5 Modellentwicklung und Modelldiskriminierung in Presto-Kinetics

Das Softwarepaket Presto-Kinetics wurde zur Behandlung von chemischen Reaktionssystemen entworfen. Dieses Programmpaket findet (nicht nur) im Bereich der Chemischen Industrie bereits seit 1996 Anwendung und wird stetig weiterentwickelt. Presto-Kinetics wurde insbesondere auch in Bezug auf die Parameterschätzung von der Autorin mitentwickelt und steht ihr sowohl als Modellier- und Rechenwerkzeug als auch in Codeform zur Verfügung. Damit besteht die Möglichkeit, das Overlap-Konzept in eine komplexe und bereits gefestigte Software einzubetten, und damit an anwendungstechnisch relevanten Problem auszutesten (siehe dazu auch [2],[6],[22],[23]). Wir wollen auf die genaue Ausprägung der Oberfläche nicht weiter eingehen, sondern nur aufführen, welche Vorzüge zum Einsatz dieser Software in vielen chemischen Unternehmen weltweit geführt haben:

Darstellung von Reaktionskinetik in reproduzierbarer Form (modulare Reaktionsschritte) In Presto-Kinetics wird ein Modell vollständig modular aufgebaut. Der Benutzer muss sowohl seine Reaktoren (mit Eigenschaften wie Betriebsart, Volumen, Temperatur) als auch die Reaktanden eines Reaktors mit ihren chemischen Eigenschaften (Dichte, Molekulare Masse, Startkonzentration) definiert haben, bevor er seine Reaktionskinetik angibt. Diese wird nicht in Form von Differentialgleichungen für die Reaktanden angegeben (obwohl auch das möglich ist). Stattdessen gibt es ein Vielzahl an implementierten Reaktionsmodulen, welche die grundlegenden Reaktionskinetiken präsentieren. Durch Auswahl der nötigen Reaktionsschritte (auch in Vielfachheit) können durch entsprechende Zuordnung der teilnehmenden Reaktanden beliebig komplexe Reaktionskinetiken dargestellt werden.

Intern wird aus allen vorliegenden Informationen das zugehörige Differentialgleichungssystem erzeugt, wie es in Kapitel 2 bereits im Beispiel beschrieben wurde.

Die Vorteile liegen klar auf der Hand: Das Modell wird nicht als komplettes Differentialgleichungssystem verlangt, was schon bei der Eingabe sehr fehleranfällig ist, sondern lässt sich in chemischer Notation niederschreiben. Das macht ein Modell für den Anwender viel leichter lesbar, und es bleibt über Anwendergenerationen hinweg benutzbar.

Ein weiterer Vorteil, der die Modellierung selber betrifft: jeder chemische Effekt, der in dem Prozess erkannt werden konnte, bleibt in seiner chemischen Darstellung erhalten und geht nicht in einem DGL-System verloren. Gleiches gilt für Reaktionsgeschwindigkeiten: auch diese bleiben einem Vorgang, dem sie zugeordnet werden konnten, zugeordnet.

Offene Modellierumgebung durch benutzerdefinierte Bausteine (Interpreterfunktionalität) Das Interpretermodul erlaubt die Eingabe von benutzerdefinierten Funktionen, die zur Beeinflussung der Reaktionskinetik eingesetzt werden können. Insgesamt ist die Modellierung beliebiger Kinetiken möglich gemacht.

Rechenbarkeit von beliebig komplexen Differentialgleichungssystemen Aufgrund von verschiedenen numerischen Solvern ist Presto-Kinetics in der Lage, verschiedene Typen von DGL-Systemen zu rechnen; insbesondere die implementierte Adaptivität und Schrittweitensteuerung machen die Solver sehr effizient. Eine genauere Beschreibung zu den Solvern findet sich in dem Handbuch [53].

Effektive Parameterschätzung Parameterschätzung (PE) ist eine der wichtigsten Funktionalitäten von Presto-Kinetics. Neben einer Parametervariation (als Hilfsmittel für einen ersten Überblick über das Vermögen der Parameter) wurde eine Boxsuche, ein Simulated Annealing sowie ein gedämpftes Gauss-Newton-Verfahren implementiert. Aufeinanderaufbauend können mit Hilfe dieser Verfahren gute Ergebnisse bei der Bestimmung von optimalen Parametern erzielt werden. Die nicht ableitungsorientierten Verfahren werden meist dazu benutzt, gute Startwerte für das Gauss-Newton-Verfahren zu erhalten. Die theoretischen Grundlagen für diese Verfahren werden im Anhang 18.1 beschrieben, die Implementation des Gauss-Newton-Verfahren müssen wir hier in aller Ausführlichkeit erläutern, da die Erweiterung des Zielfunktionals durch die Einführung des Overlap Auswirkungen auf die Implementierung hat.

