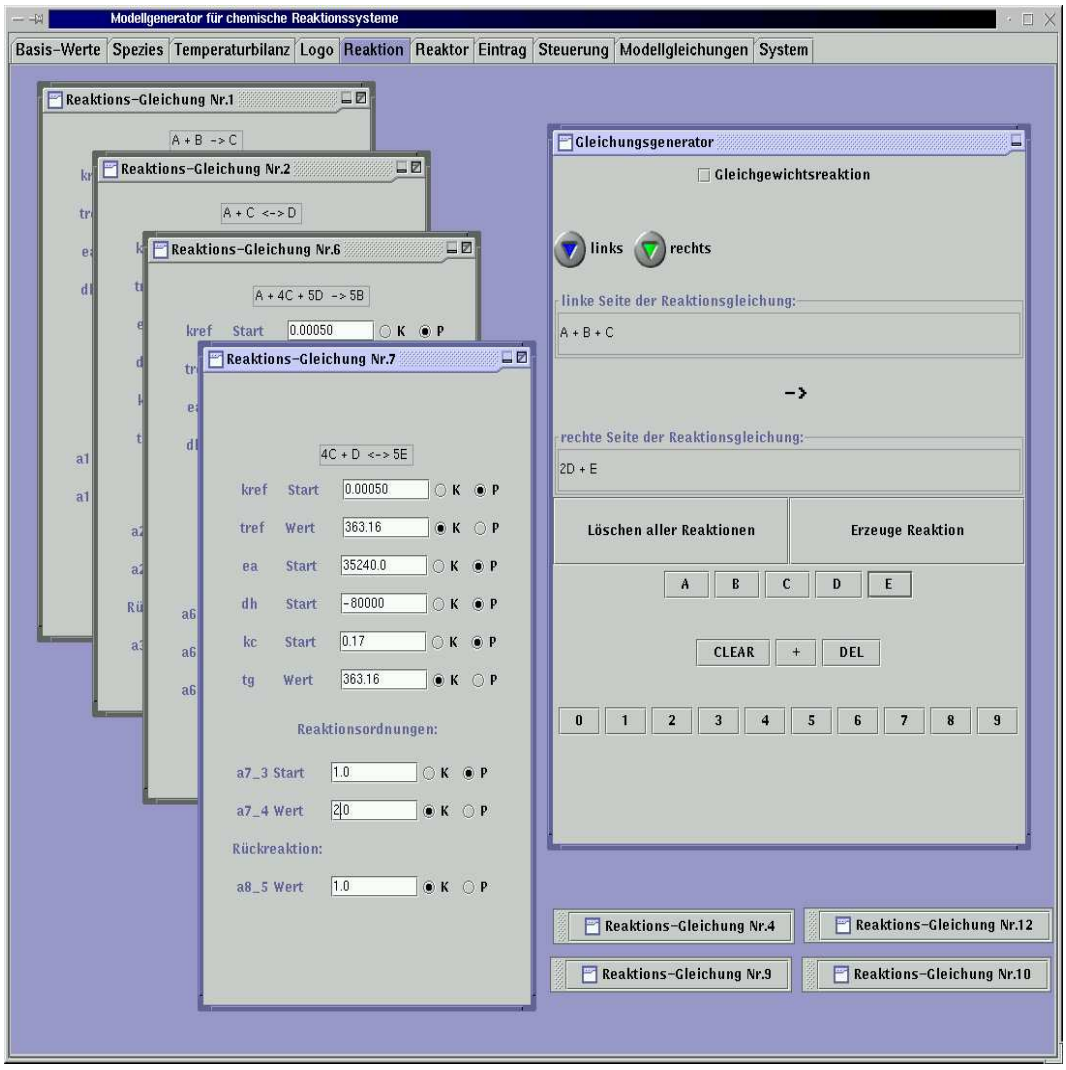


Abbildung 4.5: Eingabemaske der chemischen Reaktionen

Da man die symbolische Notation der chemischen Reaktionen nicht direkt in ein Eingabefeld eingeben kann, sondern die bereitgestellten Schaltflächen verwenden muß, können falsche Eingaben durch den Benutzer zur Laufzeit sehr leicht erkannt werden.

Spezifikation des Reaktors

Mit folgender Eingabemaske wird der Reaktor des Reaktionssystems näher bestimmt.

Abbildung 4.6: Eingabemaske Reaktor

Einträge am Reaktor

Die Festlegung der Stoffe in den einzelnen Einträgen erfolgt mit der Eingabemaske in Abbildung 4.7. Die Anfangsstoffmenge jeder Spezies kann konstant oder eine Steuergröße sein.

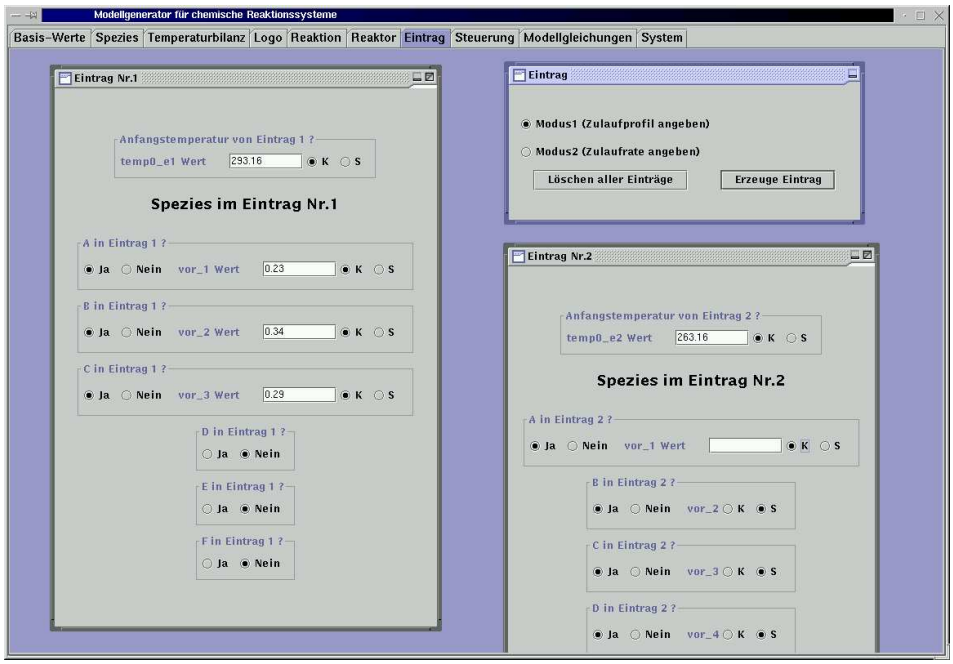


Abbildung 4.7: Eingabemaske Einträge

Ausgabe der erzeugten Modellgleichungen und Ableitungen

Mit Hilfe dieser Fenster können die einzelnen erzeugten Quelltexte ausgewählt und deren Inhalt in einem Editor-Fenster betrachtet werden. Der Benutzer kann hier auch Modifikationen an den erzeugten Quelltexten vornehmen. Der erzeugten Quelltexte für die Modellgleichungen sowie deren erste und zweite Ableitungen lassen sich aus einer aus einer Baumstruktur auswählen.

Abbildung 4.8: Eingabemaske Modellgleichungen und Ableitungen

4.3.2 Erzeugung der Quelltexte für die Modellgleichungen

Um die Kompatibilität zur bestehenden Software aufrechtzuerhalten, wurde Wert daraufgelegt, die bestehenden Schnittstellen zur Versuchsplanungssoftware VPLAN98 zu verwenden. In der Subroutine `ffcn` werden die Rechten Seiten der Differentialgleichungen bestimmt und in der Subroutine `gfcn` die rechte Seiten der algebraischen Gleichungen. Die Meßfunktionen werden in den Subroutinen `mfcn1`[3], `mfcn2`, ... spezifiziert. Die Parameterlisten lauten wie folgt:

```
subroutine ffcn( t, x, f, p, q, rwh, iwh, iflag )
subroutine gfcn( t, x, g, p, q, rwh, iwh, iflag )
subroutine mfcn1( t, x, h, p, q, rwh, iwh, iflag )
```

Dabei gilt im Falle eines Fortran77 Quelltextes:

`real*8 t`	Zeit
`real*8 x`	Vektor aller Zustandsvariablen
`real*8 f`	Vektor der rechte Seiten der Differentialgleichungen
`real*8 g`	Vektor der rechte Seiten der algebraischen Gleichungen
`real*8 h`	Meßfunktionswert
`real*8 p`	Vektor der Parameter
`real*8 q`	Vektor der Steuergrößen
`integer*4 rhw` `integer*4 iwh`	Hilfsvektoren für die Parametrisierung der Steuerfunktionen
`integer*4 iflag`	Fehlerflag

Die abstrakte[4] Klasse `Generator` implementiert für jede Zuweisung zur Generierung der Modellgleichungen eine eigene Methode. Mit dieser Methode werden die sprachspezifischen Anweisungen zur Quelltext-Generierung erzeugt. Die Klassen `GenFfcn`, `GenGfcn` und `GenMessSig` sind von der Basisklasse `Generator` abgeleitet und erzeugen durch Zugriff auf die Methoden der Basisklasse die jeweiligen Quelltexte.

Die folgende Java-Methode erzeugt zum Beispiel die Quelltext-Zeichenkette mit den Instruktionen zur Berechnung der Molzahländerungen Δn_i (siehe Abschnitt 3.1.2) der Spezies des DAE-Systems:

```
/**
 * @return String - String-Repraesentation der Delta n_i-Zuweisungen
 */
public String doDeltaN(){
    StringBuffer str_buf = new StringBuffer( "" );
    StringBuffer str_buf_right = new StringBuffer( "" );
    int k;
    int n = n_spec;    // Anzahl Spezies

    str_buf.append( F.mComm( "MOLZAHLAENDERUNG" ) + "\n\n" );
    for( int i = 0; i < n; i++ ){
        if( !react_sys.isZustands( i ) )    // nur differentielle ZV
```

[3] Im folgenden steht die Routine *mfcn1* für alle vorkommenden Meßfunktionen.

[4] Eine abstrakte Klasse in Java ist eine Klasse, von der kein Objekt instanziert werden kann, von der aber Subklassen abgeleitet werden können, die alle Methoden und Attribute der abstrakten Klasse erben.

```
            if( !react_sys.isDAE() )        // im reinen DAE-Fall alle ZV
                continue;
        String l_var =  N.delta_n + (i+1);
        str_buf_right = new StringBuffer( "" );
        str_buf_right.append( N.konz_spec + ( i + 1 ) );
        makeDepend( l_var, N.konz_spec + ( i + 1 ) );

        // Spezies in Vorlage  ?
        if( react_sys.isInVorlage( i ) ){
            str_buf_right.append( " - " + N.vorlage + ( i + 1 ) );
            makeDepend( l_var, N.vorlage + ( i + 1 ) );
        }
        // Spezies in Eintraege ?
        for( k = 0; k < n_e; k++ )
            if( kessel.isInEintrag( k, i ) ){
                str_buf_right.append( " - " + genSpecEintragName( k, i ) + ' ' );
                makeDepend( l_var, genSpecEintragName( k, i ) );
            }
        str_buf.append( mB( 2 ) + F.makeEquation(
                                l_var, str_buf_right.toString() ) + '\n' );
        // erste Ableitungen berechnen
        if( mode > 0 )
            str_buf.append( genDiff.DDeltaN( l_var, i, react_sys, 1 ) );
        // zweite Ableitungen
        if( mode2 > 0 )
            str_buf.append( genDiff.DDeltaN( l_var, i, react_sys, 2 ) );
    }
    return str_buf.toString();
}
```

Als Beispiel für einen automatisch erzeugten Quelltext, durch den vorliegenden Modellgenerator in der Programmiersprache Java, sei der folgende Fortran77-Quelltext zur Berechnung der rechten Seiten des Differentialgleichungssystems der Urethan-Reaktion (siehe Abschnitt 3.3) angefügt.

4.3.3 Automatisch erzeugter Quelltext einer Modellgleichung

```
c     ------------------------------------------------------------------
c     Automatisch erzeugtes Fortran77 File ffcn.f
c     durch Java Modellgenerator
c     Gerd Ruecker (IWR der Universitaet Heidelberg)
c     Fri Oct 15 01:37:02 GMT+03:30 1999

c     Urethan-Reaktion (ODE)

c     # Reaktionen:               3
c     # Spezies:                  5
c     # Loesungsmittel:           1
c
c     # Differentialgleichungen:  3
c     # Algebraische Gleichungen: 3
c
c     # Quelltext Zeilen:         208
c     # chemische Variablen       76
c     ------------------------------------------------------------------


      subroutine ffcn( t, x, f, p, q, rwh, iwh, iflag )

         implicit none
```

```
c     DEKLARATIONEN

         real*8 t, x(*), f(*), p(*), q(*), rwh(*)
         integer*4 iwh(*), iflag

         real*8 tref1, tref2, tref4, tg2, kref1, kref2, kref4, kc2, ea1,
      & ea2, ea4, alpha1_1, alpha1_2, alpha2_1, alpha2_3, alpha3_4, alpha
      &4_1, tmpval, tmp1(6), rgas, pi, dh1, dh2, dh3, dh4, mm1, rho1, del
      &ta_n1, mm2, rho2, delta_n2, mm3, rho3, delta_n3, mm4, rho4, mm5, r
      &ho5, mm6, rho6, na1, na2, na6, na1e1, n1e1, na6e1, n6e1, na2e2, n2
      &e2, na6e2, n6e2, feed_1, d_feed_1, feed_2, d_feed_2, d_n1e, d_n2e,
      & d_n3e, temp, tin, n1, n2, n3, n4, n5, n6, r1, k1, r2, k2, r3, k3,
      & r4, k4, v, m

         integer*4 ind, ind_1, ind1(2), ind2(2)

c     ZUSTANDSVARIABLEN

         n1 = x(1)
         n2 = x(2)
         n3 = x(3)

c     SKALIERTE PARAMETER

         kref1 = p(1) * 5.0d-4
         ea1 = p(2) * 35240.0d+0
         kref2 = p(3) * 8.0d-8
         ea2 = p(4) * 85000.0d+0
         dh2 = p(7) * (-17031.0d+0)
         kc2 = p(8) * 0.17d+0
         kref4 = p(5) * 1.0d-8
         ea4 = p(6) * 35000.0d+0

c     STEUERGROESSEN

         na1 = q(1)
         na2 = q(2)
         na6 = q(3)
         na1e1 = q(4)
         na6e1 = q(5)
         na2e2 = q(6)
         na6e2 = q(7)

c     DISKRETISIERUNG DER STEUERFUNKTIONEN

c     AUSSENTEMPERATUR DES REAKTORS

         if ( iwh(3) .eq. 0 .or. iwh(3) .eq. 1 ) then
            ind =  iwh(2) + iwh(4) - 1
            ind_1 = 0
            tin = q(ind)
         else if ( iwh(3) .eq. 2 .or. iwh(3) .eq. 3 ) then
            ind =  iwh(2) + 2*iwh(4) - 2
            ind_1 = ind + 1
            tin = q(ind) + ( t-rwh(2) ) * q(ind_1)
         endif

c     EINTRAEGE AM REAKTOR

         if ( iwh(6) .eq. 0 .or. iwh(6) .eq. 1 ) then
            ind1(1) =  iwh(5) + iwh(7) - 1
            ind2(1) = 0
            feed_1 = q(ind1(1))
            d_feed_1 = 0.0d+0
         else if ( iwh(6) .eq. 2 .or. iwh(6) .eq. 3 ) then
```

```
            ind1(1) =  iwh(5) + 2*iwh(7) - 2
            ind2(1) = ind1(1) + 1
            feed_1 = q(ind1(1)) + ( t-rwh(3) ) * q(ind2(1))
            d_feed_1 = q(ind2(1))
         endif

         if ( iwh(9) .eq. 0 .or. iwh(9) .eq. 1 ) then
            ind1(2) =  iwh(8) + iwh(10) - 1
            ind2(2) = 0
            feed_2 = q(ind1(2))
            d_feed_2 = 0.0d+0
         else if ( iwh(9) .eq. 2 .or. iwh(9) .eq. 3 ) then
            ind1(2) =  iwh(8) + 2*iwh(10) - 2
            ind2(2) = ind1(2) + 1
            feed_2 = q(ind1(2)) + ( t-rwh(4) ) * q(ind2(2))
            d_feed_2 = q(ind2(2))
         endif

c     KONSTANTEN

         alpha1_1 = 1.0d+0
         alpha1_2 = 1.0d+0
         tref1 = 363.16d+0
         dh1 = -80000.0d+0
         alpha2_1 = 1.0d+0
         alpha2_3 = 1.0d+0
         alpha3_4 = 1.0d+0
         tref2 = 363.16d+0
         tg2 = 363.16d+0
         alpha4_1 = 2.0d+0
         tref4 = 363.16d+0
         dh4 = -75000.0d+0
         mm1 = 0.11911d+0
         rho1 = 1095.0d+0
         mm2 = 0.07412d+0
         rho2 = 809.0d+0
         mm3 = 0.19323d+0
         rho3 = 1415.0d+0
         mm4 = 0.31234d+0
         rho4 = 1528.0d+0
         mm5 = 0.35733d+0
         rho5 = 1451.0d+0
         mm6 = 0.07806d+0
         rho6 = 1101.0d+0
         rgas = 8.314d+0
         pi = 3.141592653589793d+0

c     HILFSGROESSEN

         temp = tin

         n1e1 = na1e1 * feed_1
         n6e1 = na6e1 * feed_1
         n2e2 = na2e2 * feed_2
         n6e2 = na6e2 * feed_2

c     MOLZAHLAENDERUNGSRATE

         delta_n1 = n1 - na1 - n1e1
         delta_n2 = n2 - na2 - n2e2
         delta_n3 = n3

c     ALGEBRAISCHE GLEICHUNGEN

         n6 = na6 + n6e1 + n6e2
```

```
        n5 =  - 0.3333333333333333d+0*delta_n1 + 0.6666666666666666d+0*
     &delta_n2 + 0.3333333333333333d+0*delta_n3
        n4 =  - delta_n2 - delta_n3

c     GESAMTVOLUMEN IM REAKTOR

        v =  + n1*mm1/rho1 + n2*mm2/rho2 + n3*mm3/rho3 + n4*mm4/rho4 +
     &n5*mm5/rho5 + n6*mm6/rho6


c     ARRHENIUS KINETIK

        k1 = kref1 * dexp( -ea1/rgas*(1.0d+0/temp - 1.0d+0/tref1) )
        k2 = kref2 * dexp( -ea2/rgas*(1.0d+0/temp - 1.0d+0/tref2) )
        k3 = k2 / (  kc2 * dexp( -dh2/rgas*(1.0/ temp - 1.0/tg2) ) )
        k4 = kref4 * dexp( -ea4/rgas*(1.0d+0/temp - 1.0d+0/tref4) )


c     REAKTIONS-GESCHWINDIGKEITEN

        r1 = k1 * ( n1/v ) * ( n2/v )
        r2 = k2 * ( n1/v ) * ( n3/v )
        r3 = k3 * ( n4/v )
        r4 = k4 * ( n1/v )**alpha4_1

c     GESAMTZULAUFRATE DER SPEZIES

        d_n1e = na1e1 * d_feed_1
        d_n2e = na2e2 * d_feed_2
        d_n3e = 0d+0

c     RECHTE SEITEN DES DIFFERENTIALGLEICHUNGSSYSTEMS

        f(1) = v * ( - r1 - r2 + r3 - 3.0d+0*r4 ) + d_n1e
        f(2) = v * ( - r1 ) + d_n2e
        f(3) = v * ( + r1 - r2 + r3 ) + d_n3e

        iflag = 0

      end
```

4.4 Automatische Erzeugung der ersten Ableitungen

Nachdem im ersten Teil des Programms die Modell- und Meßroutinen `ffcn`, `gfcn` und `mfcn1` aus einer chemischen Spezifikation erstellt wurden, werden jetzt Richtungsableitungen automatisch in Form von Quelltexten erzeugt. Für die Modell- und Meßfunktionen `ffcn` und `gfcn` müssen also die Ableitungsquelltexte für die Richtungsableitungen nach `x`, `p`, und `q` erzeugt werden. Die Aufruflisten der erzeugten Routinen ergeben sich aus den Aufruflisten für die Modellgleichungen, erweitert um zusätzliche Variablen. Als Parameterliste wurde die gleiche wie bei der Verwendung von ADIFOR (siehe BISCHOF ET AL. (1995)) gewählt. Zum einen müssen somit die Aufruflisten der Ableitungsroutinen in den Programmpaketen VPLAN98 und DAESOL (siehe Abbildung 4.1) nicht verändert werden, zum anderen können damit auch leicht Laufzeitvergleiche zwischen dem im Rahmen der Arbeit erstellten Programm und ADIFOR gemacht werden (siehe dazu auch die numerischen Ergebnisse in Kapitel 5).