Zum Gauss-Newton-Verfahren sowie der Implementation und auftretenden numerischen Schwierigkeiten und deren Lösung liegt eine lange Liste an Literatur vor. Ein aktueller Überblick und eine systematische Herleitung und Begründung ist in der Monographie[7] gegeben. Es ist nicht Ziel dieser Arbeit, alle verschiedenen Ansätze zu diskutieren; stattdessen ist es wichtig, auf die konkrete Implementierung innerhalb von PRESTO-KINETICS einzugehen, die sich seit vielen Jahren in einer Vielzahl an Problemen bewährt hat. Es sei aber in diesem Zusammenhang schon erwähnt, dass es sich dabei um ein affin-kontravariantes, fehlerkontrolliertes, gedämpftes GN-Verfahren handelt.

Daher beschreiben wir zunächst das Gauss-Newton-Verfahren, angewendet auf die übliche Formulierung des Residuums als Abstand zwischen Messwerten und Zustandsgrößen (Residuumsfunktional) und versuchen, wenn sinnvoll, die bekannte theoretische Darstellung mit Aspekten der tatsächlichen Anwendung zu verknüpfen. Im Anschluss beschreiben wir das neue Zielfunktional (Overlapfunktional), welches auch den Overlap berücksichtigt, sowie die Auswirkungen auf die Implementierung. Die wesentlichen Punkte sind dabei die Definition der Zielfunktion und die Bestimmung der Jacobi-Matrix.

5.1 Angewandtes Gauß-Newton-Verfahren für das Residuums-Funktional

Für ein gegebenes Differentialgleichungssystem (abhängig von einem Parametervektor $p \in \mathbb{R}^n$) der Form

$$\frac{d}{dt}y(t,p) = f(y(t),p),\ y_0 = y(t_0), \tag{24}$$

wie es sich in Presto-Kinetics aus der Reaktionskinetik ergibt, werden Messwerte mit

den (numerisch) berechneten Zustandsgrößen $y(t;p) \in \mathbb{R}^m$ bzw. daraus berechneten Größen (Sensoren) $g(y(t;p)) \in \mathbb{R}^m$ verglichen.

Bemerkung 6 *Die Betrachtung von abgeleiteten Größen (Sensoren), die in einer funktionalen Abhängigkeit von Zustandsgrößen gebildet werden können und für die unter Umständen, messtechnisch bedingt, leichter Messdaten zu erheben sind als für die Zustandsgrößen selber, hat keinen Einfluss auf das Verfahren. Im folgenden werden daher der Einfachheit halber nur noch die Zustände erwähnt.*

Gegeben seien Messdatentupel $d \in \mathbb{R}^s, s \leq m$, für alle oder einen Teil der Trajektorien y an bestimmten Messpunkten $t_i, i = 1, \widehat{N}$. Jeder der Messdatenpunkte $d_i^l = d^l(t_i), l = 1, ..s$ unterliege einem Messfehler δd_i^l. Eine Messdatenbeschreibung kann also (bei Vernachlässigung der Messfehler) übersichtlich durch eine Matrix der Dimension $\widehat{N} \times (s+1)$ dargestellt werden und ist in dieser Form auch üblich

$$\begin{pmatrix} t_1 & d_1^1 & d_1^2 & .. & d_1^s \\ t_2 & d_2^1 & & & \\ .. & & & & \\ .. & & & & \end{pmatrix} \in M^{\widehat{N} \times (s+1)}$$

Bemerkung 7 *Sollten für eine/mehrere Zustandsgröße(n) in einem Messpunkt keine Werte vorliegen, so muss dies innerhalb einer Implementation entsprechend berücksichtigt werden. Wir gehen im weiteren davon aus, dass für jede der Zustandsgrößen in jedem angegebenen Messpunkt tatsächlich Daten vorliegen.*

Bei einer least-squares-Minimierung (wie in 3.2.2 diskutiert) verwenden wir für jeden Messpunkt den Abstand

$$D(i,j,p) = \left| \frac{d_i^j - y_j(t_i,p)}{s_i^j} \right|$$

mit d_i^j = Messwert der Zustandsgröße j, $y_j(t_i)$ = (numerisch ermittelter) Wert der Zustandsgröße j und s_i^j ein Skalierungswert zum Zeitpunkt t_i. s_i^j wird z.Bsp. mit $\left|d_i^j\right|$ belegt, um relative Abstände zu berücksichtigen, kann aber auch abweichend davon vorgegeben werden. Möglich ist z.Bsp. die Vorgabe eines $\widehat{s_j}$ für jede Zustandsgröße y_j zur Unterdrückung von Datenrauschen: s_i^j wird dann mit $\max(\widehat{s}_j, \left|d_i^j\right|)$ belegt. $s_i^j \neq 0$ wird vorausgesetzt.

Sei N die Gesamtzahl der Messwerte (es gilt $N = \widehat{N} * s$) und $F^R(p) \in \mathbb{R}^N$ der Vektor, der aus den $D(i,j,p)$ folgendermassen belegt wird:

$$F_k^R(p) = D(i,j,p), i = 1, ..\widehat{N}, j = 1, ..s, k = (j-1)\widehat{N} + i$$

die Einträge von F^R werden also untereinander aus den Abständen bzgl. der Zustände $y_j(p)$ für jeweils alle Messzeitpunkte t_i gebildet. Damit haben wir den Doppelindex

i, j ersetzt durch einen Einfachindex k und definieren das Zielfunktional $\mathbf{F}^R$ für das Gauss-Newton-Verfahren über

$$\mathbf{F}^R = \frac{1}{\sqrt{N}} \left\| F^R(p) \right\| = \sqrt{\frac{1}{N} \sum_k F_k^R(p)^2} \qquad \text{(Residuums-Funktional)}$$

Zur Auswertung dieses Funktionals ist eine (numerische) Lösung von (24) mit Parameter p notwendig.

Notationen Wir verwenden im folgenden daher die Notation

$$F^R(p) = F^R(p; y(p)) \text{ bzw. } F_k^R(p) = F_k^R(p; y(p))$$

um diese Abhängigkeit hervorzuheben. Die Zeitabhängigkeit $y(p) = y(t; p)$ können wir notationsmäßig vernachlässigen, den Zielfunktionalsvektor F^R kürzen wir mit der allgemeineren Notation F ab.

Sei auch weiterhin N die Gesamtzahl der Messwerte, n die Anzahl der Parameter.

Das implementierte gedämpfte Gauß-Newton-Verfahren ist grob in folgende Schritte unterteilt (wir verzichten auf Darstellungen von feineren Aspekten des Verfahrens wie z.Bsp. die Dämpfungsstrategie, wenn sie für unsere Betrachtungen nicht relevant sind):

Gegeben sei ein Startwert $p_0 \in \mathbb{R}^n$ für die Parameter sowie eine Toleranz TOL .

Algorithmus mit Aufwandsbetrachtung

1. Startschritt: Bestimme $F_0 = F(p_0; y(p_0))$
   Aufwand: benötigt eine Integration des Systems in p_0

2. Iterationsvorschrift: Angenommen, das Gauß-Newton-Verfahren befinde sich nun im l-ten Schritt mit aktuellem Parameter $p = p^l$ (der Index l wird nur erwähnt wenn wichtig). Es werden die Ableitungen des Zielfunktionalvektors nach den Parametern benötigt, d.h. die einzelnen Werte

   $$J_{r,k} = \frac{\partial F_r}{\partial p_k}, \forall r, k, r = 1, ..N, k = 1, ..n$$

   müssen bestimmt werden, um die Jacobi-Matrix

   $$J(p) \in \mathbb{R}^{N \times n}$$

   bilden zu können. Diese Werte werden über einen numerischen Differenzenquotienten approximiert. Mit einem geeigneten δp_k wird $F(p + \delta p_k; y(p + \delta p_k))$ berechnet, der Zielfunktionsvektor wird damit jeweils für einen gestörten Parameter $p_k, k = 1, ..n$, ausgewertet. Insgesamt resultiert daraus die Approximation $\tilde{J}(p) \in \mathbb{R}^{N \times n}$ der Jacobi-Matrix $J(p)$ als Spaltenmatrix der

   $$\tilde{J}_k(p) = F_p^k(y(p)) = \frac{F(p + \delta p_k; y(p + \delta p_k)) - F(p; y(p))}{\delta p_k} \in \mathbb{R}^{N \times 1} \qquad (25)$$

δp_k wird relativ auf den aktuellen Wert des Parameters p_k bezogen: $\delta p_k = \Delta \cdot p_k$. Zur Bestimmung von Δ siehe (5.1.3).