Als Beispiel sei die Parameterliste von `ffcn`, abgeleitet nach dem Vektor der Zustandsvariablen `x`, angegeben.

```
subroutine x_ffcn( x_d_, t, x, x_x, ldx_x, f, x_f, ldx_f, p, q, rw
&h, iwh, iflag )
```

Die zusätzlichen Variablen sind folgendermaßen definiert:

`real*8 x_d_`	Anzahl der Richtungen, nach denen abgeleitet wird
`real*8 x_x(ldx_x,*)`	Matrix mit den einzelnen Richtungen (Seed-Matrix)
`real*8 ldx_x`	Führende Dimension von `x_x`
`real*8 x_f(ldx_f,*)`	Richtungsableitungsmatrix mit dem Matrix-Matrix Produkt aus Jacobi-Matrix und `x_x`
`real*8 ldx_f`	Führende Dimension von `x_f`

Die Parameterlisten für die Ableitungen nach `p` und `q` (`p_ffcn` und `q_ffcn`) sowie die korrespondierenden Ableitungen der Modellfunktion `gfcn` werden analog konstruiert.

Um nun die Ableitungen automatisch zu ermitteln, wird zuerst der in Abschnitt 2.1.4 definierte Abhängigkeitsgraph G bestimmt. Alle zur Berechnung der abhängigen Variablen `f`, `g` oder `h` auftretenden unabhängigen Variablen `x`, `p`, `q` sowie alle auftretenden Zwischengrößen bilden die Knoten des gerichteten, azyklischen Graphs G. Eine Kante e verläuft genau dann von einem Knoten v_i nach v_j, wenn die Größe v_j direkt von v_i abhängt. Bei der Generierung der einzelnen Ableitungsinstruktionen wird dann der Graph G verwendet, um die Berechnung in einem Vorwärtslauf (siehe Abschnitt 2.2.3) durchzuführen:

Im folgenden bezeichne der Vektor u_1 den Vektor derjenigen Variablen, nach denen jeweils abgeleitet wird. Dies kann im konkreten Fall einer der drei Vektoren `x`, `p` oder `q` sein. Um die Richtungsableitungen in einem Vorwärtslauf zu ermitteln, wird jede einzelne Instruktion des Quelltextes untersucht und eventuell nach der Kettenregel abgeleitet. Als aktive Variable wird eine Größe bezeichnet, die direkt oder indirekt von mindestens einer Komponente von x_d abhängt. Ist dabei die Variable v auf der linken Seite aktiv, so wird dieser Variablen ein Gradientenobjekt d_v (siehe Gleichung (2.15)) zugeordnet. Zur Berechnung aller einzelnen Komponenten von d_v müssen nach Gleichung (2.14) die partiellen Ableitungen aller aktiven Variablen berechnet und aufsummiert werden. Um zu testen, ob eine Variable aktiv ist, wird der Abhängigkeitsgraph G verwendet. Eine Variable v ist nach Gleichung (2.8) genau dann aktiv, wenn es mindestens einen Weg im Graphen G von einer beliebigen Komponente des Vektors u_1 nach v gibt.

Die folgende Java-Methode erzeugt die Anweisungen für die Richtungsableitungen der Gesamtmasse M_{ges} nach u_1:

```
/**
 * @return String - String-Repraesentation der ersten Ableitung der Gesamtmasse
 * @param l_var - Namen der Variablen auf der rechten Seite
 * @param n_s   - Anzahl der Spezies
 *
 */
public String DMass( String l_var, int n_s ){
    String str1, str2;
    StringBuffer diff_str_buf = new StringBuffer( "" ); // Hilfspuffer

    if( !doAv( l_var ) )          // haengt linke Seite von abzuleitendem Vektor ab ?
        return "";
    else
        ind_vec_save = ind_vec;   // Hilfsvektor zur Speicherung der Indices
                                  // fuer den Sparse-Modus
```

```
        for( int i = 0; i < n_s; i++ ){     // durchlaufe alle Spezies
            str1 = N.konz_spec + ( i + 1 ); // Name der Spezies
            str2 = N.mol + ( i + 1 );       // Name der Molmasse
            if( doAv( str1 ) )         // haengt Variable vom abzuleitenden Vektor ab ?
                                       // -> berechne partielle Ableitung
                diff_str_buf.append( " + " + gNoDV( str1 ) + "*" + str2 );
            if( doAv( str2 ) )         // haengt Variable vom abzuleitenden Vektor ab ?
                diff_str_buf.append( " + " + gNoDV( str2 ) + "*" + str1 );
        }
        // Erzeuge Schleife fuer die Gradientenberechnung
        return F.makeLoop( laufDir_str, "1", numDir_str,
                 F.makeEquation( gNoDV( l_var ), diff_str_buf.toString() ),
                                          1, ind_vec_save ) + '\n' ;
}
```

Die resultierenden Fortran77-Anweisungen sehen dann wie folgt aus:

```
C     ABLEITUNG DER GESAMTMASSE NACH ZUSTANDSVARIABLEN
        m = + n1*mm1 + n2*mm2 + n3*mm3 + n4*mm4
        do x_i_ = 1, x_d_
           x_m(x_i_) =  + x_n1(x_i_)*mm1 + x_n2(x_i_)*mm2 + x_n3(x_i_)*
     &mm3
        enddo
```

Da die Struktur der abzuleitenden Modellgleichungen durch die Modellerstellung bekannt ist, können die Ableitungen sehr effizient erzeugt werden. Um möglichst effiziente Ableitungsquelltexte zu generieren, wurden bei den einzelnen Instruktionen die folgende Prinzipien angewendet:

Verwendung von temporären Variablen:

Da durch die Verwendung von temporären Variablen die redundanten Berechnungen in Schleifen vermieden werden, ergibt sich auch eine verringerte Laufzeit des Programmes. Der folgende Auszug aus dem Quelltext `x_ffcn` demonstriert dieses Prinzip:

```
c     REAKTIONS-GESCHWINDIGKEITEN

         r1 = k1 * ( n1/v ) * ( n2/v )
         tmpval = alpha1_1 + alpha1_2
         tmpval1 = k1 * 1.0d+0 / ( v * v )
         tmpval2 = tmpval * r1 / v
         tmp3(1) = tmp1(1) * tmp2(1)
         tmp3(2) = tmp1(2) * tmp2(2)

C        SCHLEIFE UEBER DIE RICHTUNGEN
         do x_i_ = 1, x_d_
            tmp(2) = -x_v(x_i_) * tmpval2
            tmp4(1) = tmp3(1) * x_n1(x_i_)
            tmp4(2) = tmp3(2) * x_n2(x_i_)
            tmpval2 = tmp4(1) + tmp4(2)
            tmp(3) = tmpval1 * tmpval2
C           ZUWEISUNG DES GRADIENTEN
            x_r1(x_i_) = tmp(2) + tmp(3)
         enddo
```

Effizientes Ableiten von Termen mit vielen Multiplikationen:

Sei dazu f definiert als:

$$f(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} x_i$$

Die Berechnung der Funktion benötigt n Multiplikationen. Die Bestimmung der partiellen Ableitung von f nach x_1 erfolgt mit einer einzigen Division und einer Multiplikation, falls die Variable x_1 an keiner Stelle Null ist.

$$\frac{\partial f}{\partial x_1} = f/x_1 \cdot \dot{x}_1$$

Um die Ableitung auf diese Weise zu berechnen, muß daher immer getestet werden, ob die Variable nicht Null ist. Als ein Beispiel dient der folgende Fortran77-Code-Auszug. Dabei wurden die partiellen Ableitungen der rechten Seite `f` nach `v`, `aw` und `m` nach diesem Prinzip erzeugt. Diese Variablen können zu keinem Zeitpunkt Null werden, so daß die Division durch diese Variablen erlaubt ist:

```
        w_reak = v * ( + r1*dh1 )
        w_feed =  + dme1*cp*(temp0_e1-temp)
        w_cool =  + kd*aw*(tw-temp)
        f(1) = ( w_reak + w_feed + w_cool ) / ( m*cp )

C     BERECHNUNG DES GRADIENTEN DER RECHTEN SEITEN
        tmp(1) = w_reak/v
        tmp(2) = v*dh1
        tmp(3) = dme1*cp
        tmp(4) = w_cool/aw
        tmp(5) = kd*aw
        tmp(6) = m*cp
        tmp(7) = f(1)/m
        do x_i_ = 1, x_d_
           x_f(x_i_, 1) =  ( x_v(x_i_) * tmp(1) + x_r1(x_i_) * tmp(2) -
     &x_temp(x_i_)*tmp(3) + x_aw(x_i_) * tmp(4) + x_tw(x_i_) * tmp(5) -x
     &_temp(x_i_)*tmp(5) ) / tmp(6) - x_m(x_i_) * tmp(7)
        enddo
```

Ein Beispiel für einen vollständigen Quelltext (`x_ffcn`) der ersten Ableitung von `ffcn` nach den Zustandsvariablen `x` findet man im Anhang A.1.1.

4.5 Automatische Erzeugung der zweiten Ableitungen

Die Generierung der zweiten Ableitungen erfolgt analog zur Berechnung der ersten Ableitungen. Dabei wird der Quelltext der ersten Ableitungen um Instruktionen zur Berechnung der zweiten Ableitungen erweitert.

Die Parameterliste der ersten Ableitung wird dabei um Variablen für die zweiten Ableitungen ergänzt. Als Beispiel wird die Parameterliste der Routine `q_x_ffcn` angegeben:

`real*8 q__d_`	Anzahl Richtungen bei der zweiten Ableitung
`real*8 q__q(ldq__q,*)`	Matrix mit den einzelnen Richtungen für die zweite Ableitung in den Zeilen
`real*8 ldq__q`	Führende Dimension von `q__q`
`real*8 q_x_f(ldq_x_f,ldx_f,*)`	Richtungsableitungstensor mit dem Produkt aus Ableitungstensor und `q__q`
`real*8 ldq_x_f`	Führende Dimension von `q_x_f`

Um die einzelnen partiellen zweiten Ableitungen zu erzeugen, wird wieder der Abhängigkeitsgraph G verwendet. Die Berechnung der partiellen zweiten Ableitung $\frac{\partial^2 v}{\partial u_1 \partial u_2}$ einer Variable v auf der rechten Seite einer Zuweisung erfolgt genau dann, wenn v von u_1 *und* u_2 abhängt.

Der folgende Auszug aus der Datei `q_x_ffcn` veranschaulicht die Vorgehensweise exemplarisch. Berechnet wird dabei die zweite Richtungsableitung für die Zwischengröße `r_1`.

```
      r1 = k1 * ( n1/v ) * ( n2/v )
      tmpval = alpha1_1 + alpha1_2
      tmpval1 = 1.0d+0 / ( v * v )
      tmpval2 = tmpval * r1 / v
      tmp7(1,2) = 1.0d+0
      tmp1(1) = n2
      tmp2(1) = 1.0d+0
      tmp5(1) = ( alpha1_1 - 1.0d+0 ) * 1.0d+0 * n1**(alpha1_1-2.0d+0
     &) * tmp1(1)
      tmp3(1) = tmp1(1) * tmp2(1)
      tmp7(2,1) = 1.0d+0
      tmp1(2) = n1
      tmp2(2) = 1.0d+0
      tmp5(2) = ( alpha1_2 - 1.0d+0 ) * 1.0d+0 * n2**(alpha1_2-2.0d+0
     &) * tmp1(2)
      tmp3(2) = tmp1(2) * tmp2(2)

      do q__i_ = 1, q__d_
         tmp(1) = q__k1(q__i_) * r1 / k1
         tmp(2) = -q__v(q__i_) * tmpval * r1 / v
         q__r1(q__i_) = tmp(1) + tmp(2)
      enddo
      do x_i_ = 1, x_d_
         tmp(2) = -x_v(x_i_) * tmpval2
         tmp4(1) = tmp3(1) * x_n1(x_i_)
         tmp6(1) = tmp5(1) * x_n1(x_i_) * tmp1(1)
         tmp8(1) = tmp2(1) * x_n1(x_i_)
         tmp4(2) = tmp3(2) * x_n2(x_i_)
         tmp6(2) = tmp5(2) * x_n2(x_i_) * tmp1(2)
         tmp8(2) = tmp2(2) * x_n2(x_i_)
         tmpval2 = tmp4(1) + tmp4(2)
         tmp(3) = k1 * tmpval1 * tmpval2
         x_r1(x_i_) = tmp(2) + tmp(3)
         do q__i_ = 1, q__d_
            tmpval4 = q__k1(q__i_) * r1 / k1-q__v(q__i_) * tmpval * r
     &1 / v
         block(2) = - tmpval * r1 / v * ( q_x_v(q__i_,x_i_) - q__v(q_
     &_i_) * x_v(x_i_) / v + x_v(x_i_) * tmpval4 / r1 )
         block(3) = q__k1(q__i_) * tmpval1 * tmpval2 - q__v(q__i_) *
     &tmpval * tmp(3) / v + k1 * tmpval1 * ( tmp6(1) * q__n1(q__i_) + tm
     &p3(1) * q_x_n1(q__i_,x_i_) + tmp6(2) * q__n2(q__i_) + tmp3(2) * q_
     &x_n2(q__i_,x_i_) )
         q_x_r1(q__i_,x_i_) = block(2) + block(3)
         enddo
      enddo
```

Ein Beispiel für einen vollständigen Quelltext (`q_x_ffcn`) der zweiten Ableitung von `ffcn` nach den Zustandsvariablen `x` und den Steuergrößen `q` findet man im Anhang A.1.2.

Kapitel 5

Numerische Ergebnisse

5.1 Bemerkungen zur Testumgebung

In diesem Kapitel wird das erstellte Programm an verschiedenen praxisnahen Beispielen aus der chemischen Industrie auf Effizienz getestet. Die erzeugten Ableitungen aus Abschnitt 4.4 und 4.5 werden dabei mit den Ableitungen des Software-Paketes ADIFOR sowie mit Numerischen Differenzen verglichen.

Die Tests wurden auf einem AMD-K6/233 mit 128 MB Arbeitsspeicher unter Linux 2.0.35 durchgeführt. Die Zeitmessung erfolgt mit dem UNIX-Programm *time*. Da die Messung der Laufzeit einer einzigen Routine unterhalb der Meßgenauigkeit liegt, wird aus einem Treiberprogramm die jeweilige Routine aufgerufen und die dortigen Anweisungen mehrmals (10 000 mal) in einer Schleife ausgeführt. Die Angabe der Laufzeiten erfolgt in Sekunden.

Bei den vorgestellten Beispielen handelt es sich um Reaktionssysteme, die sich durch die Modellierung aus Kapitel 3 beschreiben lassen. Es handelt sich somit um eine Klasse von Reaktionssystemen, die sich durch den in dieser Arbeit entwickelten Modellgenerator mathematisch beschreiben lassen. Die Reaktionssysteme wurden gemeinsam vom IWR und der BASF AG (Ludwigshafen) modelliert. Die Optimierungsaufgaben wurden mit der Versuchsplanungssoftware VPLAN98 (KÖRKEL (1999)) behandelt. Bei den Parametern handelt es sich beispielsweise um Frequenzfaktoren, Aktivierungsenergien oder Reaktionsenthalpien der einzelnen Reaktionen. Alle auftretenden Zuläufe sowie die Innentemperatur des Reaktors werden durch Steuerfunktionen, wie in Abschnitt 3.1.3 beschrieben, parametrisiert. Die auftretenden Steuergrößen sind die ursprünglichen Steuergrößen (z.B. Anfangsstoffmengen oder Anfangstemperaturen) und die Variablen aus der Parametrisierung der Steuerfunktionen. Je feiner die Parametrisierung der Steuerfunktionen gewählt wurde, desto mehr Steuergrößen treten auf.

Die Notation der algebraischen Gleichungen erfolgt mit den Molzahländerungen Δn der Spezies aus Gleichung (3.23). Dabei gilt:

$$\Delta n := \text{Stoffmenge} - \text{Anfangsstoffmenge} - \text{zugeflossene Stoffmenge}.$$

Bei den ersten zwei betrachteten Beispielen werden die zeitlich variablen Stoffmengen aller auftretenden Spezies durch zwei Plots dargestellt. Der erste Plot ergibt sich aus der Integration der DAE ohne eine Optimierung der Versuchsplanungsvariablen. Dabei

werden die Steuergrößen auf sinnvolle Initialisierungswerte gesetzt. Der zweite Plot ergibt sich nach der Optimierung mit VPLAN98.

Die Laufzeiten der dabei auftretenden Ableitungsroutinen werden in einer Tabelle für jedes Beispiel angegeben. Verglichen wird dabei die Ableitungserzeugung mit Numerischen Differenzen (DQ), ADIFOR, sowie mit dem in dieser Arbeit implementierten Modell- und Ableitungsgenerator MGAD in einer *Dense*-Version und einer *Sparse*-Version (MGAD-S).

Bei der Sparse-Version wird ausgenutzt, daß die Seed-Matrizen der einzelnen Richtungen aus Einheitsmatrizen bestehen. Die Berechnung der Gradientenvektoren erfolgt somit nicht für alle Richtungen, sondern nur für diejenigen Werte, deren korrespondierender Seed-Matrix Eintrag von Null verschieden ist. Somit wird eine große Anzahl von Multiplikationen mit Null vermieden, und die Laufzeit der erzeugten Ableitungen kann sich deutlich verringern. Dies trifft vor allem bei Ableitungen nach Steuergrößen zu, da in der Regel eine Variable auf der rechten Seite einer Zuweisung nur von sehr wenigen Steuergrößen abhängt. Vor allem bei zweiten Ableitungen konnten mit dieser Methode erhebliche Laufzeitverbesserungen erzielt werden.

In jeder Tabelle werden zusätzlich die Laufzeiten der ersten Ableitungen ($\sum_1$), der zweiten Ableitungen ($\sum_2$) sowie aller Ableitungen ($\sum_{1+2}$) aufsummiert. Bei den Summationen der Laufzeiten der von ADIFOR, MGAD und MGAD-S erzeugten Ableitungsroutinen stehen in Klammern noch die Verbesserungsfaktoren im Vergleich mit den durch Differenzenquotienten ermittelten Ableitungen.

5.2 Die Urethan-Reaktion

Bei der Urethan-Reaktion handelt es sich (vgl. BAUER ET AL. (1998a)) um eine Simultan- und eine Konsekutivreaktion mit einem chemischen Gleichgewicht. Die Reaktanden des Reaktionssystems sind Phenylisocyanat (A), Butanol (B), Urethan (C), Allophanat (D) und Isocyanurat (E). Das einzige Lösungsmittel ist Dimethylsulfoxid. Es ist ein Reaktionssystem, das in der chemischen Kinetik sehr häufig vorkommt. Zum Beispiel werden bei der Polyurethanproduktion Diisocyanate und Diole als Edukte eingesetzt. Während der Hauptreaktion zum Urethan entstehen Folgereaktionen zum Allophanat. Die Bildung des Allophanats führt zu nichtlinearen Polymeren, die bei einer sehr hohen Allophanatkonzentration zu Vernetzungen führen können. Bei der Modellierung der verfahrenstechnischen Komponenten wurden viele praktische Aspekte des konkreten Laboralltags mitberücksichtigt, wie beispielsweise zur Verfügung stehende Reagenzien, geeignete Apparaturen oder der Nacht-Tag-Arbeitsrhythmus des Laborpersonals.

Die Reaktion findet in einem Rührkessel mit zwei Zuläufen unter einer Semi-Batch-Fahrweise statt. In der Vorlage befinden sich die Spezies A und B. Im ersten Zulauf befindet sich A, im zweiten B (jeweils gelöst im Lösungsmittel). Die Innentemperatur des Reaktors wird dabei durch eine Steuerfunktion festgelegt. Die acht unbekannten Parameter dieses Modells sind die Frequenzfaktoren $k_{ref_1}, k_{ref_2}, k_{ref_4}$, die Aktivierungsenergien $E_{a_1}, E_{a_2}, E_{a_4}$, die Gleichgewichtskonstante k_{c_2} und die Reaktionsenthalpie ΔH_2. Die Stoffmenge $n_1, \ldots, n_6$ der beteiligten Spezies sowie die Stoffmenge n_6 des Lösungsmittels sind die zeitlich variablen Zustandsvariablen des DAE-Systems.

Das Reaktionsschema lautet:

$$\begin{array}{rcl} A+B & \longrightarrow & C \\ A+C & \rightleftharpoons & D \\ 3A & \longrightarrow & E \end{array} \tag{5.1}$$

Das Aufstellen des DAE-Systems ergibt drei Differentialgleichungen und drei algebraische Gleichungen. Das resultierende DAE-System lautet somit:

$$\begin{aligned} \dot{n}_1 &= -V \cdot (r_1 + r_2 - r_3 + 3r_4) + dn_{e_1} \\ \dot{n}_2 &= -V \cdot r_1 + dn_{e_2} \\ \dot{n}_3 &= V \cdot (r_1 + r_2 + r_3) \\ 0 &= -\Delta n_4 - \Delta n_2 - \Delta n_3 \\ 0 &= -\Delta n_5 - \frac{1}{3}\Delta n_1 + \frac{2}{3}\Delta n_2 + \frac{1}{3}\Delta n_3 \\ 0 &= \Delta n_6 \end{aligned}$$

mit den Zwischengrößen

$$\begin{aligned} dn_{e_1} &= na_{e_{1,1}} \cdot dfeed_1 \\ dn_{e_2} &= na_{e_{2,2}} \cdot dfeed_2 \\ r_1 &= k_1 \cdot \frac{n_1}{V} \cdot \frac{n_2}{V} \\ r_2 &= k_2 \cdot \frac{n_1}{V} \cdot \frac{n_3}{V} \\ r_3 &= k_3 \cdot \frac{n_4}{V} \\ r_4 &= k_4 \cdot \left(\frac{n_1}{V}\right)^{\alpha_{4,1}} \\ k_i &= k_{ref_i} \cdot \exp\left(\frac{E_{a_i}}{R} \cdot \left(\frac{1}{T} - \frac{1}{T_{ref_i}}\right)\right), \quad i = 1,2,4 \\ k_3 &= k_2 \cdot \left(k_{c_2} \cdot \exp\left(\frac{\Delta H_2}{R} \cdot \left(\frac{1}{T} - \frac{1}{T_{g_2}}\right)\right)\right)^{-1} \\ V &= \sum_{j=1}^{6} \frac{n_j \cdot M_j}{\varrho_j} \end{aligned}$$

und den Anfangswerten (die auch Steuergrößen sein können):

$$n_1(t_0) = q_1, \quad n_2(t_0) = q_2, \quad n_3(t_0) = 0, \quad t_0 = 0$$

Bei allen weiteren im Modell auftretenden Größen handelt es sich um Konstanten (Molmassen M_j, Dichten ϱ_j, Gaskonstante R und Referenztemperaturen T_{ref_i}, T_{g_2}).

Die Abbildungen 5.1 und 5.2 der Lösungskurven des DAE-Systems zeigen deutlich die veränderten Einstellungen der Steuergrößen vor und nach der Optimierung, welche sich in den beiden unterschiedlich verlaufenden Kurven der Stoffmengen bemerkbar machen.

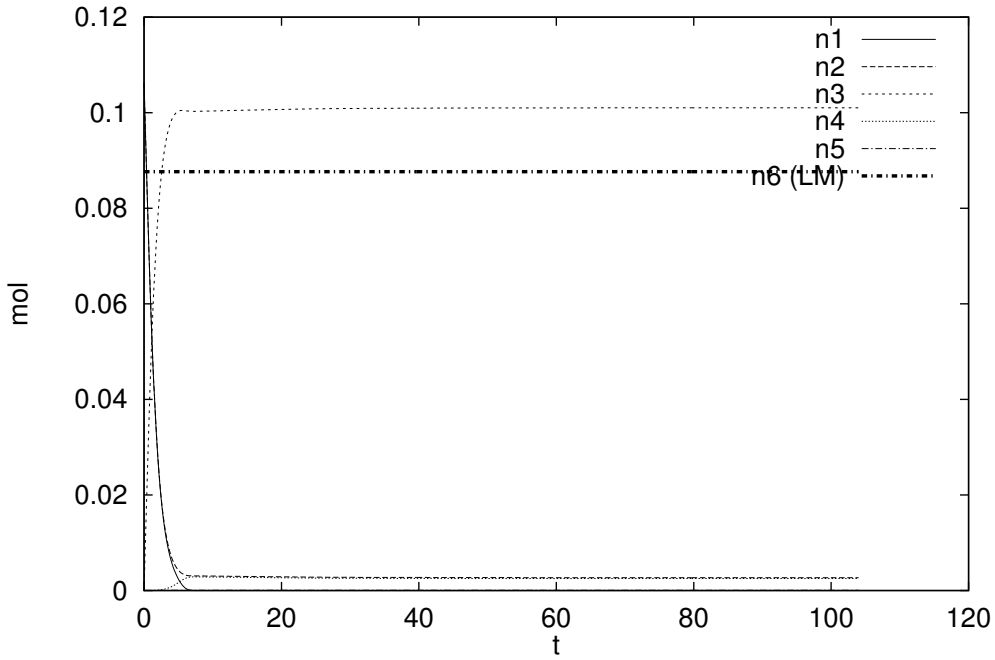


Abbildung 5.1: Trajektorien bei der Urethan-Reaktion vor der Optimierung

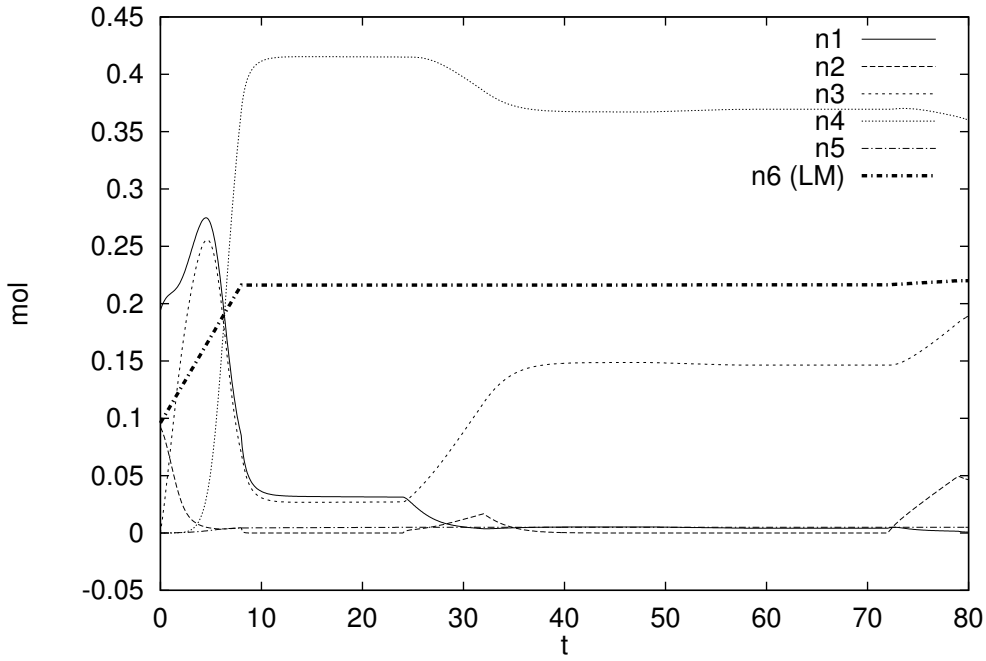


Abbildung 5.2: Trajektorien bei der Urethan-Reaktion nach der Optimierung des Versuchsplans mit dem A-Kriterium

Problemgrößen (Urethan-Reaktion)

Steuergrößen:	75
Parameter:	8
diff. ZV:	3
alg. ZV:	3
lokale Variablen:	76

Routine	Laufzeit	Quelltextgröße
`ffcn`	0.34	201 Zeilen
`mfcn1`	0.30	184 Zeilen

Routine	DQ	ADIFOR	Faktor	MGAD	Fak.	MGAD-S	Fak.
`x_ffcn`	1.43	0.61	2.34	0.75	1.90	0.59	2.42
`p_ffcn`	3.54	1.06	3.34	0.99	3.57	0.55	6.44
`q_ffcn`	22.17	8.46	2.62	8.23	2.69	0.99	22.39
`x_x_ffcn`	6.00	2.19	2.78	3.03	1.98	2.42	2.48
`x_p_ffcn`	13.93	3.65	3.81	2.83	4.92	1.48	9.41
`q_x_ffcn`	115.01	37.21	3.09	45.43	2.53	2.47	46.56
`q_p_ffcn`	269.80	79.93	3.38	98.76	2.73	2.15	125.49
`x_mfcn1`	1.33	0.39	3.41	0.36	3.69	0.32	4.16
`p_mfcn1`	3.05	0.33	9.24	0.39	7.82	0.30	10.17
`q_mfcn1`	26.12	2.95	8.85	2.63	9.93	0.39	66.97
`x_x_mfcn1`	5.48	0.56	9.79	0.42	13.05	0.35	15.66
`x_p_mfcn1`	12.60	0.34	37.06	0.46	27.39	0.33	38.18
`q_x_mfcn1`	104.94	4.16	25.23	1.63	64.38	0.47	223.28
`q_p_mfcn1`	236.54	0.60	394.23	1.11	213.10	0.42	563.19
$\sum_1$	57.64	13.80 (4.18)		13.35 (4.32)		3.14 (13.36)	
$\sum_2$	764.30	128.64 (5.94)		153.67 (4.98)		10.09 (75.75)	
$\sum_{1+2}$	8821.94	142.44 (5.77)		167.02 (4.92)		13.23 (62.13)	

Tabelle 5.1: Laufzeiten der Ableitungen bei der Urethan-Reaktion

5.3 Die Phosphin-Reaktion

Bei der Phosphin-Reaktion handelt es sich um ein Reaktionssystem mit einer Reaktion in einem semi-batch Reaktor und einem Zulauf. Die Reaktanden der Reaktion sind Phosphin (A), Halogenid (B) und Phosphoniumsalz (C). In der Vorlage befindet sich Spezies A, im Zulauf Spezies B; jeweils gelöst im Lösungsmittel. Da die Reaktion stark exotherm abläuft, ist eine isotherme Reaktionsführung nicht oder nur sehr schwer möglich. Deshalb wird eine zusätzliche Differentialgleichung für die Temperatur der Reaktionsmasse (siehe Abschnitt 3.1.4) berücksichtigt. Die vier unbekannten Parameter in diesem Modell sind die beiden Reaktionsordnungen $\alpha_{1\,1}$ und $\alpha_{1\,2}$ sowie die kinetischen Parameter Aktivierungsenergie E_{a_1} und der Frequenzfaktor k_{ref_1}. Die Anfangsstoffmenge $na_1(t_0)$ der Spezies A sowie die Anfangstemperatur $T(t_0)$ des Reaktors werden als Steuergrößen modelliert. Die restlichen Steuergrößen ergeben sich aus der Parametrisierung der Steuerfunktionen für den Zulauf und für die Temperatur.

Das Reaktionsschema lautet:

$$A + B \longrightarrow C \tag{5.2}$$

Mit den bekannten Größen aus Tabelle 3.3 ergibt sich das resultierende DAE-System zu:

$$
\begin{aligned}
\dot{n}_1 &= -V \cdot r_1 \\
\dot{T} &= \frac{w_1 + w_2 + w_3}{M_{ges} \cdot c_p} \\
0 &= -\Delta n_1 + \Delta n_2 \\
0 &= \Delta n_1 + \Delta n_3 \\
0 &= \Delta n_4
\end{aligned}
$$

mit

$$
\begin{aligned}
w_1 &:= V \cdot r_1 \cdot \Delta H_1 \\
w_2 &:= dfeed_1 \cdot (na_{e_{1,2}} \cdot M_2 + na_{e_{1,4}} \cdot M_4) \cdot c_p \cdot (T_{e_1} - T) \\
w_3 &:= k_d \cdot A_w \cdot (T_w - T) \\
r_1 &= k_1 \cdot \left(\frac{n_1}{V}\right)^{\alpha_{1\,1}} \cdot \left(\frac{n_2}{V}\right)^{\alpha_{1\,2}} \\
k_1 &= k_{ref_1} \cdot \exp\left(\frac{E_{a_1}}{R} \cdot \left(\frac{1}{T} - \frac{1}{T_{ref_1}}\right)\right) \\
V &= \sum_{j=1}^{4} \frac{n_j \cdot M_j}{\varrho_i}, \quad M_{ges} = \sum_{j=1}^{4} n_j \cdot M_j
\end{aligned}
$$

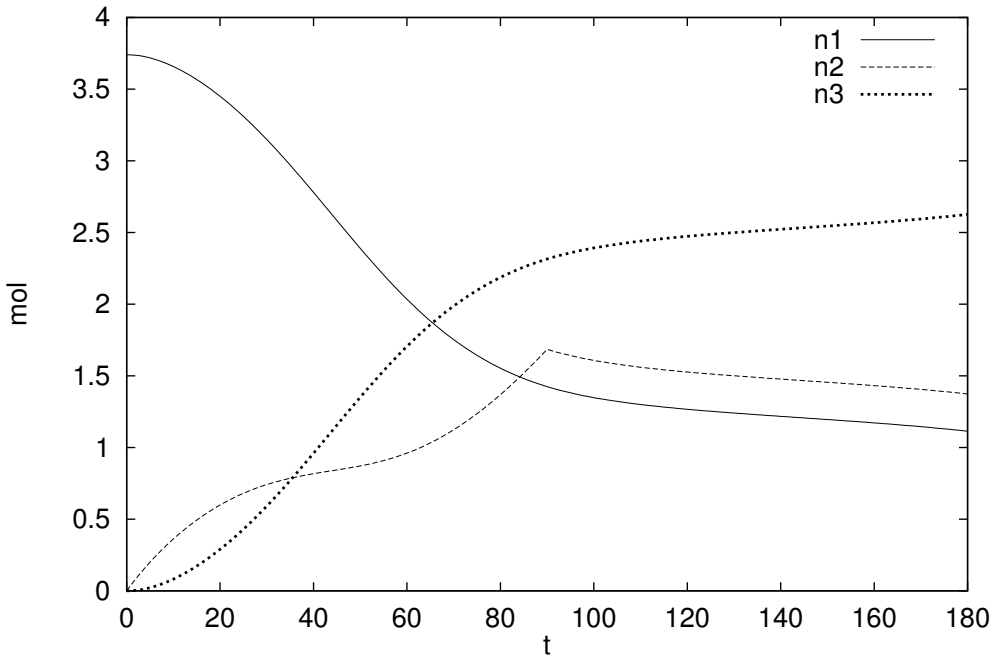


Abbildung 5.3: Trajektorien bei der Phosphin-Reaktion vor der Optimierung

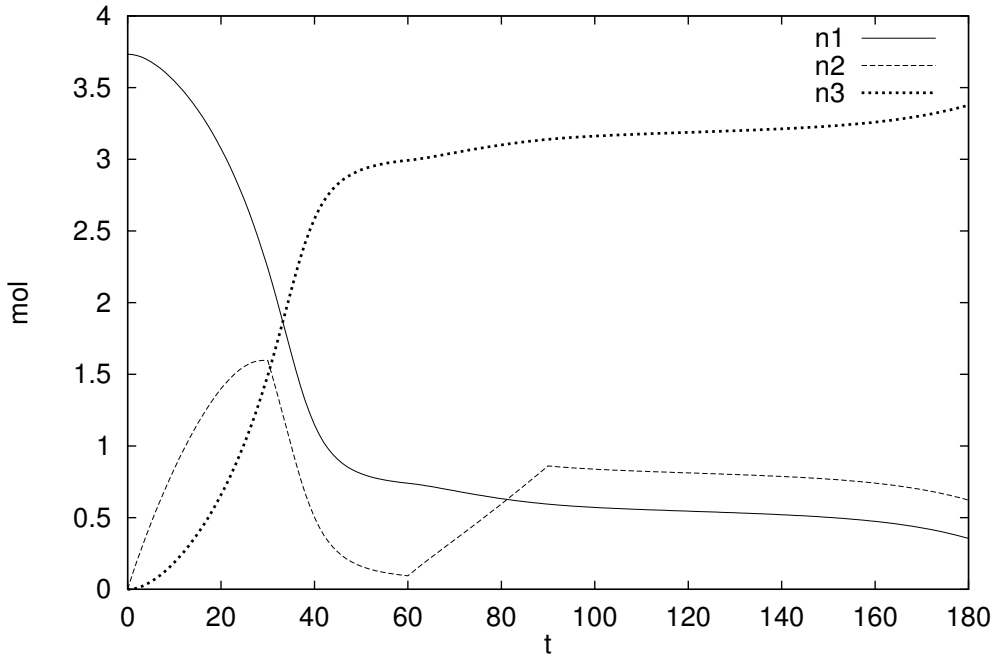


Abbildung 5.4: Trajektorien bei der Phosphin-Reaktion nach der Optimierung des Versuchsplans mit dem E-Kriterium