Aufwand: Da nur ein einseitiger Differenzenquotient gebildet wird, ist pro Parameter eine Integration von (24) nötig, die nur zur Bestimmung des Differenzenquotienten benutzt wird; $F(p)$ ist aus dem vorangegangenen Schritt bereits vorhanden

3. Korrektur: Die Korrektur der Parameter im l-ten Schritt des Verfahrens wird jetzt bestimmt über den Zusammenhang

$$\tilde{J}(p^l)\Delta p^l = -\lambda_l \cdot F(p^l; y(p^l)) \tag{26}$$

welcher mit Hilfe einer QR-Zerlegung gelöst werden kann (wie es auch in [7] empfohlen wird). Auch wenn die Jacobi-Matrix nicht vollen Rang besitzt, wird eine Lösung der Gleichung gefunden (siehe dazu auch 5.1.4). Das Vorliegen von insensitiven Parametern (Null-Spalte in der Jacobi-Matrix) führt zu einer (gewünschten) Belegung von $\Delta p_i^l = 0$ für die entsprechenden p_i, das Vorliegen von korrelierten Parametern kann aber zu unerwünschten Belegungen, die einen „sehr großen" Schritt in bestimmte Richtungen bedeuten, führen. An dieser Stelle kann das Verfahren der sogenannten Reduzierten Richtungen greifen, welches wir in Ausführlichkeit in Kap. 5.3 beschreiben werden.

Der eingeführte Dämpfungsparameter λ_l ist hier der Vollständigkeit halber aufgeführt; er wird über eine Dämpfungsstrategie gesteuert, die gemäß [7] implementiert ist, und die unabhängig vom Verfahren der Reduzierten Richtungen angewendet wird.

Aufwand: eine QR-Zerlegung (ist zu vernachlässigen)

4. Abbruchtest: Es muss getestet werden, ob der neue Parameter $p^{l+1} = p^l + \Delta p^l$ im Sinne eines (*natürlichen*) *Monotonietests* eine „Verbesserung" bringt. Es wird dazu $F(p^{l+1}; y(p^{l+1}))$ berechnet, was eine erneute Lösung von (24) verlangt. Unterschreitet das Zielfunktional den verlangten Wert TOL für das Residuum $\|F\| < TOL$, so kann das Verfahren beendet werden. Da das Zielfunktional aufgrund seiner hier benutzten Definition mit skalierten Messwerten eine direkte praktische Bedeutung hat, ist eine Wahl von $TOL = 0.01$ typisch. Da diese Anforderung in der Regel selten erreicht wird (dies würde einen relativen, durchschnittlichen Fehler pro Messgröße von 1% erfordern), wird als zweiter Abbruchtest ermittelt, ob der Wert des Zielfunktionals gegenüber dem zuletzt ermittelten Wert tatsächlich vermindert werden konnte $\left\|F(p^{l+1})\right\| < \left\|F(p^l)\right\|$. Gegebenenfalls ist eine (mehrfache) Dämpfung der Parameterkorrektur Δp^l nötig (Reduzierung des Newton-Schrittes in der Länge). Wird schließlich eine Korrektur akzeptiert, geht es zurück zu Schritt 2, anderenfalls wird das Verfahren abgebrochen. Die Konvergenz im Parameterraum wird hier nicht beachtet, da es sich um ein residuen-orientiertes Verfahren handelt.

Aufwand: benötigt eine Integration des Systems in p^{l+1}, die aber direkt für den nächsten Schritt benutzt werden kann

Bemerkung 8 *Das Verfahren in dieser Form berücksichtigt die Messdatenstreuung nur dann, wenn diese bei der Wahl des Skalierungswertes s beachtet wird.*

5.1.1 Güte der Jacobi-Matrix

Die Güte der Jacobi-Matrix ist essentiell für das Verfahren. Ist die Jacobi-Matrix ungenau bestimmt, so wird in (26) ein „falsches" Nullstellenproblem gelöst. Die Jacobi-Matrix sollte daher, wann immer möglich, analytisch bestimmt werden. Dies ist in der Regel und insbesondere in unseren Anwendungen nicht möglich, so dass die numerische Berechnung mit Hilfe eines Differenzenquotienten mit Störung δp_k herangezogen werden muss. Zur bestmöglichen Wahl von δp_k siehe Kap. 5.1.3.