Routine	DQ	ADIFOR	Faktor	MGAD	Fak.	MGAD-S	Fak.
`x_ffcn`	0.60	0.40	1.50	0.69	0.87	0.65	0.92
`p_ffcn`	1.33	0.56	2.38	1.52	0.88	1.48	0.90
`q_ffcn`	4.65	1.39	3.35	1.60	2.91	0.82	5.67
`x_x_ffcn`	1.25	0.80	1.56	1.13	1.12	0.98	1.28
`x_p_ffcn`	2.74	1.05	2.61	2.71	1.01	2.60	1.05
`q_x_ffcn`	8.63	3.42	2.52	3.07	2.81	1.35	6.39
`q_p_ffcn`	19.57	5.11	3.83	9.95	1.97	3.84	5.10
`x_mfcn1`	0.62	0.34	1.82	0.39	1.59	0.36	1.72
`p_mfcn1`	1.39	0.32	4.34	0.37	3.76	0.35	3.97
`q_mfcn1`	4.78	0.95	5.03	1.10	4.35	0.44	10.86
`x_x_mfcn1`	1.31	0.45	2.91	0.45	2.91	0.35	3.74
`x_p_mfcn1`	2.85	0.32	8.91	0.36	7.92	0.30	9.50
`q_x_mfcn1`	8.92	1.25	7.14	1.37	6.53	0.49	18.20
`q_p_mfcn1`	20.16	0.53	38.04	0.88	22.91	0.43	41.14
$\sum_1$	13.57	3.56 (3.76)		5.67 (2.36)		4.10 (3.26)	
$\sum_2$	65.43	12.93 (5.06)		19.92 (3.28)		10.34 (6.32)	
$\sum_{1+2}$	78.80	16.89 (4.67)		25.59 (3.08)		14.44 (5.45)	

Tabelle 5.2: Laufzeiten der Ableitungen bei der Phosphin-Reaktion

Problemgrößen (Phosphin-Reaktion)

Steuergrößen:	15
Parameter:	4
Diff. ZV:	1
Alg. ZV:	3
lokale Variablen:	53

Routine	Laufzeit	Quelltextgröße
`ffcn`	0.28	180 Zeilen
`mfcn1`	0.29	155 Zeilen

5.4 Phasentransferkatalyse

Hierbei handelt es sich um ein Beispiel von Dr. A. Kud aus der Literatur mit vorgeschalteten schnellen chemischen Gleichgewichten und Lösungsreaktionen im Reaktor. Das System besteht aus den 20 Spezies A–T, sechs einfachen Reaktionen und neun Gleichgewichtsreaktionen. Das Reaktionsschema ist gegeben durch:

$$
\begin{array}{lcc}
1) & A & \rightleftharpoons & B + C \\
2) & C + D & \rightleftharpoons & E + F \\
3) & D & \rightleftharpoons & G + F \\
4) & H & \rightleftharpoons & B + F \\
5) & I + C & \longrightarrow & J + K \\
6) & I + F & \longrightarrow & L + K \\
7) & I + L & \longrightarrow & M + G + K \\
8) & J + F & \longrightarrow & L + C \\
9) & I + J + C & \longrightarrow & N + K \\
10) & O + D & \longrightarrow & P + 2G + 2K \\
11) & P + D & \rightleftharpoons & Q \\
12) & Q & \rightleftharpoons & R + G \\
13) & R & \rightleftharpoons & S + G \\
14) & L & \rightleftharpoons & T + G \\
15) & C + D & \rightleftharpoons & E + F
\end{array}
\tag{5.3}
$$

Das mit Hilfe des Modellgenerators bestimmte DAE-System wird durch die folgenden Formeln bestimmt.

Die algebraischen Variablen ergeben sich durch:

$$
\begin{aligned}
0 &= \Delta n_{14} + \frac{1}{2}\Delta n_1 + \frac{1}{2}\Delta n_3 + \frac{1}{2}\Delta n_5 + \frac{1}{2}\Delta n_{10} \\
0 &= \Delta n_{15} - na_{15} + \frac{1}{2}\Delta n_9 + \frac{1}{2}\Delta n_{11} \\
0 &= \Delta n_{16} + \Delta n_2 - \Delta n_3 - \Delta n_4 - \Delta n_5 - \Delta n_6 - \Delta n_{11} + \Delta n_{13} \\
0 &= \Delta n_{18} - \Delta n_1 - \Delta n_2 + 2\Delta n_4 + \Delta n_5 + \Delta n_6 + \Delta n_7 + \Delta n_{12} + 2\Delta n_{17} \\
0 &= \Delta n_{19} + \Delta n_1 + \Delta n_3 - \Delta n_4 + \Delta n_7 - \frac{1}{2}\Delta n_9 + \frac{1}{2}\Delta n_{11} - \Delta n_{12} - \Delta n_{13} \\
&\quad -\Delta n_{17} \\
0 &= \Delta n_{20} - \Delta n_1 - \Delta n_3 - \Delta n_5 + \Delta n_9 + \Delta n_{12} + 2\Delta n_{13}
\end{aligned}
$$

Die Differentialgleichungen des DAE-Systems lauten:

$$
\begin{aligned}
\dot{n}_1 &= V \cdot (-r_1 + r_2) \\
\dot{n}_2 &= V \cdot (r_1 - r_2 + r_7 - r_8) \\
\dot{n}_3 &= V \cdot (r_1 - r_2 - r_3 + r_4 - r_9 + r_{12} - r_{13} - r_{23} + r_{24}) \\
\dot{n}_4 &= V \cdot (-r_3 + r_4 - r_5 + r_6 - r_{14} - r_{15} + r_{16} - r_{23} + r_{24}) \\
\dot{n}_5 &= V \cdot (r_3 - r_4 + r_{23} - r_{24}) \\
\dot{n}_6 &= V \cdot (r_3 - r_4 + r_5 - r_6 + r_7 - r_8 - r_{10} - r_{12} + r_{23} - r_{24}) \\
\dot{n}_7 &= V \cdot (r_5 - r_6 + r_{11} + 2 \cdot r_{14} + r_{17} - r_{18} + r_{19} - r_{20} + r_{21} - r_{22}) \\
\dot{n}_9 &= V \cdot (-r_9 - r_{10} - r_{11} + r_{13}) \\
\dot{n}_{10} &= V \cdot (r_9 - r_{12} - r_{13}) \\
\dot{n}_{11} &= V \cdot (r_9 + r_{10} + r_{11} + r_{13} + 2 \cdot r_{14}) \\
\dot{n}_{12} &= V \cdot (r_{10} - r_{11} + r_{12} - r_{21} + r_{22}) \\
\dot{n}_{13} &= V \cdot (r_{11}) \\
\dot{n}_{13} &= V \cdot (r_{15} - r_{16} - r_{17} + r_{18})
\end{aligned}
$$

Die Reaktionsgeschwindigkeiten lauten:

$$
\begin{aligned}
r_1 &= k_1 \cdot \frac{n_1}{V} & r_{13} &= k_{13} \cdot \frac{n_3 \cdot n_9 \cdot n_{10}}{V^3} \\
r_2 &= k_2 \cdot \frac{n_2 \cdot n_3}{V^2} & r_{14} &= k_{14} \cdot \frac{n_4 \cdot n_{15}}{V^2} \\
r_3 &= k_3 \cdot \frac{n_3 \cdot n_4}{V^2} & r_{15} &= k_{15} \cdot \frac{n_4 \cdot n_{16}}{V^2} \\
r_4 &= k_4 \cdot \frac{n_5 \cdot n_6}{V^2} & r_{16} &= k_{16} \cdot \frac{n_{17}}{V} \\
r_5 &= k_5 \cdot \frac{n_4}{V} & r_{17} &= k_{17} \cdot \frac{n_{17}}{V} \\
r_6 &= k_6 \cdot \frac{n_6 \cdot n_7}{V} & r_{18} &= k_{18} \cdot \frac{n_7 \cdot n_{18}}{V^2} \\
r_7 &= k_7 \cdot \frac{n_8}{V} & r_{19} &= k_{19} \cdot \frac{n_{18}}{V} \\
r_8 &= k_8 \cdot \frac{n_2 \cdot n_6}{V^2} & r_{20} &= k_{20} \cdot \frac{n_7 \cdot n_{19}}{V^2} \\
r_9 &= k_9 \cdot \frac{n_3 \cdot n_9}{V^2} & r_{21} &= k_{21} \cdot \frac{n_{12}}{V} \\
r_{10} &= k_{10} \cdot \frac{n_6 \cdot n_9}{V^2} & r_{22} &= k_{22} \cdot \frac{n_7 \cdot n_{20}}{V^2} \\
r_{11} &= k_{11} \cdot \frac{n_9 \cdot n_{12}}{V^2} & r_{23} &= k_{23} \cdot \frac{n_3 \cdot n_4}{V^2} \\
r_{12} &= k_{12} \cdot \frac{n_6 \cdot n_{10}}{V^2} & r_{24} &= k_{24} \cdot \frac{n_5 \cdot n_6}{V^2}
\end{aligned}
$$

mit den korrespondierenden Reaktionsgeschwindigkeitskonstanten und des Volumens:

$$
\begin{aligned}
k_i &= \begin{cases} k_{ref_i} \cdot \exp\left(\dfrac{E_{a_i}}{R} \cdot \left(\dfrac{1}{T} - \dfrac{1}{T_{ref_i}}\right)\right), \\ \qquad \text{falls } i = 1,3,5,7,9,19,11,12,13,14,15,17,19,21,23 \\ k_{i-1} \cdot \left(k_{c_{i-1}} \cdot \exp\left(\dfrac{\Delta H_{i-1}}{R} \cdot \left(\dfrac{1}{T} - \dfrac{1}{T_{g_{i-1}}}\right)\right)\right)^{-1}, \\ \qquad \text{falls } i = 2,4,6,8,16,18,20,22,23 \end{cases} \\
V &= \sum_{j=1}^{20} \frac{n_j \cdot M_j}{\varrho_j}
\end{aligned}
$$

In den Abbildungen 5.5, 5.6 und 5.7 werden die Trajektorien des DAE-Systems graphisch dargestellt. Die einzelnen chemischen Reaktionen werden zu Gruppen zusammengefaßt, und die jeweils beteiligten Spezies werden in einer gemeinsamen Abbildung dargestellt.

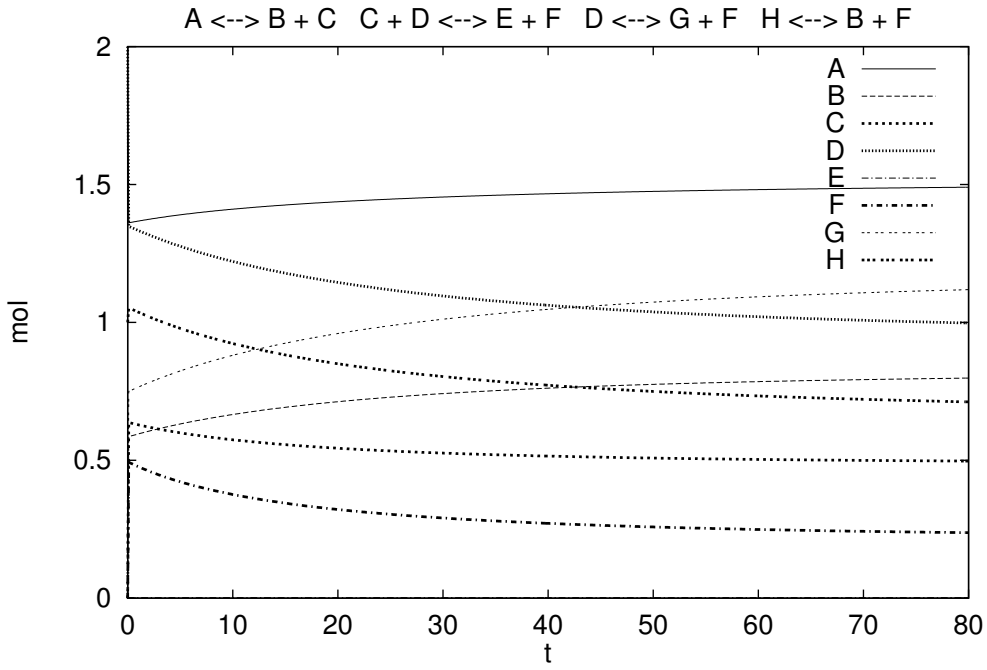


Abbildung 5.5: Trajektorien bei den schnellen Gleichgewichtsreaktionen (Phasentransferkatalyse)

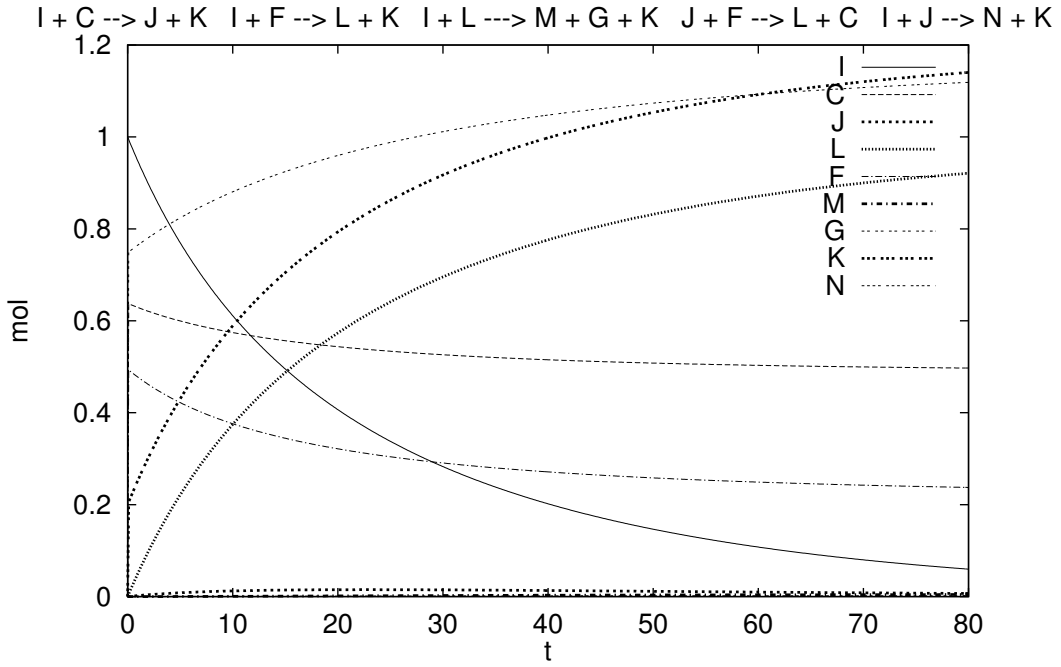


Abbildung 5.6: Trajektorien bei den einfachen Reaktionen (Phasentransferkatalyse)

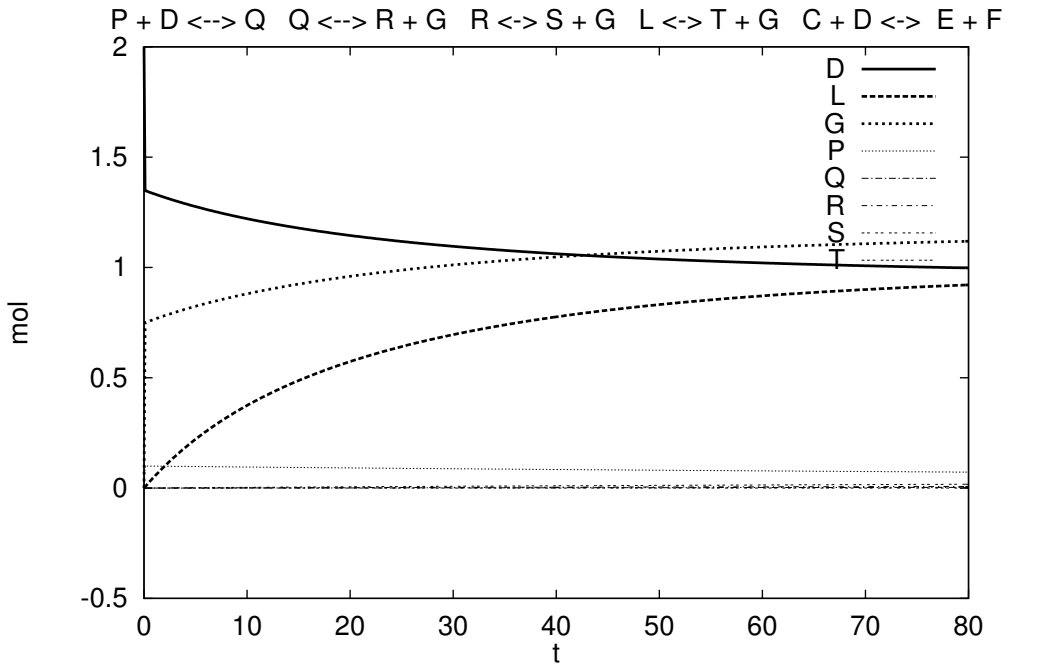


Abbildung 5.7: Trajektorien bei den Reaktionen der Endprodukte (Phasentransferkatalyse)

Problemgrößen (Phasentransferkatalyse)

Steuergrößen:	7
Parameter:	5
Diff. ZV:	13
Alg. ZV:	7
lokale Variablen:	276

Routine	Laufzeit	Quelltextgröße
`ffcn`	1.50	427 Zeilen
`mfcn1`	1.41	392 Zeilen

Routine	DQ	ADIFOR	Faktor	MGAD	Fak.	MGAD-S	Fak.
`x_ffcn`	28.20	8.88	3.18	11.23	2.51	11.05	2.56
`p_ffcn`	11.66	2.76	4.22	2.85	4.09	2.26	5.16
`q_ffcn`	13.05	6.42	2.03	6.91	1.89	6.50	2.01
`x_x_ffcn`	428.33	204.83	2.09	298.37	1.44	206.77	2.07
`x_p_ffcn`	175.49	17.14	10.24	29.63	5.92	8.72	20.13
`q_x_ffcn`	211.72	128.03	1.66	170.26	1.24	134.38	1.58
`q_p_ffcn`	85.15	11.29	7.54	20.61	4.13	10.23	8.32
`x_mfcn1`	26.43	4.25	6.22	4.39	6.02	3.01	8.78
`p_mfcn1`	10.97	1.79	6.13	2.10	5.22	1.98	5.54
`q_mfcn1`	11.73	2.25	5.21	2.51	4.67	2.01	5.84
`x_x_mfcn1`	373.60	8.00	46.70	10.34	36.13	6.99	53.45
`x_p_mfcn1`	156.47	1.99	78.62	4.18	37.43	2.20	71.12
`q_x_mfcn1`	175.51	5.59	31.40	6.53	26.88	3.78	46.43
`q_p_mfcn1`	73.17	1.99	36.77	2.68	27.30	2.03	36.04
$\sum_1$	102.04	26.35 (3.87)		29.99 (3.40)		26.81 (3.90)	
$\sum_2$	1679.44	378.86 (4.43)		542.60 (3.10)		375.10 (4.48)	
$\sum_{1+2}$	1781.48	405.21 (4.40)		572.59 (3.11)		401.91 (4.43)	

Tabelle 5.3: Laufzeiten der Ableitungen bei der Phasentransferkatalyse

5.5 Radikalische Polymerisation

Beim Erhitzen von Ethylen mit Sauerstoff unter Druck entsteht nach MORRISON & BOYD (1978) ein Molekül mit sehr hoher Molmasse (ca. 20 000 $[kg/mol]$), ein Alkan von sehr großer Kettenlänge (Polyethylen). Die Bildung von Polyethylen ist ein Beispiel für eine Polymerisation, d.h. den *Zusammenschluß vieler kleiner Moleküle zu sehr großen Molekülen*, den sogenannten *Polymeren*.

Um eine Radikalische Polymerisation einzuleiten, muß eine kleine Menge eines *Initiators* vorhanden sein. Zu den am häufigsten gebrauchten Initiatoren gehören nach MORRISON & BOYD (1978) Peroxide wie z.B. Wasserstoffperoxid (H_2O_2), deren Wirkung auf dem Zerfall in freie Radikale beruhen. Ein aus dem Peroxid entstehendes Radikal (z.B. $HO\cdot$) lagert sich an ein Alkenmolekül an und erzeugt dabei ein neues Radikal, das wiederum mit einem Alkenmolekül reagiert. Dies führt zu einer Kette von Reaktionen. Als Alken wird im folgenden Ethylen (C_2H_4) verwendet:

$$\left.\begin{array}{lcl} H_2O_2 & \longrightarrow & HO\cdot + HO\cdot \\ HO\cdot + C_2H_4 & \longrightarrow & HO - C_2H_4\cdot \end{array}\right\} \text{ Kettenstart}$$

$$\left.\begin{array}{lcll} HO-C_2H_4\cdot & + & C_2H_4 & \longrightarrow & HO-C_2H_4-C_2H_4\cdot \\ HO-(C_2H_4)-C_2H\cdot & + & C_2H_4 & \longrightarrow & HO-(C_2H_4)_2-C_2H_4\cdot \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ HO-(C_2H_4)_{n-2}-C_2H_4\cdot & + & C_2H_4 & \longrightarrow & HO-(C_2H_4)_{n-1}-C_2H_4\cdot \end{array}\right\} \text{Ketten-schritte}$$

Um eine Polymerisation endlicher Kettenschritte zu modellieren, wird im folgenden angenommen, daß das größte gebildete Radikal am Ende der Polymerisation 20 (C_2H_4)-Moleküle besitzt. Es hat somit die Form $HO-(C_2H_4)_{19}-C_2H_4\cdot$.

Unter Vernachlässigung des Zerfalls von H_2O_2 in Radikale lautet das symbolische Reaktionsschema für den obigen Prozeß:

$$\begin{array}{lcl} A+B & \longrightarrow & C \\ C+B & \longrightarrow & D \\ D+B & \longrightarrow & E \\ E+B & \longrightarrow & F \\ \vdots & \vdots & \vdots \\ S+B & \longrightarrow & T \\ T+B & \longrightarrow & U \\ U+B & \longrightarrow & V \end{array} \tag{5.4}$$

Die Parameter des Modells sind die 20 Frequenzfaktoren k_{ref_i} der einzelnen Gleichungen. Dabei wurden die Startwerte für die Parameter so gewählt, daß die k_{ref_i} mit steigendem Index kleiner werden. Dies entspricht auch der realistischen Situation, daß größere Molekülketten weniger schnell mit anderen Molekülen reagieren. Das Startradikal $HO\cdot$ und das im Überschuß vorhandene Ethylen sind zu Beginn in der Reaktorvorlage. Das Startradikal $HO\cdot$ befindet sich im Zufluß. Der komplette Zufluß dieser Spezies erfolgt an zwei Zeitpunkten (siehe Abbildung 5.8). Bei der Modellierung dieses Prozesses werden Reaktionen mit höheren Ketten sowie Abbruchreaktionen zweier Radikale vernachlässigt.

Mit den benötigten Zwischenwerten

$$\begin{aligned} r_1 &= k_1 \cdot \left(\frac{n_1}{V}\right) \cdot \left(\frac{n_2}{V}\right) \\ r_i &= k_i \cdot \left(\frac{n_2}{V}\right) \cdot \left(\frac{n_{i+1}}{V}\right), \quad i = 2,\ldots,20 \\ k_i &= k_{ref_i} \cdot \exp\left(\frac{E_{a_i}}{R} \cdot \left(\frac{1}{T} - \frac{1}{T_{ref_i}}\right)\right), \quad i = 1,\ldots,20 \\ V &= \sum_{j=1}^{23} \frac{n_j \cdot M_j}{\varrho_j} \end{aligned}$$

ergibt das Aufstellen des DAE-Systems 20 Differentialgleichungen und drei algebraische Gleichungen:

$$
\begin{aligned}
\dot{n}_1 &= -V \cdot r_1 + dn_{e_1} \\
\dot{n}_2 &= -V \cdot \sum_{i=1}^{20} r_i + dn_{e_2} \\
\dot{n}_3 &= V \cdot (r_1 - r_2) \\
\dot{n}_4 &= V \cdot (r_2 - r_3) \\
\vdots &\quad \vdots \qquad\qquad \vdots \\
\dot{n}_{19} &= V \cdot (r_{17} - r_{18}) \\
\dot{n}_{20} &= V \cdot (r_{18} - r_{19}) \\
0 &= \Delta n_{21} + 20\Delta n_1 - \Delta n_2 + \sum_{i=3}^{20} (22 - i) \cdot \Delta n_i \\
0 &= \Delta n_{22} + 19\Delta n_1 - \Delta n_2 - \sum_{i=3}^{20} (21 - i) \cdot \Delta n_i \\
0 &= \Delta n_{23}
\end{aligned}
$$

Die Zugabe des Startradikals an genau zwei Zeitpunkten läßt sich anhand der Abbildungen 5.8 bis 5.11 der Trajektorien deutlich erkennen. Man sieht, daß zwei fast identische Phasenverläufe auftreten. An den beiden Zeitpunkten, an denen die Zuläufe erfolgen, reagieren die entstehenden Radikale mit dem Ethylen, bis das Endprodukt mit Stoffmenge n_{22} entsteht. Die Abbildungen lassen auch die unterschiedlichen Reaktionsgeschwindigkeiten der einzelnen Reaktionen erkennen. Je größer das entstehende Radikal, desto kleiner die jeweilige Reaktionsgeschwindigkeit.

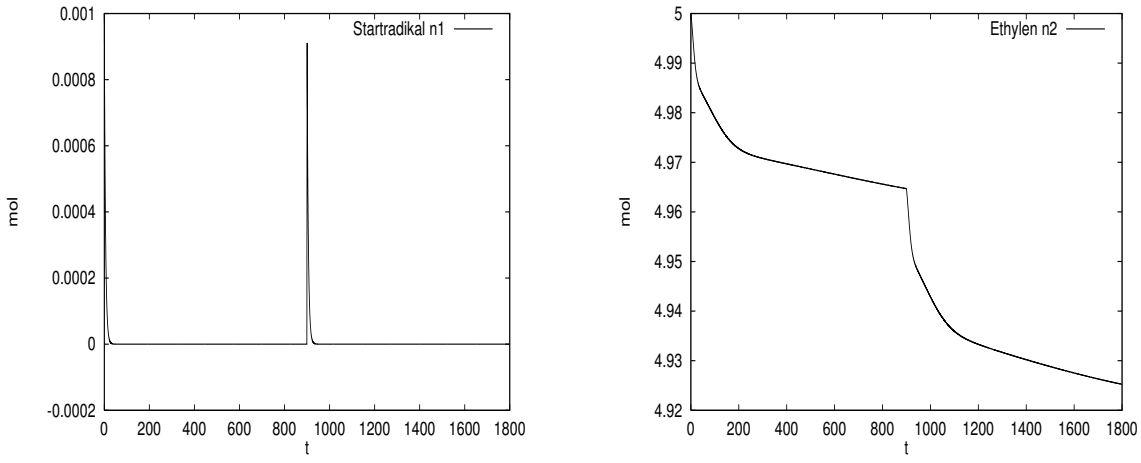


Abbildung 5.8: Trajektorien von Startradikal und Ethylen bei Polymerisation

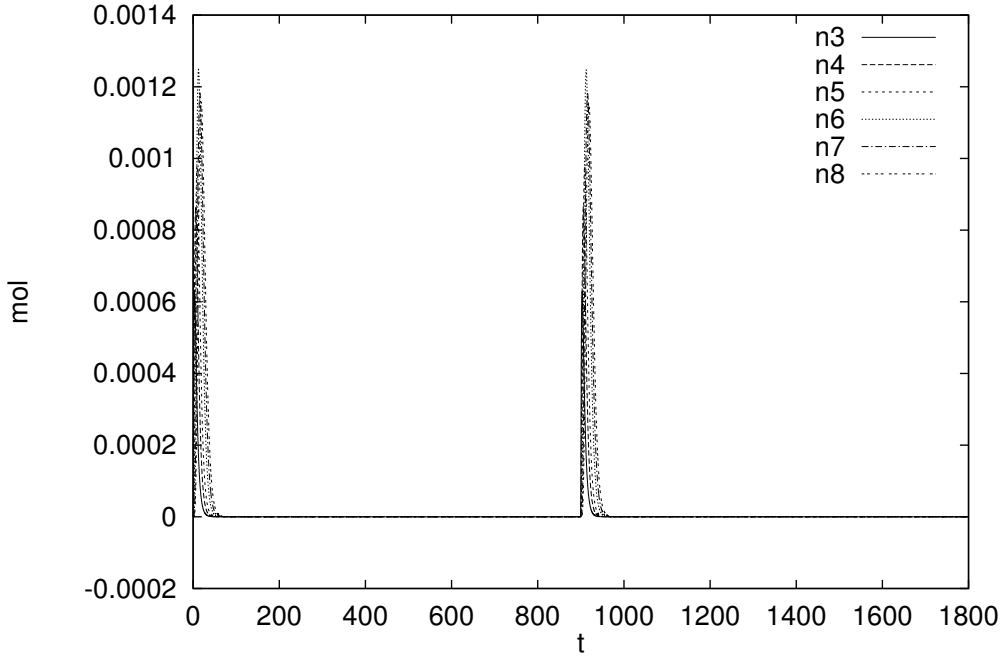


Abbildung 5.9: Trajektorien der Radikale 1 bis 6 bei Polymerisation

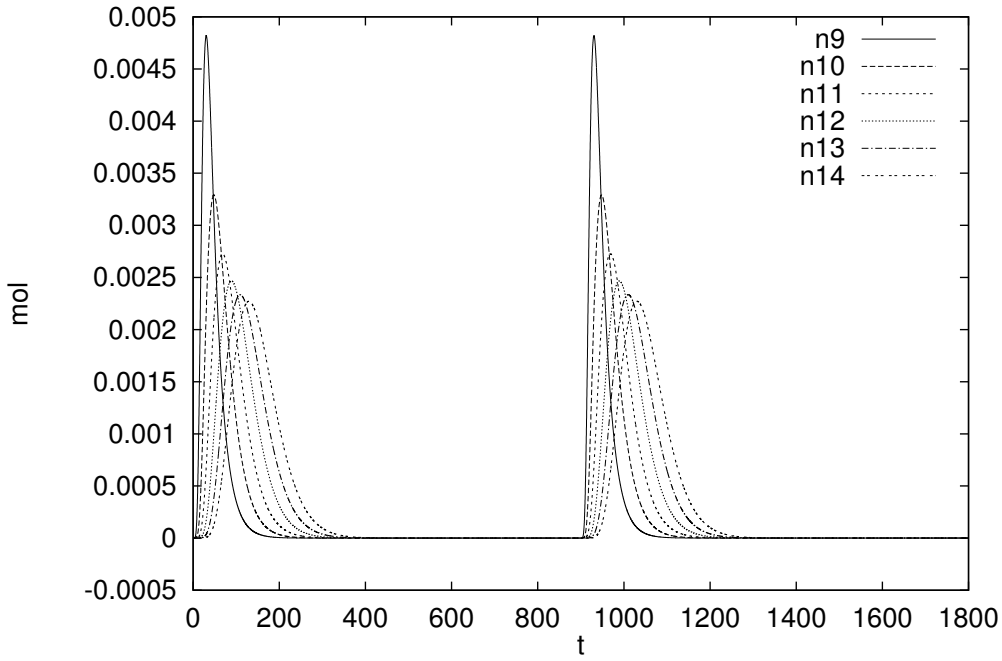


Abbildung 5.10: Trajektorien der Radikale 7 bis 12 bei Polymerisation

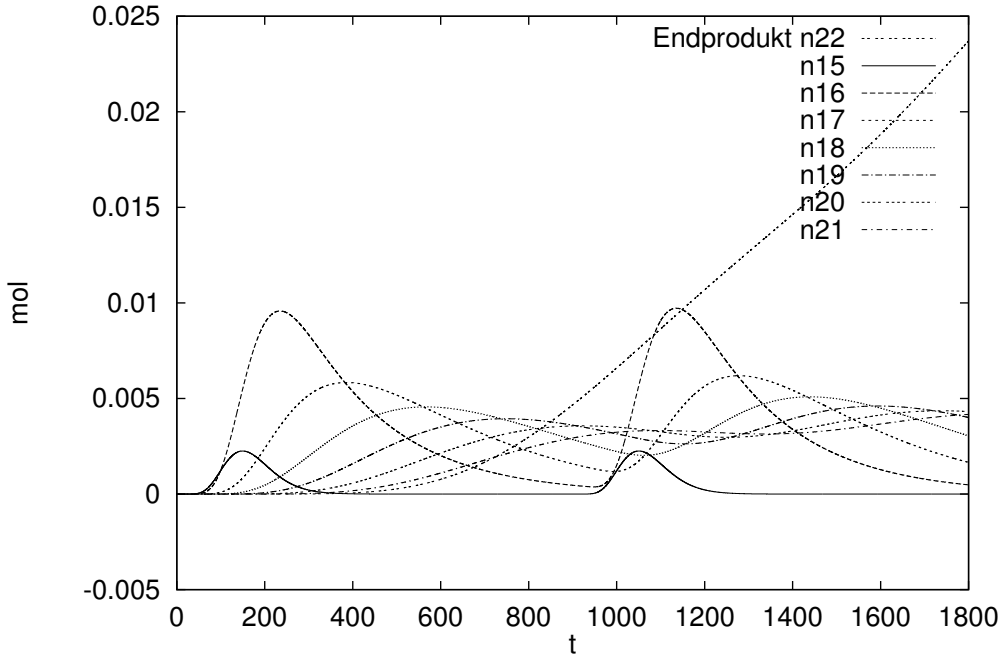


Abbildung 5.11: Trajektorien der Radikale 13 bis 22 bei Polymerisation

Routine	DQ	ADIFOR	Faktor	MGAD	Fak.	MGAD-S	Fak.
`x_ffcn`	36.32	13.53	2.68	16.18	2.24	15.99	2.27
`p_ffcn`	35.61	10.36	3.43	11.20	3.27	3.22	11.06
`q_ffcn`	51.74	21.86	2.37	22.89	2.26	11.60	4.46
`x_x_ffcn`	881.76	447.91	1.97	653.22	1.35	620.29	1.42
`x_p_ffcn`	900.44	237.60	2.01	303.17	1.38	93.05	9.68
`q_x_ffcn`	1360.02	750.44	1.81	934.36	1.46	621.77	2.19
`q_p_ffcn`	1362.83	796.72	1.71	640.66	2.13	87.63	15.55
`x_mfcn1`	30.51	6.13	4.98	6.15	4.96	4.40	6.93
`p_mfcn1`	30.42	1.32	23.06	2.91	10.45	1.42	21.42
`q_mfcn1`	43.50	3.53	12.32	4.15	10.48	2.29	19.00
`x_x_mfcn1`	652.43	15.61	41.80	17.37	37.56	11.49	56.78
`x_p_mfcn1`	644.62	2.50	257.85	3.48	185.24	2.53	254.79
`q_x_mfcn1`	1020.67	14.65	69.67	12.94	78.88	9.48	107.67
`q_p_mfcn1`	1008.48	3.33	302.85	5.38	187.45	2.24	450.21
$\sum_1$	228.10	56.73 (4.02)		63.48 (3.59)		38.92 (5.86)	
$\sum_2$	7831.25	2268.76 (3.45)		2570.58 (3.05)		1448.48 (5.41)	
$\sum_{1+2}$	8059.35	2325.49 (3.47)		2634.06 (3.06)		1487.40 (5.42)	

Tabelle 5.4: Laufzeiten der Ableitungen bei der Polymerisation

Problemgrößen (Polymerisation)

Steuergrößen:	32
Parameter:	20
Diff. ZV:	20
Alg. ZV:	3
lokale Variablen:	288

Routine	Laufzeit	Quelltextgröße
`ffcn`	1.56	476 Zeilen
`mfcn1`	1.52	429 Zeilen

5.6 Zusammenfassung der Ergebnisse

Die Gesamtlaufzeiten aller Beispiele waren bei allen getesteten Ableitungsroutinen bei Numerischen Differenzen (DQ) am höchsten. Bei allen drei Verfahren des AD (ADIFOR, MGAD, MGAD-S) konnten somit deutliche Laufzeitverbesserungen gegenüber dem DQ erzielt werden. Beim Polymerisations-Beispiel sind die Gesamtlaufzeiten aller Ableitungen z.B. 8059.35 s (DQ), 2634.06 s (MGAD), 2325.49 s (ADIFOR) und 1487.40 s (MGAD-S). Die Gesamtlaufzeiten der einzelnen Ableitungsroutinen sind dabei stark von den jeweiligen Beispielen abhängig. So konnte bei MGAD-S (Urethan-Beispiel) ein Faktor von 62.13 und beim Phasentransferkatalyse-Beispiel nur ein Faktor von 4.43 gegenüber dem Numerischen Differenzieren erzielt werden. Dies liegt dabei an der jeweils unterschiedlichen Anzahl der Steuergrößen (75 im Vergleich zu 9).

Das Abschneiden des DQ bei den Meßfunktionen (`mfcn1`) im Vergleich mit den AD-Methoden läßt sich dadurch erklären, daß die rechte Seite einer Meßfunktion nur von sehr wenigen Variablen abhängt und somit eine Reihe von berechneten Zuweisungen nicht bei der Auswertung der rechten Seite eingehen. Da hier nach Konstruktion in den Meßfunktion alle möglichen Zwischenvariablen berechnet werden, ist die Laufzeit einer Meßfunktions-Routine im Verhältnis zu der durch AD generierten korrespondierenden Ableitung recht hoch. Da der Aufwand zur Berechnung einer Ableitung durch DQ proportional zum Aufwand der jeweiligen Funktion (unabhängig von der Struktur der Funktion) ist, ergibt sich hierbei ein deutlicher Vorteil für die durch AD erzeugten Ableitungsroutinen.

Bei ersten und zweiten Ableitungen nach den Zustandsvariablen `x` ist die Laufzeitverbesserung von MGAD-S im Vergleich zu ADIFOR wenig signifikant (2.19 bzw. 2.42 bei der Routine `x_x_ffcn` beim Urethan-Beispiel). Deutlich läßt sich aber eine Geschwindigkeitssteigerung im Sparse-Modus gegenüber den von ADIFOR erzeugten Ableitungen nach p und q erkennen. Die Laufzeitverbesserung steigt dabei mit steigender Anzahl der Richtungen, nach denen abgeleitet wird, linear an. Deutlich läßt sich dies bei der Ableitungsroutine `q_x_ffcn` (Urethan-Beispiel) sehen. Dort benötigt die von MGAD-S erzeugte Routine im Schnitt 2.47 s und die von ADIFOR erzeugte Ableitungsroutine 37.21 s. Graphisch stellt diese Tatsache die Abbildung 5.12 dar. Berechnet wird dabei der Quotient aus der Laufzeit von ADIFOR und der Laufzeit von MGAD-S in Abhängigkeit der Anzahl der Ableitungsrichtungen $q\_d\_$.