5.1.2 Aufwandsbetrachtung

Aus numerischer und aufwandstechnischer Sicht ist bei diesem Algorithmus wichtig, dass bei der Fortsetzung in Schritt 2 der aktuelle Wert des Zielfunktionals $F(p^{l+1})$ (resultierend aus dem Abbruchtest) schon vorliegt.

Dazu muss man sich folgendes klarmachen: Bei Vorliegen von nur einer Versuchsbeschreibung mit Messdaten (ein sogenanntes *Messdatenblatt*) erfordert die Auswertung von $F(p)$ die Integration des gesamten Differentialgleichungssystems jeweils bis zu den einzelnen Messzeiten. Bei einem komplexen System mit vielen (hunderten) Messzeitpunkten und evtl. noch anderen, aus der Modellbeschreibung resultierenden anzufahrenden Integrationszeitpunkten (Startzeitpunkt eines Feed, Wechsel einer Füllstrategie in bestimmten Zeitpunkten und ähnliches) kann das zeitaufwändig sein, da der Solver seine Schrittweiten entsprechend adaptiv anpassen muss (ein optimaler Schritt, der über einen nächsten anzufahrenden Zeitpunkt hinausgerät, wird entsprechend abgeschnitten) und insbesondere jeden neuen Schritt mit „kleiner" Schrittweite beginnt.

Bei Vorliegen von mehreren Messdatenblättern (es liegen also Versuche mit verschiedenen Bedingungen vor, wobei die Parameter für alle Versuche gleichzeitig gefittet werden müssen) müssen entsprechend viele Integrationen, jeweils initialisiert mit den entsprechenden Versuchsbedingungen, durchgeführt werden, um einen Zielfunktionalsvektor zu belegen.

Die Einführung eines *dense output* zur Bestimmung der simulierten Werte $y_j(t_i, p)$ in den gegebenen t_i ist nicht implementiert. Ein solches Verfahren könnte den Aufwand, den wir zuvor beschrieben haben, reduzieren, indem es ohne zusätzliche Stops das System zunächst mit optimalen Schritten löst und dann aus den berechneten Werten die Werte in den Messzeitpunkten (etwa durch Interpolation) ermittelt. Der erzielte Zeitgewinn muss natürlich mit dem zusätzlichen Aufwand bezüglich der weiteren Auswertungen verglichen werden; eine Abschätzung der erzielten Genauigkeiten der $y_j(t_i, p)$ müsste durchgeführt werden. Erfahrung mit tatsächlichen Anwendungen zeigt aber, dass gerade bei Vorliegen von Messdaten (an bestimmten Messzeitpunkten) diese Messzeitpunkte sinnvollerweise angefahren werden: als gutes Beispiel kann das Temperaturprofil genannt werden. Dabei werden, parallel zu Messdaten, welche die Konzentrationen von Reaktanten betreffen, auch Temperaturdaten erhoben. Diese werden genutzt, um die Simulation des Modells zu steuern (wenn nicht intern eine Wärmebilanz mitgerechnet wird). Damit müssen aber alle Zeitpunkte dieses Temperaturprofils angefahren werden, so dass direkt alle simulierten Werte in diesen Punkten abgegriffen werden können und ein dense output überflüssig machen.