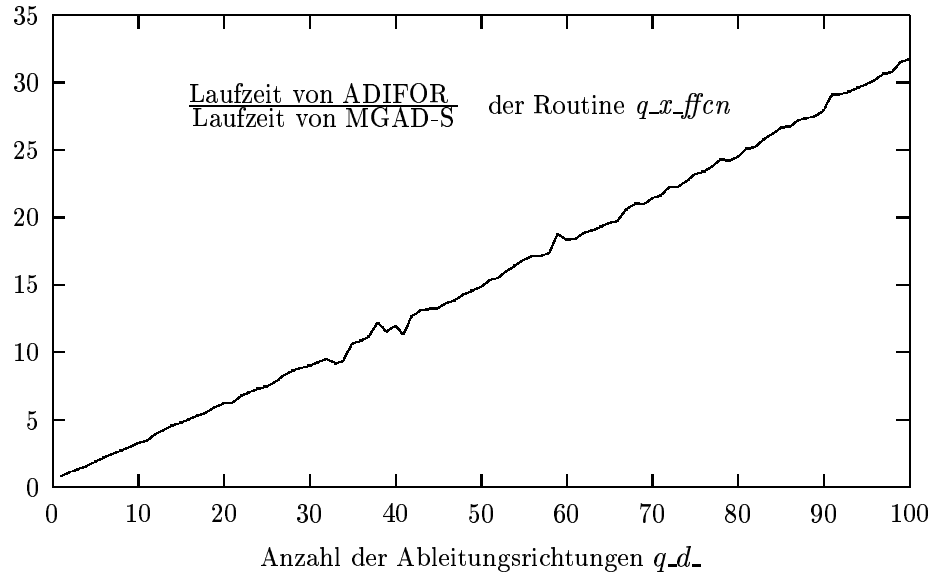


Abbildung 5.12: Laufzeitverbesserung in Abhängigkeit der Anzahl der Ableitungsrichtungen

Kapitel 6

Resümee und Ausblick

Eine bescheidene Wahrheit zu finden ist wichtiger als
über die erhabensten Dinge weitschweifig zu diskutieren,
ohne jemals zu einer Wahrheit zu gelangen.

— Galileo Galilei

In der vorliegenden Diplomarbeit wurde der Einsatz von Verfahren des automatischen Differenzierens bei Optimierungsproblemen aus der chemischen Industrie untersucht. Es wurde ein Modellgenerator für chemische Reaktionssysteme in einem Rührreaktor entwickelt, um die Modellgleichungen, die das chemische System mathematisch beschreiben, automatisch zu erstellen. Mit der implementierten Benutzeroberfläche lassen sich Modellgleichungen in der Sprache Fortran77 durch den Benutzer spezifizieren, ohne selbst Quelltexte zu editieren. Dabei werden gleichzeitig die Modellgleichungen und die ersten und zweiten Ableitungen effizient, strukturausnutzend und automatisch erzeugt. Die Arbeit zeigt darüberhinaus, daß das herkömmliche Verfahren zur Ableitungserzeugung von Computerprogrammen, das Numerische Differenzieren, aufgrund der auftretenden numerischen Ungenauigkeiten nicht für die Berechnung von höheren Ableitungen bei den hier behandelten Funktionen geeignet ist. Vor allem bei der Verwendung von SQP-Lösern und der Integration von Differentialgleichungssystemen durch adaptive Schemata wie z.B. dem Integrator DAESOL werden exakte Ableitungen benötigt.

Mit dem untersuchten Software-Paket ADIFOR lassen sich Ableitungen analytisch korrekt und, im Vergleich zu Numerischen Differenzen, mit in der Regel geringeren Laufzeiten ermitteln. Da dieses Paket keine strukturellen Informationen über die abzuleitenden Funktionen kennt, können auftretende Strukturen bei der Erstellung nicht ausgenutzt werden. In dieser Arbeit werden die Modellgleichungen und deren Ableitungen in einem Schritt erstellt. Deshalb können alle strukturellen Informationen des Modells ausgenutzt werden, und es kann ein in puncto Laufzeit optimaler Ableitungs-Quelltext erzeugt werden.

Die prinzipielle Vorgehensweise zur automatischen Ableitungserzeugung läßt sich im weiteren auch auf andere Problemklassen, wie z.B. mehrphasige Reaktionssysteme, oder auch auf mechanische Systeme anwenden. Der programmiertechnische Aufwand kann dabei, abhängig vom konkreten Modell, sehr hoch sein, da vor allem multiplikative Terme für zweite Ableitungen sehr schnell sehr groß werden können.

Abschließend läßt sich vermerken, daß für die Bestimmung der Ableitungen der Modellgleichungen aus Kapitel 3 das im Rahmen dieser Arbeit entwickelte Programm als sehr effizient im Hinblick auf Genauigkeit und Laufzeit angesehen werden kann.

Anhang A

Quelltexte

A.1 Automatisch Erzeugte Fortran-Routinen

In den folgenden Abschnitten werden exemplarisch Fortran77-Quelltexte der Modellgleichungen und deren Ableitungen aufgeführt, die im Rahmen der Diplomarbeit erzeugt wurden. Bei den Beispielen handelt es sich um Reaktionssysteme, die in Kapitel 4.1 beschriebenen werden. Anschließend wird der Quelltext für die Ableitungserzeugung der beschriebenen Gleichungen mit Numerischen Differenzen angegeben.

A.1.1 Automatisch erzeugte Ableitungs-Routine x_ffcn

```
c     ------------------------------------------------------------------
c     Automatisch erzeugtes Fortran77 File 'x_ffcn.f'
c     durch Java Modellgenerator
c     Gerd Ruecker (IWR der Universitaet Heidelberg)
c     Mon Nov 08 06:38:28 GMT+03:30 1999

c     Phosphin-Reaktion (ODE)

c     # Reaktionen:               1
c     # Spezies:                  3
c     # Loesungsmittel:           1
c
c     # Differentialgleichungen:  2
c     # Algebraische Gleichungen: 3
c
c     # Quelltext Zeilen:         282
c     # chemische Variablen       75
c     ------------------------------------------------------------------


      subroutine x_ffcn( x_d_, t, x, x_x, ldx_x, f, x_f, ldx_f, p, q, rw
     &h, iwh, iflag )


         implicit none


c     DEKLARATIONEN


         integer*4 maxDir
         parameter( maxDir = 100 )
```

```
      real*8 t, x(*), f(*), p(*), q(*), rwh(*)
      integer*4 iwh(*), iflag

c     ABLEITUNGEN

      integer*4 x_i_
      integer*4 x_d_,ldx_x, ldx_f
      integer*4 ind, ind_1, ind1(1), ind2(1)
      real*8 x_x(ldx_x, *), x_f(ldx_f, *)

      real*8 tref1, kref1, ea1, alpha1_1, alpha1_2, tmpval, tmp1(4),
     &tmpval1, tmpval2, tmpval3, tmpval4, tmpval5, tmp(7), tmp2(4), tmp3
     &(4), tmp4(4), tmp5(4), tmp6(4), tmp7(4,4), tmp8(4), tmp10(4), log_
     &t(4), d_log_t(4), log_t1(4), log_t2(4), log_t3(4), log_t4(4), log_
     &t5(4), block(5), temp0_e1, kd, d, vw, cp, cww, rhow, height, aw, t
     &w, w_reak, w_feed, w_cool, rgas, pi, dh1, mm1, rho1, delta_n1, mm2
     &, rho2, mm3, rho3, mm4, rho4, na1, na4, na2e1, n2e1, na4e1, n4e1,
     &feed_1, d_feed_1, d_n1e, dme1, temp, temp0, tin, n1, n2, n3, n4, r
     &1, k1, v, m

c     DEKLARATIONEN FUER DIE ERSTEN ABLEITUNGEN

      real*8 x_height(maxDir), x_aw(maxDir), x_tw(maxDir), x_delta_n1
     &(maxDir), x_temp(maxDir), x_n1(maxDir), x_n2(maxDir), x_n3(maxDir)
     &, x_r1(maxDir), x_k1(maxDir), x_v(maxDir), x_m(maxDir)

c     ZUSTANDSVARIABLEN

      temp = x(1)
      do x_i_ = 1, x_d_
         x_temp(x_i_) = x_x(x_i_,1)
      enddo
      n1 = x(2)
      do x_i_ = 1, x_d_
         x_n1(x_i_) = x_x(x_i_,2)
      enddo

c     SKALIERTE PARAMETER

      alpha1_1 = p(1) * 1.0d+0
      alpha1_2 = p(2) * 1.0d+0
      kref1 = p(4) * 1.0d+0
      ea1 = p(3) * 100000.0d+0

c     STEUERGROESSEN

      temp0 = q(2)
      na1 = q(1)
      na4 = q(3)

c     DISKRETISIERUNG DER STEUERFUNKTIONEN

c     AUSSENTEMPERATUR DES REAKTORS
```

```
      if ( iwh(3) .eq. 0 .or. iwh(3) .eq. 1 ) then
         ind =  iwh(2) + iwh(4) - 1
         ind_1 = 0
         tin = q(ind)
      else if ( iwh(3) .eq. 2 .or. iwh(3) .eq. 3 ) then
         ind =  iwh(2) + 2*iwh(4) - 2
         ind_1 = ind + 1
         tin = q(ind) + ( t-rwh(2) ) * q(ind_1)
      endif

c     EINTRAEGE AM REAKTOR

      if ( iwh(6) .eq. 0 .or. iwh(6) .eq. 1 ) then
         ind1(1) =  iwh(5) + iwh(7) - 1
         ind2(1) = 0
         feed_1 = q(ind1(1))
         d_feed_1 = 0.0d+0
      else if ( iwh(6) .eq. 2 .or. iwh(6) .eq. 3 ) then
         ind1(1) =  iwh(5) + 2*iwh(7) - 2
         ind2(1) = ind1(1) + 1
         feed_1 = q(ind1(1)) + ( t-rwh(3) ) * q(ind2(1))
         d_feed_1 = q(ind2(1))
      endif

c     KONSTANTEN

      tref1 = 365.16d+0
      dh1 = 108000.0d+0
      mm1 = 261.32d+0
      rho1 = 1000.0d+0
      mm2 = 122.6d+0
      rho2 = 1000.0d+0
      mm3 = 383.92d+0
      rho3 = 1000.0d+0
      mm4 = 100.0d+0
      rho4 = 1000.0d+0
      na2e1 = 4.0d+0
      na4e1 = 8.0d+0
      cp = 2.0d+0
      temp0_e1 = 293.16d+0
      kd = 1600.0d+0
      d = 0.15d+0
      vw = 1.0d+0
      cww = 0.12d+0
      rhow = 900.0d+0
      rgas = 8.314d+0
      pi = 3.141592653589793d+0

c     HILFSGROESSEN

      n2e1 = na2e1 * feed_1
      n4e1 = na4e1 * feed_1
      dme1 = d_feed_1 * ( na2e1*mm2 + na4e1*mm4 )

c     MOLZAHLAENDERUNG

      delta_n1 = n1 - na1
      do x_i_ = 1, x_d_
         x_delta_n1(x_i_) = x_n1(x_i_)
      enddo
```

```
c     ALGEBRAISCHE GLEICHUNGEN


         n2 =  + n2e1 + delta_n1
         do x_i_ = 1, x_d_
            x_n2(x_i_) =  + x_delta_n1(x_i_)
         enddo
         n3 =  - delta_n1
         do x_i_ = 1, x_d_
            x_n3(x_i_) =  - x_delta_n1(x_i_)
         enddo
         n4 = na4 + n4e1


c     GESAMTVOLUMEN IM REAKTOR


         tmp1(1) = mm1 / rho1
         tmp1(2) = mm2 / rho2
         tmp1(3) = mm3 / rho3
         tmp1(4) = mm4 / rho4
         v =  + n1*tmp1(1) + n2*tmp1(2) + n3*tmp1(3) + n4*tmp1(4)
         do x_i_ = 1, x_d_
            x_v(x_i_) =  + x_n1(x_i_)*tmp1(1) + x_n2(x_i_)*tmp1(2) + x_n
     &3(x_i_)*tmp1(3)
         enddo


         m = + n1*mm1 + n2*mm2 + n3*mm3 + n4*mm4
         do x_i_ = 1, x_d_
            x_m(x_i_) =  + x_n1(x_i_)*mm1 + x_n2(x_i_)*mm2 + x_n3(x_i_)*
     &mm3
         enddo


c     VARIABLEN FUER TEMPERATURBILANZ-DGL


         height = 4.0d+0* v * 0.001d+0 / (d*d * pi)
         do x_i_ = 1, x_d_
            x_height(x_i_) = x_v(x_i_)*height/v
         enddo
         aw = pi*d*height + pi * d*d / 4.0d+0
         do x_i_ = 1, x_d_
            x_aw(x_i_) = x_height(x_i_)*pi*d
         enddo
         tmpval = 2.0d+0*vw*rhow*cww
         tw = (kd*aw*temp + tmpval*tin) / (kd*aw+tmpval)
         do x_i_ = 1, x_d_
            x_tw(x_i_) = x_temp(x_i_)*kd*aw / (kd*aw+tmpval) + ( (x_aw(x
     &_i_)*kd*temp) * (kd*aw+tmpval ) - ( kd*aw*temp + tmpval*tin) * (x_
     &aw(x_i_)*kd ) )/ ( ( kd*aw+tmpval ) * ( kd*aw+tmpval ) )
         enddo


c     ARRHENIUS KINETIK


         tmpval = -ea1/rgas*(1.0d+0/temp - 1.0d+0/tref1)
         k1 = kref1 * dexp( tmpval )
         tmpval1 = ea1/( rgas*temp*temp )*k1
         do x_i_ = 1, x_d_
            x_k1(x_i_) =  + x_temp(x_i_)*tmpval1
         enddo


c     REAKTIONS-GESCHWINDIGKEITEN
```

```
         r1 = k1 * ( n1/v )**alpha1_1 * ( n2/v )**alpha1_2
         tmpval = alpha1_1 + alpha1_2
         tmpval1 = v**( -tmpval )
         tmpval2 = tmpval * r1 / v
         tmp1(1) = n2**alpha1_2
         tmp2(1) = alpha1_1 * n1**(alpha1_1-1.0d+0)
         tmp3(1) = tmp1(1) * tmp2(1)
         tmp1(2) = n1**alpha1_1
         tmp2(2) = alpha1_2 * n2**(alpha1_2-1.0d+0)
         tmp3(2) = tmp1(2) * tmp2(2)
         do x_i_ = 1, x_d_
            tmp(1) = x_k1(x_i_) * r1 / k1
            tmp(2) = -x_v(x_i_) * tmpval2
            tmp4(1) = tmp3(1) * x_n1(x_i_)
            tmp4(2) = tmp3(2) * x_n2(x_i_)
            tmpval2 = tmp4(1) + tmp4(2)
            tmp(3) = k1 * tmpval1 * tmpval2
            x_r1(x_i_) = tmp(1) + tmp(2) + tmp(3)
         enddo

c     GESAMTZULAUFRATE DER SPEZIES

         d_n1e = 0d+0

c     RECHTE SEITEN DES DIFFERENTIALGLEICHUNGSSYSTEMS

         w_reak = v * ( + r1*dh1 )
         w_feed =  + dme1*cp*(temp0_e1-temp)
         w_cool =  + kd*aw*(tw-temp)
         f(1) = ( w_reak + w_feed + w_cool ) / ( m*cp )
         do x_i_ = 1, x_d_
            x_f(x_i_,1) = ( x_v(x_i_) * w_reak / v + x_r1(x_i_)*v*dh1 -
     &x_temp(x_i_)*dme1*cp + x_aw(x_i_)*w_cool/aw + x_tw(x_i_)*kd*aw - x
     &_temp(x_i_)*kd*aw ) / ( m*cp ) - x_m(x_i_)*f(1)/m
         enddo
         tmpval =  - r1
         do x_i_ = 1, x_d_
            tmpval1 =  - x_r1(x_i_)
            x_f(x_i_,2) = x_v(x_i_) * tmpval  +  v * tmpval1
         enddo
         f(2) = v * (tmpval ) + d_n1e

         iflag = 0
      end
```