5.1.3 Wahl von Δ

Die Qualität der Richtungsableitung ($\delta p_k = \Delta \cdot p_k$) hängt wesentlich von der Wahl des Δ ab. Bei einem ODE-Solver, der mit Genauigkeit *eps* die Lösung des DGL-Systems (24) bestimmt, ist die Wahl von $\Delta = \sqrt{eps}$ optimal, ist aber oft zu groß für eine „gute" Richtungsableitung. Der Grund ist, dass wir hier mit Genauigkeiten von *eps* $= 1e-2$ bis $1e-5$ (sogenannten technischen Genauigkeiten) arbeiten, was zu Parameterstörungen für den Differenzquotienten von bis zu 10% führen kann. Die in der Literatur [17] beschriebene automatische Differentiation, die auf einer Differentiation aller Terme direkt im Programmcode basiert, kann hier aufgrund der modularen Struktur des Programms und der möglichen Anbindung von externen Bibliotheken nicht verwendet werden. Dieses Vorgehen bietet sich eher für „klassische" ODEs mit einheitlichen, zusammengefassten rechten Seiten an. Um trotzdem eine größere Genauigkeit und gleichzeitig eine effiziente Berechnung der Ableitungen zu erreichen, verwenden wir hier eine interne numerische Differentiation IND. In der ursprünglichen Formulierung [5] wird dazu die gesamte Diskretisierung in Abhängigkeit von einem Parametervektor p formuliert und auch schon nach p differenziert implementiert. Im Falle einer adaptiven Schrittweitensteuerung (für feste Schrittweiten entspricht die IND der konventionellen Ableitung) kann dann in jedem Zeitschritt die Variation nach den Parametern mit einheitlicher Schrittweite direkt mitintegriert werden. Somit hat man am Ende der numerischen Integration direkt das Diskretisierungsverfahren als eine Gesamtauswertung einer Funktion mit (etwa) Maschinengenauigkeit *epsmach* differenziert – im Gegensatz zur numerischen Differentiation des gesamten adaptiven Lösungsprozesses, der nur mit Genauigkeit *eps* möglich ist.

Der Nachteil der so durchgeführten IND besteht allerdings in einer Vergrößerung des Systems und der – wenn nicht formalen, so doch praktischen – Einschränkung auf ODEs. Daher gibt es in Presto-Kinetics eine alternative Variante der IND. Dabei wird während der adaptiven Integration des Systems mit Parameter p die Schrittweitensteuerung (Zeitgitter)und gegebenenfalls auch Ortsgitter für partielle Differentialgleichungen (PDEs) gespeichert und für die gestörten Parameter $p + dp$ die folgenden

Integrationen jeweils entlang dieser Gitter ausgeführt. Damit wird der erhöhte Speicheraufwand der originalen (parallen) IND durch einen sequentiellen Prozess vermieden. Der Aufwand insgesamt ist aber kaum größer. Als einzige Einschränkung ist der Fall zu nennen, bei dem während der Basisintegration Schrittweiten in der Größenordnung TOL auftreten. Dann kann die IND zu falschen (zu großen) Ableitungen führen, was sich in extremer Dämpfung des Gauss–Newton-Verfahrens zeigt. Die arithmetische Genauigkeit unter Windows mit C++ liegt bei etwa 5e-18, so dass ein Δ von $1e-8$ sinnvoll und bewährt ist.

5.1.4 Bestimmung der Δp

Die Lösung Δp des Problems (N :Anzahl Messungen, n Anzahl Parameter)

$$\tilde{J}(p)\Delta p = -\lambda \cdot F(p; y(p)), \tilde{J}(p), F \in \mathbb{R}^{N\times n}, p \in \mathbb{R}^n, \Delta p \in \mathbb{R}^n, y(p) \in \mathbb{R}^N \tag{27}$$

läßt sich formal bestimmen zu

$$\Delta p = -\lambda \cdot \tilde{J}(p)^+ \cdot F(p; y(p)), \tilde{J}(p), F \in \mathbb{R}^{N\times n} \tag{28}$$

wobei die Matrix $\tilde{J}(p)^+$ die Pseudoinverse der Matrix $\tilde{J}(p)$ darstellt. Im Falle von $N = n$ und unter der Annahme, $\tilde{J}(p)$ besitze vollen Rang, entspricht die Pseudoinverse der Inversen von $\tilde{J}(p)$. Beide Voraussetzungen sind in den Anwendungen in der Regel nicht erfüllt (es liegen meist sehr viel mehr Messdaten als Parameter vor und die Unabhängigkeit der Parameter ist nicht immer gegeben), so dass wir von einer nicht-quadratischen Matrix $\tilde{J}(p)$ bzw. $\tilde{J}(p)^+$ mit nicht-vollem Rang ausgehen müssen, und die Lösung von (27) nicht mehr eindeutig sein muss. Nach [7] (Kap. 4.1) gibt es drei grundsätzliche Ansätze für die (numerische) Lösung von (27), von denen eine die QR-Zerlegung darstellt. Die numerischen Routinen für QR-Zerlegung und Konditionsanalyse werden hier mit einer C++-Implementierung gemäß [10] durchgeführt (enthalten im TIDE-Paket). Im Zusammenhang mit den reduzierten Richtungen wurden die notwendigen LAPACK-Routinen angebunden.