A.1.2 Automatisch erzeugte Ableitungs-Routine q_x_ffcn

```
c     ------------------------------------------------------------------
c     Automatisch erzeugtes Fortran77 File 'q_x_ffcn.f'
c     durch Java Modellgenerator
c     Gerd Ruecker (IWR der Universitaet Heidelberg)
c     Mon Nov 08 06:29:35 GMT+03:30 1999

c     Urethan-Reaktion (ODE)

c     # Reaktionen:               3
c     # Spezies:                  5
c     # Loesungsmittel:           1
c
c     # Differentialgleichungen:  3
```

```
c     # Algebraische Gleichungen: 3
c
c     # Quelltext Zeilen:         625
c     # chemische Variablen       98
c     ------------------------------------------------------------------

      subroutine q_x_ffcn( q__d_, x_d_, t, x, x_x, ldx_x, f, x_f, q_x_f,
     & ldq_x_f, ldx_f, p, q, q__q, ldq__q, rwh, iwh, iflag )

         implicit none

c     DEKLARATIONEN

         integer*4 maxDir, maxDir2
         parameter( maxDir = 100 )
         parameter( maxDir2 = 100 )
         real*8 t, x(*), f(*), p(*), q(*), rwh(*)
         integer*4 iwh(*), iflag

c     ABLEITUNGEN

         integer*4 x_i_
         integer*4 x_d_,ldx_x, ldx_f
         integer*4 ind, ind_1, ind1(2), ind2(2)
         real*8 x_x(ldx_x, *), x_f(ldx_f, *)
         integer*4 q__i_
         integer*4 q__d_, ldq__q, ldq_x_f
         real*8 q__q(ldq__q,*), q_x_f(ldq_x_f,ldx_f,*)

         real*8 tref1, tref2, tref4, tg2, kref1, kref2, kref4, kc2, ea1,
     & ea2, ea4, alpha1_1, alpha1_2, alpha2_1, alpha2_3, alpha3_4, alpha
     &4_1, tmpval, tmp1(6), tmpval1, tmpval2, tmpval3, tmpval4, tmpval5,
     & tmp(7), tmp2(6), tmp3(6), tmp4(6), tmp5(6), tmp6(6), tmp7(6,6), t
     &mp8(6), tmp10(6), log_t(6), d_log_t(6), log_t1(6), log_t2(6), log_
     &t3(6), log_t4(6), log_t5(6), block(5), rgas, pi, dh1, dh2, dh3, dh
     &4, mm1, rho1, delta_n1, mm2, rho2, delta_n2, mm3, rho3, delta_n3,
     &mm4, rho4, mm5, rho5, mm6, rho6, na1, na2, na6, na1e1, n1e1, na6e1
     &, n6e1, na2e2, n2e2, na6e2, n6e2, feed_1, d_feed_1, feed_2, d_feed
     &_2, d_n1e, d_n2e, d_n3e, temp, tin, n1, n2, n3, n4, n5, n6, r1, k1
     &, r2, k2, r3, k3, r4, k4, v, m

c     DEKLARATIONEN FUER DIE ERSTEN ABLEITUNGEN

         real*8 x_delta_n1(maxDir), x_delta_n2(maxDir), x_delta_n3(maxDi
     &r), x_n1(maxDir), x_n2(maxDir), x_n3(maxDir), x_n4(maxDir), x_n5(m
     &axDir), x_r1(maxDir), x_r2(maxDir), x_r3(maxDir), x_r4(maxDir), x_
     &v(maxDir)

c     DEKLARATIONEN FUER DIE ZWEITEN ABLEITUNGEN

         real*8 q__na1(maxDir2), q__na2(maxDir2), q__na6(maxDir2), q__na
     &1e1(maxDir2), q__na6e1(maxDir2), q__na2e2(maxDir2), q__na6e2(maxDi
     &r2), q__tin(maxDir2), q__feed_1(maxDir2), q__d_feed_1(maxDir2), q_
     &_feed_2(maxDir2), q__d_feed_2(maxDir2), q__temp(maxDir2), q__n1e1(
     &maxDir2), q__n6e1(maxDir2), q__n2e2(maxDir2), q__n6e2(maxDir2), q_
     &_delta_n1(maxDir2), q__delta_n2(maxDir2), q__n6(maxDir2), q__n5(ma
     &xDir2), q__n4(maxDir2), q__v(maxDir2), q__k1(maxDir2), q__k2(maxDi
     &r2), q__k3(maxDir2), q__k4(maxDir2), q__n1(maxDir2), q__n2(maxDir2
     &), q__r1(maxDir2), q__n3(maxDir2), q__r2(maxDir2), q__r3(maxDir2),
     & q__r4(maxDir2)
         real*8 q_x_n6(maxDir2,maxDir), q_x_n5(maxDir2,maxDir), q_x_n4(m
     &axDir2,maxDir), q_x_v(maxDir2,maxDir), q_x_n1(maxDir2,maxDir), q_x
     &_n2(maxDir2,maxDir), q_x_r1(maxDir2,maxDir), q_x_n3(maxDir2,maxDir
     &), q_x_r2(maxDir2,maxDir), q_x_r3(maxDir2,maxDir), q_x_r4(maxDir2,
     &maxDir)
```

```
c     ZUSTANDSVARIABLEN

      n1 = x(1)
      do x_i_ = 1, x_d_
         x_n1(x_i_) = x_x(x_i_,1)
      enddo
      n2 = x(2)
      do x_i_ = 1, x_d_
         x_n2(x_i_) = x_x(x_i_,2)
      enddo
      n3 = x(3)
      do x_i_ = 1, x_d_
         x_n3(x_i_) = x_x(x_i_,3)
      enddo

c     SKALIERTE PARAMETER

      kref1 = p(1) * 5.0d-4
      ea1 = p(2) * 35240.0d+0
      kref2 = p(3) * 8.0d-8
      ea2 = p(4) * 85000.0d+0
      dh2 = p(7) * (-17031.0d+0)
      kc2 = p(8) * 0.17d+0
      kref4 = p(5) * 1.0d-8
      ea4 = p(6) * 35000.0d+0

c     STEUERGROESSEN

      na1 = q(1)
      do q__i_ = 1, q__d_
         q__na1(q__i_) = q__q(q__i_,1)
      enddo
      na2 = q(2)
      do q__i_ = 1, q__d_
         q__na2(q__i_) = q__q(q__i_,2)
      enddo
      na6 = q(3)
      do q__i_ = 1, q__d_
         q__na6(q__i_) = q__q(q__i_,3)
      enddo
      na1e1 = q(4)
      do q__i_ = 1, q__d_
         q__na1e1(q__i_) = q__q(q__i_,4)
      enddo
      na6e1 = q(5)
      do q__i_ = 1, q__d_
         q__na6e1(q__i_) = q__q(q__i_,5)
      enddo
      na2e2 = q(6)
      do q__i_ = 1, q__d_
         q__na2e2(q__i_) = q__q(q__i_,6)
      enddo
      na6e2 = q(7)
      do q__i_ = 1, q__d_
         q__na6e2(q__i_) = q__q(q__i_,7)
      enddo

c     DISKRETISIERUNG DER STEUERFUNKTIONEN

c     AUSSENTEMPERATUR DES REAKTORS
```

```
         if ( iwh(3) .eq. 0 .or. iwh(3) .eq. 1 ) then
            ind =  iwh(2) + iwh(4) - 1
            ind_1 = 0
            tin = q(ind)
            do q__i_ = 1, q__d_
               q__tin(q__i_) = q__q(q__i_,ind)
            enddo
         else if ( iwh(3) .eq. 2 .or. iwh(3) .eq. 3 ) then
            ind =  iwh(2) + 2*iwh(4) - 2
            ind_1 = ind + 1
            tin = q(ind) + ( t-rwh(2) ) * q(ind_1)
            do q__i_ = 1, q__d_
               q__tin(q__i_) = q__q(q__i_, iwh(2) + 2*iwh(4) - 2 ) + ( t
     &-rwh(2) )*q__q(q__i_, iwh(2) + 2*iwh(4) - 1 )
            enddo
         endif

c     EINTRAEGE AM REAKTOR

         if ( iwh(6) .eq. 0 .or. iwh(6) .eq. 1 ) then
            ind1(1) =  iwh(5) + iwh(7) - 1
            ind2(1) = 0
            feed_1 = q(ind1(1))
            do q__i_ = 1, q__d_
               q__feed_1(q__i_) = q__q(q__i_,ind1(1))
            enddo
            d_feed_1 = 0.0d+0
            do q__i_ = 1, q__d_
               q__d_feed_1(q__i_) = 0.0d+0
            enddo
         else if ( iwh(6) .eq. 2 .or. iwh(6) .eq. 3 ) then
            ind1(1) =  iwh(5) + 2*iwh(7) - 2
            ind2(1) = ind1(1) + 1
            feed_1 = q(ind1(1)) + ( t-rwh(3) ) * q(ind2(1))
            d_feed_1 = q(ind2(1))
            do q__i_ = 1, q__d_
               q__feed_1(q__i_) = q__q(q__i_, iwh(5) + 2*iwh(7) - 2 ) +
     &( t-rwh(3) )*q__q(q__i_, iwh(5) + 2*iwh(7) - 1 )
               q__d_feed_1(q__i_) = q__q(q__i_, iwh(5) + 2*iwh(7) - 1 )
            enddo
         endif
         if ( iwh(9) .eq. 0 .or. iwh(9) .eq. 1 ) then
            ind1(2) =  iwh(8) + iwh(10) - 1
            ind2(2) = 0
            feed_2 = q(ind1(2))
            do q__i_ = 1, q__d_
               q__feed_2(q__i_) = q__q(q__i_,ind1(2))
            enddo
            d_feed_2 = 0.0d+0
            do q__i_ = 1, q__d_
               q__d_feed_2(q__i_) = 0.0d+0
            enddo
         else if ( iwh(9) .eq. 2 .or. iwh(9) .eq. 3 ) then
            ind1(2) =  iwh(8) + 2*iwh(10) - 2
            ind2(2) = ind1(2) + 1
            feed_2 = q(ind1(2)) + ( t-rwh(4) ) * q(ind2(2))
            d_feed_2 = q(ind2(2))
            do q__i_ = 1, q__d_
               q__feed_2(q__i_) = q__q(q__i_, iwh(8) + 2*iwh(10) - 2 ) +
     & ( t-rwh(4) )*q__q(q__i_, iwh(8) + 2*iwh(10) - 1 )
               q__d_feed_2(q__i_) = q__q(q__i_, iwh(8) + 2*iwh(10) - 1 )
            enddo
         endif

c     KONSTANTEN
```

```
          alpha1_1 = 1.0d+0
          alpha1_2 = 1.0d+0
          tref1 = 363.16d+0
          dh1 = -80000.0d+0
          alpha2_1 = 1.0d+0
          alpha2_3 = 1.0d+0
          alpha3_4 = 1.0d+0
          tref2 = 363.16d+0
          tg2 = 363.16d+0
          alpha4_1 = 2.0d+0
          tref4 = 363.16d+0
          dh4 = -75000.0d+0
          mm1 = 0.11911d+0
          rho1 = 1095.0d+0
          mm2 = 0.07412d+0
          rho2 = 809.0d+0
          mm3 = 0.19323d+0
          rho3 = 1415.0d+0
          mm4 = 0.31234d+0
          rho4 = 1528.0d+0
          mm5 = 0.35733d+0
          rho5 = 1451.0d+0
          mm6 = 0.07806d+0
          rho6 = 1101.0d+0
          rgas = 8.314d+0
          pi = 3.141592653589793d+0

c     HILFSGROESSEN

          temp = tin
          do q__i_ = 1, q__d_
             q__temp(q__i_) = q__tin(q__i_)
          enddo
          n1e1 = na1e1 * feed_1
          do q__i_ = 1, q__d_
             q__n1e1(q__i_) = q__na1e1(q__i_)*feed_1 + na1e1*q__feed_1(q_
     &_i_)
          enddo
          n6e1 = na6e1 * feed_1
          do q__i_ = 1, q__d_
             q__n6e1(q__i_) = q__na6e1(q__i_)*feed_1 + na6e1*q__feed_1(q_
     &_i_)
          enddo
          n2e2 = na2e2 * feed_2
          do q__i_ = 1, q__d_
             q__n2e2(q__i_) = q__na2e2(q__i_)*feed_2 + na2e2*q__feed_2(q_
     &_i_)
          enddo
          n6e2 = na6e2 * feed_2
          do q__i_ = 1, q__d_
             q__n6e2(q__i_) = q__na6e2(q__i_)*feed_2 + na6e2*q__feed_2(q_
     &_i_)
          enddo

c     MOLZAHLAENDERUNG

          delta_n1 = n1 - na1 - n1e1
          do x_i_ = 1, x_d_
             x_delta_n1(x_i_) = x_n1(x_i_)
          enddo
          do q__i_ = 1, q__d_
             q__delta_n1(q__i_) =  - q__na1(q__i_) - q__n1e1(q__i_)
          enddo
```

```
          delta_n2 = n2 - na2 - n2e2
          do x_i_ = 1, x_d_
             x_delta_n2(x_i_) = x_n2(x_i_)
          enddo
          do q__i_ = 1, q__d_
             q__delta_n2(q__i_) =  - q__na2(q__i_) - q__n2e2(q__i_)
          enddo
          delta_n3 = n3
          do x_i_ = 1, x_d_
             x_delta_n3(x_i_) = x_n3(x_i_)
          enddo

c     ALGEBRAISCHE GLEICHUNGEN

          n6 = na6 + n6e1 + n6e2
          do q__i_ = 1, q__d_
             q__n6(q__i_) = q__na6(q__i_) + q__n6e1(q__i_) + q__n6e2(q__i
     &_)
          enddo
          n5 =  - 0.3333333333333333d+0*delta_n1 + 0.6666666666666666d+0*
     &delta_n2 + 0.3333333333333333d+0*delta_n3
          do x_i_ = 1, x_d_
             x_n5(x_i_) =  - 0.3333333333333333d+0*x_delta_n1(x_i_) + 0.6
     &666666666666666d+0*x_delta_n2(x_i_) + 0.3333333333333333d+0*x_delt
     &a_n3(x_i_)
             do q__i_ = 1, q__d_
                q_x_n5(q__i_,x_i_) = 0.0d+0
             enddo
          enddo
          do q__i_ = 1, q__d_
             q__n5(q__i_) =  - 0.3333333333333333d+0*q__delta_n1(q__i_) +
     & 0.6666666666666666d+0*q__delta_n2(q__i_)
          enddo
          n4 =  - delta_n2 - delta_n3
          do x_i_ = 1, x_d_
             x_n4(x_i_) =  - x_delta_n2(x_i_) - x_delta_n3(x_i_)
             do q__i_ = 1, q__d_
                q_x_n4(q__i_,x_i_) = 0.0d+0
             enddo
          enddo
          do q__i_ = 1, q__d_
             q__n4(q__i_) =  - q__delta_n2(q__i_)
          enddo

c     GESAMTVOLUMEN IM REAKTOR

          tmp1(1) = mm1 / rho1
          tmp1(2) = mm2 / rho2
          tmp1(3) = mm3 / rho3
          tmp1(4) = mm4 / rho4
          tmp1(5) = mm5 / rho5
          tmp1(6) = mm6 / rho6
          v =  + n1*tmp1(1) + n2*tmp1(2) + n3*tmp1(3) + n4*tmp1(4) + n5*t
     &mp1(5) + n6*tmp1(6)
          do x_i_ = 1, x_d_
             x_v(x_i_) =  + x_n1(x_i_)*tmp1(1) + x_n2(x_i_)*tmp1(2) + x_n
     &3(x_i_)*tmp1(3) + x_n4(x_i_)*tmp1(4) + x_n5(x_i_)*tmp1(5)
             do q__i_ = 1, q__d_
                q_x_v(q__i_,x_i_) = 0.0d+0
             enddo
          enddo
          do q__i_ = 1, q__d_
             q__v(q__i_) =  + q__n4(q__i_)*tmp1(4) + q__n5(q__i_)*tmp1(5)
     & + q__n6(q__i_)*tmp1(6)
          enddo
```

```
c     ARRHENIUS KINETIK

          tmpval = -ea1/rgas*(1.0d+0/temp - 1.0d+0/tref1)
          k1 = kref1 * dexp( tmpval )
          tmpval1 = ea1/( rgas*temp*temp )*k1
          do q__i_ = 1, q__d_
             q__k1(q__i_) =  + q__temp(q__i_)*ea1/( rgas*temp*temp )*k1
          enddo
          tmpval = -ea2/rgas*(1.0d+0/temp - 1.0d+0/tref2)
          k2 = kref2 * dexp( tmpval )
          tmpval1 = ea2/( rgas*temp*temp )*k2
          do q__i_ = 1, q__d_
             q__k2(q__i_) =  + q__temp(q__i_)*ea2/( rgas*temp*temp )*k2
          enddo
          tmpval = -dh2/rgas*(1.0/ temp - 1.0/tg2)
          tmpval1 = dh2/(rgas*temp*temp)*k3
          k3 = k2 / (  kc2 * dexp( tmpval ) )
          do q__i_ = 1, q__d_
             q__k3(q__i_) = q__k2(q__i_)*k3/k2 - q__temp(q__i_)*dh2/( rga
     &s*temp*temp )*k3
          enddo
          tmpval = -ea4/rgas*(1.0d+0/temp - 1.0d+0/tref4)
          k4 = kref4 * dexp( tmpval )
          tmpval1 = ea4/( rgas*temp*temp )*k4
          do q__i_ = 1, q__d_
             q__k4(q__i_) =  + q__temp(q__i_)*ea4/( rgas*temp*temp )*k4
          enddo

c     REAKTIONS-GESCHWINDIGKEITEN

          r1 = k1 * ( n1/v ) * ( n2/v )
          tmpval = alpha1_1 + alpha1_2
          tmpval1 = 1.0d+0 / ( v * v )
          tmpval2 = tmpval * r1 / v
          tmp7(1,2) = 1.0d+0
          tmp1(1) = n2
          tmp2(1) = 1.0d+0
          tmp5(1) = ( alpha1_1 - 1.0d+0 ) * 1.0d+0 * n1**(alpha1_1-2.0d+0
     &) * tmp1(1)
          tmp3(1) = tmp1(1) * tmp2(1)
          tmp7(2,1) = 1.0d+0
          tmp1(2) = n1
          tmp2(2) = 1.0d+0
          tmp5(2) = ( alpha1_2 - 1.0d+0 ) * 1.0d+0 * n2**(alpha1_2-2.0d+0
     &) * tmp1(2)
          tmp3(2) = tmp1(2) * tmp2(2)
          do q__i_ = 1, q__d_
             tmp(1) = q__k1(q__i_) * r1 / k1
             tmp(2) = -q__v(q__i_) * tmpval * r1 / v
             q__r1(q__i_) = tmp(1) + tmp(2)
          enddo
          do x_i_ = 1, x_d_
             tmp(2) = -x_v(x_i_) * tmpval2
             tmp4(1) = tmp3(1) * x_n1(x_i_)
             tmp6(1) = tmp5(1) * x_n1(x_i_) * tmp1(1)
             tmp8(1) = tmp2(1) * x_n1(x_i_)
             tmp4(2) = tmp3(2) * x_n2(x_i_)
             tmp6(2) = tmp5(2) * x_n2(x_i_) * tmp1(2)
             tmp8(2) = tmp2(2) * x_n2(x_i_)
             tmpval2 = tmp4(1) + tmp4(2)
             tmp(3) = k1 * tmpval1 * tmpval2
             x_r1(x_i_) = tmp(2) + tmp(3)
             do q__i_ = 1, q__d_
```

```
          tmpval4 = q__k1(q__i_) * r1 / k1-q__v(q__i_) * tmpval * r
&1 / v
       block(2) = - tmpval * r1 / v * ( q_x_v(q__i_,x_i_) - q__v(q_
&_i_) * x_v(x_i_) / v + x_v(x_i_) * tmpval4 / r1 )
       block(3) = q__k1(q__i_) * tmpval1 * tmpval2 - q__v(q__i_) *
&tmpval * tmp(3) / v + k1 * tmpval1 * ( tmp6(1) * q__n1(q__i_) + tm
&p3(1) * q_x_n1(q__i_,x_i_) + tmp6(2) * q__n2(q__i_) + tmp3(2) * q_
&x_n2(q__i_,x_i_) )
       q_x_r1(q__i_,x_i_) = block(2) + block(3)
       enddo
    enddo


    r2 = k2 * ( n1/v ) * ( n3/v )
    tmpval = alpha2_1 + alpha2_3
    tmpval1 = 1.0d+0 / ( v * v )
    tmpval2 = tmpval * r2 / v
    tmp7(1,3) = 1.0d+0
    tmp1(1) = n3
    tmp2(1) = 1.0d+0
    tmp5(1) = ( alpha2_1 - 1.0d+0 ) * 1.0d+0 * n1**(alpha2_1-2.0d+0
&) * tmp1(1)
    tmp3(1) = tmp1(1) * tmp2(1)
    tmp7(3,1) = 1.0d+0
    tmp1(3) = n1
    tmp2(3) = 1.0d+0
    tmp5(3) = ( alpha2_3 - 1.0d+0 ) * 1.0d+0 * n3**(alpha2_3-2.0d+0
&) * tmp1(3)
    tmp3(3) = tmp1(3) * tmp2(3)
    do q__i_ = 1, q__d_
       tmp(1) = q__k2(q__i_) * r2 / k2
       tmp(2) = -q__v(q__i_) * tmpval * r2 / v
       q__r2(q__i_) = tmp(1) + tmp(2)
    enddo
    do x_i_ = 1, x_d_
       tmp(2) = -x_v(x_i_) * tmpval2
       tmp4(1) = tmp3(1) * x_n1(x_i_)
       tmp6(1) = tmp5(1) * x_n1(x_i_) * tmp1(1)
       tmp8(1) = tmp2(1) * x_n1(x_i_)
       tmp4(3) = tmp3(3) * x_n3(x_i_)
       tmp6(3) = tmp5(3) * x_n3(x_i_) * tmp1(3)
       tmp8(3) = tmp2(3) * x_n3(x_i_)
       tmpval2 = tmp4(1) + tmp4(3)
       tmp(3) = k2 * tmpval1 * tmpval2
       x_r2(x_i_) = tmp(2) + tmp(3)
       do q__i_ = 1, q__d_
          tmpval4 = q__k2(q__i_) * r2 / k2-q__v(q__i_) * tmpval * r
&2 / v
       block(2) = - tmpval * r2 / v * ( q_x_v(q__i_,x_i_) - q__v(q_
&_i_) * x_v(x_i_) / v + x_v(x_i_) * tmpval4 / r2 )
       block(3) = q__k2(q__i_) * tmpval1 * tmpval2 - q__v(q__i_) *
&tmpval * tmp(3) / v + k2 * tmpval1 * ( tmp6(1) * q__n1(q__i_) + tm
&p3(1) * q_x_n1(q__i_,x_i_) + tmp6(3) * q__n3(q__i_) + tmp3(3) * q_
&x_n3(q__i_,x_i_) )
       q_x_r2(q__i_,x_i_) = block(2) + block(3)
       enddo
    enddo


    r3 = k3 * ( n4/v )
    tmpval = alpha3_4
    tmpval1 = 1.0d+0 / v
    tmpval2 = tmpval * r3 / v
    tmp1(4) = 1.0d+0
    tmp2(4) = 1.0d+0
    tmp5(4) = ( alpha3_4 - 1.0d+0 ) * 1.0d+0 * n4**(alpha3_4-2.0d+0
```

```
&) * tmp1(4)
    tmp3(4) = tmp1(4) * tmp2(4)
    do q__i_ = 1, q__d_
       tmp(1) = q__k3(q__i_) * r3 / k3
       tmp(2) = -q__v(q__i_) * tmpval * r3 / v
       tmp4(4) = tmp3(4) * q__n4(q__i_)
       tmpval2 = tmp4(4)
       tmp(3) = k3 * tmpval1 * tmpval2
       q__r3(q__i_) = tmp(1) + tmp(2) + tmp(3)
    enddo
    do x_i_ = 1, x_d_
       tmp(2) = -x_v(x_i_) * tmpval2
       tmp4(4) = tmp3(4) * x_n4(x_i_)
       tmp6(4) = tmp5(4) * x_n4(x_i_) * tmp1(4)
       tmp8(4) = tmp2(4) * x_n4(x_i_)
       tmpval2 = tmp4(4)
       tmp(3) = k3 * tmpval1 * tmpval2
       x_r3(x_i_) = tmp(2) + tmp(3)
       do q__i_ = 1, q__d_
          tmpval4 = q__k3(q__i_) * r3 / k3-q__v(q__i_) * tmpval * r
&3 / v  +  k3 * tmpval1 * (  + q__n4(q__i_) * tmp3(4) )
       block(2) = - tmpval * r3 / v * ( q_x_v(q__i_,x_i_) - q__v(q_
&_i_) * x_v(x_i_) / v + x_v(x_i_) * tmpval4 / r3 )
       block(3) = q__k3(q__i_) * tmpval1 * tmpval2 - q__v(q__i_) *
&tmpval * tmp(3) / v + k3 * tmpval1 * ( tmp6(4) * q__n4(q__i_) + tm
&p3(4) * q_x_n4(q__i_,x_i_) )
       q_x_r3(q__i_,x_i_) = block(2) + block(3)
       enddo
    enddo

    r4 = k4 * ( n1/v )**alpha4_1
    tmpval = alpha4_1
    tmpval1 = v**( -tmpval )
    tmpval2 = tmpval * r4 / v
    tmp1(1) = 1.0d+0
    tmp2(1) = alpha4_1 * n1**(alpha4_1-1.0d+0)
    tmp5(1) = ( alpha4_1 - 1.0d+0 ) * alpha4_1 * n1**(alpha4_1-2.0d
&+0) * tmp1(1)
    tmp3(1) = tmp1(1) * tmp2(1)
    do q__i_ = 1, q__d_
       tmp(1) = q__k4(q__i_) * r4 / k4
       tmp(2) = -q__v(q__i_) * tmpval * r4 / v
       q__r4(q__i_) = tmp(1) + tmp(2)
    enddo
    do x_i_ = 1, x_d_
       tmp(2) = -x_v(x_i_) * tmpval2
       tmp4(1) = tmp3(1) * x_n1(x_i_)
       tmp6(1) = tmp5(1) * x_n1(x_i_) * tmp1(1)
       tmp8(1) = tmp2(1) * x_n1(x_i_)
       tmpval2 = tmp4(1)
       tmp(3) = k4 * tmpval1 * tmpval2
       x_r4(x_i_) = tmp(2) + tmp(3)
       do q__i_ = 1, q__d_
          tmpval4 = q__k4(q__i_) * r4 / k4-q__v(q__i_) * tmpval * r
&4 / v
       block(2) = - tmpval * r4 / v * ( q_x_v(q__i_,x_i_) - q__v(q_
&_i_) * x_v(x_i_) / v + x_v(x_i_) * tmpval4 / r4 )
       block(3) = q__k4(q__i_) * tmpval1 * tmpval2 - q__v(q__i_) *
&tmpval * tmp(3) / v + k4 * tmpval1 * ( tmp6(1) * q__n1(q__i_) + tm
&p3(1) * q_x_n1(q__i_,x_i_) )
       q_x_r4(q__i_,x_i_) = block(2) + block(3)
       enddo
    enddo

c     GESAMTZULAUFRATE DER SPEZIES
```

```
        d_n1e = na1e1 * d_feed_1
        d_n2e = na2e2 * d_feed_2
        d_n3e = 0d+0


c     RECHTE SEITEN DES DIFFERENTIALGLEICHUNGSSYSTEMS


        tmpval =  - r1 - r2 + r3 - 3.0d+0*r4
        do x_i_ = 1, x_d_
           tmpval1 =  - x_r1(x_i_) - x_r2(x_i_) + x_r3(x_i_) - 3.0d+0*x
     &_r4(x_i_)
           x_f(x_i_,1) = x_v(x_i_) * tmpval  +  v * tmpval1
           do q__i_ = 1, q__d_
              q_x_f(q__i_,x_i_,1) = q_x_v(q__i_,x_i_) * tmpval + x_v(x_
     &i_) * (  - q__r1(q__i_) - q__r2(q__i_) + q__r3(q__i_) - 3.0d+0*q__
     &r4(q__i_) ) + q__v(q__i_) * tmpval1 + v * (  - q_x_r1(q__i_,x_i_)
     &- q_x_r2(q__i_,x_i_) + q_x_r3(q__i_,x_i_) - 3.0d+0*q_x_r4(q__i_,x_
     &i_) )
           enddo
        enddo
        f(1) = v * (tmpval ) + d_n1e
        tmpval =  - r1
        do x_i_ = 1, x_d_
           tmpval1 =  - x_r1(x_i_)
           x_f(x_i_,2) = x_v(x_i_) * tmpval  +  v * tmpval1
           do q__i_ = 1, q__d_
              q_x_f(q__i_,x_i_,2) = q_x_v(q__i_,x_i_) * tmpval + x_v(x_
     &i_) * (  - q__r1(q__i_) ) + q__v(q__i_) * tmpval1 + v * (  - q_x_r
     &1(q__i_,x_i_) )
           enddo
        enddo
        f(2) = v * (tmpval ) + d_n2e
        tmpval =  + r1 - r2 + r3
        do x_i_ = 1, x_d_
           tmpval1 =  + x_r1(x_i_) - x_r2(x_i_) + x_r3(x_i_)
           x_f(x_i_,3) = x_v(x_i_) * tmpval  +  v * tmpval1
           do q__i_ = 1, q__d_
              q_x_f(q__i_,x_i_,3) = q_x_v(q__i_,x_i_) * tmpval + x_v(x_
     &i_) * (  + q__r1(q__i_) - q__r2(q__i_) + q__r3(q__i_) ) + q__v(q__
     &i_) * tmpval1 + v * (  + q_x_r1(q__i_,x_i_) - q_x_r2(q__i_,x_i_) +
     & q_x_r3(q__i_,x_i_) )
           enddo
        enddo
        f(3) = v * (tmpval ) + d_n3e

        iflag = 0
      end
```

A.1.3 Zweite Ableitungen mit Numerischen Differenzen

```
c     Gerd Ruecker (IWR der Universitaet Heidelberg)
c     Mon Oct 18 20:18:57 GMT+03:30 1999
c
c     Berechnung der zweiten Ableitung der rechten Seiten
c     nach Steuergroessen mit dem einseitigen Differenzenquotienten
c     Berechnung des Deltas mit 2-Norm


      subroutine q_x_ffcn( q__d_, x_d_, t, x, x_x, ldx_x, f, x_f, q_x_f,
     & ldq_x_f, ldx_f, p, q, q__q, ldq__q, rwh, iwh, iflag )


        implicit none
```

```
c     DEKLARATIONEN

         integer*4 maxDir, maxDir2
         parameter( maxDir = 100 )
         parameter( maxDir2 = 100 )

         real*8 t, x(*), f(*), p(*), q(*), rwh(*)
         integer*4 iwh(*), iflag



c     ABLEITUNGEN

         integer*4 x_i_
         integer*4 x_d_,ldx_x, ldx_f
         real*8 x_x(ldx_x, *), x_f(ldx_f, *)

         integer*4 q__i_
         integer*4 q__d_, ldq__q, ldq_x_f
         real*8 q__q(ldq__q,*), q_x_f(ldq_x_f,ldx_f,*)



c     einseitiger Differenzenquotient

         integer*4 lp
         integer*4 lp2
         real*8 delta2, q__delta(maxDir), x_f_delta(maxDir,20)
         integer*4 i

c     Berechnung eines optimalen Deltas mit 2-Norm
         delta2 = 0.0d+0
         do lp2 = 1, iwh(1)
            delta2 = delta2 + q(lp2) * q(lp2)
         enddo
         delta2 = 0.00000001 * sqrt(delta2) + 0.00000001

         call x_ffcn( x_d_, t, x, x_x, ldx_x, f, x_f, ldx_f, p, q, rwh,
     &iwh, iflag )

         do q__i_ = 1, q__d_

            do lp2 = 1, iwh(1)
               q__delta(lp2) = q(lp2) + delta2 * q__q(q__i_,lp2)
            enddo

            call x_ffcn( x_d_, t, x, x_x, ldx_x, f, x_f_delta, ldx_f, p,
     & q__delta, rwh, iwh, iflag )

            do x_i_ = 1, x_d_

               do lp = 1, 20
                  q_x_f(q__i_,x_i_,lp) = 1.0d+0/delta2 * ( x_f(x_i_,lp)
     &- x_f_delta(x_i_,lp) )
               enddo

            enddo

         enddo



      end
```

A.2 Implementierte Packages und Java-Klassen

`shared`	gemeinsame Datenstrukturen, Modellierung der Parameter, Zustandsvariablen und Steuergrößen
`chem`	chemische Modellierung eines Reaktionssystems, Wärmebilanzführung, verfahrenstechnische Komponenten, Bestimmung des DAE-Systems
`math`	abstrakter Datentyp für Matrizen, Berechnung eines vollrangigen Subsystems des DAE-Systems, Gaußalgorithmus
`misc`	Klassen aller statischen Methoden, Namenskonventionen für die Quelltexte, Methoden zum Lesen und Schreiben von Dateien
`code`	Klassen für die Erzeugung der Fortran- oder C-Quelltexte der Modellgleichungen und deren erste und zweite Ableitungen
`gui`	Klassen für die Erstellung der graphischen Benutzeroberfläche

Tabelle A.1: Packages des implementierten Java-Programmes MGAD

```
public  class Data
public  class Globals
public  class Data
public  class MainWindow      extends JFrame
public  class ValuePanel      extends JPanel
public  class FilesPanel      extends JPanel
public  class BasePanel       extends JPanel  implements ItemListener
public  class ReactorPanel    extends JPanel  implements ItemListener
public  class EquationPanel   extends JPanel  implements ItemListener,
                                                ActionListener
public  class SpeciesPanel    extends JPanel
public  class EintragPanel    extends JPanel
public  class WaermePanel     extends JPanel  implements ItemListener
public  class ViewFilesPanel  extends JPanel
```

Tabelle A.2: Klassen im Package *gui*

public	class	IntC	
public	class	DoubleC	
public	class	Einheit	
public	class	Einheitenliste	
public	class	CommonDataSup	implements Serializable
public final	class	CommonData	
public	class	Zeit	
public abstract	class	KPS	
public	class	Kosten	extends KPS
public	class	Konst	extends KPS
public	class	Steuer	extends KPS
public	class	Param	extends KPS
public	class	Zustands	extends KPS
public	class	Steuerintervall	
public	class	Steuerfunktion	
public	class	Dynamik	
public	class	Experiment	

Tabelle A.3: Klassen im Package *shared*

public	class React	
public	class Species	
public	class Waermebilanz	
public abstract	class Reaktor	
public	class Kessel	extends Reaktor
public	class Technique	
public	class ReactSystem	

Tabelle A.4: Klassen im Package *chem*

public	class Matrix	
public abstract	class Gauss	
public	class GenBilanz	

Tabelle A.5: Klassen im Package *math*

public final	class StatMeth	
public	class ReadInput	
public	class PlotExpression	
public	class PlotFile	

Tabelle A.6: Klassen im Package *misc*

public	class Node	
public	class Graph	
public final	class N	
public	class Coding	
public	class Fortran	extends Object
public abstract	class Generator	
public	class GenFfcn	extends Generator
public	class GenGfcn	extends Generator
public	class GenMessSig	extends Generator
public	class GenPlot	extends Generator
public	class GenDiff	
public	class GenDiff2	

Tabelle A.7: Klassen im Package *code*

Tabellenverzeichnis

Abbildungsverzeichnis

Index

Literaturverzeichnis

Barrow, G. M.: (1973). *Physikalische Chemie.* Bohman Vieweg, 5. Auflage.

Bauer, I.: (1996). *DAESOL, a BDF-code for the numerical solution of differential algebraic equations.*

Bauer, I.: (1999). *Numerische Verfahren zur Lösung von Anfangswertaufgaben und zur Generierung von ersten und zweiten Ableitungen mit Anwendung auf Optimierungsaufgaben in Chemie und Verfahrenstechnik.* Doktorarbeit, Universität Heidelberg.

Bauer, I., Heilig, M., Körkel, S., Kud, A., Mayer, A., Wörz, O.: (1998a). *Optimale Versuchsplanung am Beispiel einer Phosphin- und Urethanreaktion.* In *Optimale Versuchsplanung für nichtlineare Prozesse.* DECHEMA, e.V.

Bauer, I., Körkel, S., Bock, H. G., Schlöder, J. P.: (1998b). *Optimale Versuchsplanung für dynamische Systeme aus der chemischen Reaktionskinetik.* In *Optimale Versuchsplanung für nichtlineare Prozesse.* DECHEMA, e.V.

Bischof, C., Carle, A., Corlisss, G., Griewank, A., Hovland, P.: (1992). ADIFOR – Generating Derivative Codes from Fortran Programs. *Scientific Programming*, 1(1), S. 1–29.

Bischof, C., Carle, A., Khademi, P., Mauer, A., Hovland, P.: (1995). *ADIFOR 2.0 Users's Guide (Revision C).* Mathematics and Computer Science Division, Aragonne Laboratory. Technical Memorandum No. 192.

Bock, H. G.: (1985). *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen.* Doktorarbeit, Universität Bonn.

Bock, H. G.: (1998). *Numerische Mathematik 2.* (Unveröffentlichte) Vorlesungsmitschrift, Interdisziplinäres Zentrum für Wissenschaftliches Rechnen der Universität Heidelberg, Wintersemester 1997/1998.

Bock, H. G., Schlöder, J. P., Schulz, V. H.: (1994). *Numerik großer Differential-Algebraischer Gleichungen – Simulation und Optimierung.* Technischer Bericht, Interdisziplinäres Zentrum für Wissenschaftliches Rechnen der Universität Heidelberg.

Christianson, B.: (1992). Automatic Hessians by reverse accumulation. *IMA Journal of Numerical Analysis*, 12, S. 135–150.

Curtis, R., Powell, M. J. D., Reid, J. K.: (1974). On the estimation of sparse Jacobian matrices. *J. Inst. Math. Appl.*, 13, S. 117–119.

FANTANA, A., GUTFLEISCH, M.: (1998). *Berechnung der Bilanzmatrix aus der Stöchiometriematrix bei der Aufstellung des DAE-Systems von chemischen Reaktionen.* Interdisziplinäres Zentrum für Wissenschaftliches Rechnen der Universität Heidelberg. Software-Praktikum.

FISCHER, G.: (1989). *Lineare Algebra.* Vieweg Studium.

GEITNER, U.: (1995). *Automatisches Berechnung von dünnbesetzten Jacobimatrizen nach dem Ansatz von Newsam-Ramsdell.* Diplomarbeit, TU Dresden, Dept. of Mathematics.

GILL, P., MURRAY, W., SAUNDERS, M. A.: (1998). *User's Guide for SNOPT 5.3: a Fortran Package for Large-Scale Nonlinear Programming.* Technischer Bericht, Department of Mathematics, University of California, San Diego.

GILL, P., MURRAY, W., WRIGHT, M.: (1981). *Practical Optimization.* Academic Press Limited.

GRIEWANK, A.: (1994). *Tutorial on Computational Differentiation and Optimization.* In *XV. International Mathematical Programming Symposium Ann Arbor Michigan.* Institute of Scientific Computing Technical University Dresden and Mathematics and Computer Science Division, Aragonne National Laboratory.

GRIEWANK, A., CHRISTIANSEN, B.: (1998). Evaluating derivatives - principles and techniques of computational differentiation. Draft of Book, Institute of Scientific Computing, Technical University Dresden.

GRIEWANK, A., JUEDES, D., UTKE, J.: (1996). ADOL-C: A Package for the Automatic Differentiation Written in C/C++. *ACM Trans. Math. Software*, 22.

GRIEWANK, A., UTKE, J.: (1995). *Automatisches Differenzieren als kombinatorisches Problem.* In Beutelspacher, A. (Hrsg), *Jahrbuch Überblick Mathematik 1995.* Vieweg Verlag.

HEUSER, H.: (1990). *Lehrbuch der Analysis.* II. B. G. Teubner Verlag, Stuttgart, 8. Auflage.

JAKUBITH, M.: (1991). *Chemische Verfahrenstechnik.* VCH Verlagsgesellschaft.

KEDEM, G.: (1980). Automatic differentiation of computer programs. *ACM Trans. Math. Software*, 6, S. 150–165.

KNUTH, D. E.: (1973). *The Art of Computer Programming.* I. Addison-Wesley, 2. Auflage.

KÖRKEL, S.: (1999). *VPLAN98–Dokumentation.* in Vorbereitung.

KUD, A.: (1997). *Wärmebilanz einphasiger, idealer Rührkessel.* (Unveröffentlichte) schriftliche Ausarbeitung.

LOHNER, R. J.: (1993). *Verified Computing and Programs in PASCAL-XSC.* Doktorarbeit, Universität Karlsruhe.

MOELWYN-HUGHES, E. A.: (1970). *Physikalische Chemie.* Georg Thieme Verlag, Stuttgart.

MORRISON, R. T., BOYD, R. N.: (1978). *Lehrbuch der organischen Chemie.* Verlag Chemie, New York.

NEWSAM, G. N., RAMSDELL, J. D.: (1983). Estimation of sparse Jacobian matrices. *SIAM journal Alg. Disc. Meth.*, 4, S. 404.

PUKELSHEIM, F.: (1993). *Optimal Designs of Experiments.* Wiley series in probability and mathematical statistics. John Wiley & Sons, New York.

SCHLÖDER, J. P.: (1988). *Numerische Methoden zur Behandlung hochdimensionaler Aufgaben der Parameteridentifizierung.* Doktorarbeit, Universität Bonn.

SPEELPENNING, B.: (1980). *Compiling fast Partial Derivatives of Functions given by Algorithms.* Doktorarbeit, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801.

STOER, J., BULIRSCH, R.: (1971). *Einführung in die Numerische Mathematik*, Band I und II. Springer-Verlag, Heidelberg.

VAUCK, W. R. A., MÜLLER, H. A.: (1978). *Grundoperationen chemischer Verfahrenstechnik.* Verlag Theodor Steinkopff, Dresden.

WALTER, W.: (1991). *Gewöhnliche Differentialgleichungen.* Springer-Verlag, Heidelberg, 5. Auflage.