5.2 Gauß-Newton-Verfahren für das Overlap-Funktional

Seien Differentialgleichungssystem und Messwerte wie in (5.1) gegeben. In jedem Messpunkt wollen wir nun nicht mehr nur den Abstand zwischen Messwert und simuliertem Wert betrachten, sondern zusätzlich die erzeugte Überlappung zwischen Messdatenwahrscheinlichkeitsdichteverteilung und Wahrscheinlichkeitsdichteverteilung des Modells in diesem Messpunkt betrachten, wie wir sie im Overlap (22) definiert haben. Ebenso, wie oben der Vektor $F^R(p) \in \mathbb{R}^N$ belegt wurde, wird nun zusätzlich ein Vektor $F^O(p, \sigma_p) \in \mathbb{R}^N$ mit Einträgen aus den aus den Messabweichungen ermittelten Overlaps definiert, der neben p auch den Parameter σ_p beachtet.

Fasst man nun die Parameter zu $\widehat{p} = (p, \sigma_p)$ zusammen und erweitert den Zustandsraum zu $\widehat{y} = (y, \frac{\partial y}{\partial p})$, so kann man das Gauss-Newton-Verfahren aus Kap. 5.1 formal

anwenden auf

$$\widehat{y}' = \hat{f}(\widehat{y}; \widehat{p})$$

mit entsprechend komplizierterer Zielfunktion. Der oben skizzierte Algorithmus soll so weit wie möglich benutzt werden, da er insbesondere eine Strategie zur Dämpfung und Abbruchkriterien beinhaltet. Die Effektivität des Algorithmus ist über viele Jahre der Anwendung gesichert. Eine Voraussetzung für die Anwendung ist aber, wie oben schon erwähnt, eine möglichst hohe Güte der Jacobi-Matrix, die wir im folgenden vorrangig beachten wollen.

Notation Zur Unterscheidung der Verfahren bzw. der Jacobi-Matrizen werden wir das bisher beschriebene Gauss-Newton-Verfahren als das Standardverfahren bezeichnen.

5.2.1 Definition der Zielfunktion

(Wir ersetzen hier die Notation des Overlaps in einem Messpunkt F_O.aus Kapitel 4, Definition 3 durch f_O,damit die Definition von $F^O(p, \sigma_p)$ eindeutig ist) Der Overlap f_O ist definitionsgemäß ein Wert kleiner als 1; damit er möglichst groß wird, sollte sein Abstand von 1 minimiert werden. Daher wird in jedem Messpunkt t_i für jeden Zustand y_j der Wert $1 - f_O$ betrachtet . Die Einträge von $F^O(p, \sigma_p)$ bestimmen sich dann aus allen vorliegenden f_O.

Im folgenden seien die $\sigma_p \in \mathbb{R}^n$ die zu bestimmenden Varianzen der Parameter p, und α ein Gewichtungsfaktor; der mit F^R gekennzeichnete Vektor entspricht dem bisherigen Zielfunktionalvektor (Residuums-Funktional). Man kann nun auf verschiedene Arten ein erweitertes Zielfunktional erzeugen:

1. In jedem Messpunkt wird die Summe aus Abstand und Overlap betrachtet:

   $$F(p, \sigma_p; y(p), \frac{\partial y}{\partial p}(p)) := (\alpha F^R(p; y(t)) + (1 - F^O(p, \sigma_p; y(p), \frac{\partial y}{\partial p}(p))))^T, 1 = (1, ..1) \in \mathbb{N}^N$$

   Dies beschreibt einen N-dimensionalen Vektor, in dem jede Komponente k ihren Wert aus Abstand F_k^R und Overlap F_k^Oadditiv ermittelt, oder

2. In jedem Messpunkt werden diese Beiträge getrennt betrachtet:

   $$F(p, \sigma_p; y(p), \frac{\partial y}{\partial p}(p)) := (\alpha F^R(p; y(t)), \quad (1 - F^O(p, \sigma_p; y(p), \frac{\partial y}{\partial p}(p))))^T, 1 = (1, ..1) \in \mathbb{N}^N$$

was einen $2N$-dimensionalen Vektor darstellt. Wir haben uns (schon aus Skalierungsgründen) für die zweite Definition entschieden, und definieren als neues Zielfunktional das Overlap-Funktional $\mathbf{F}^O$

$$\mathbf{F}^O = \frac{1}{\sqrt{2 \cdot N}} \|F(p)\| = \sqrt{\frac{1}{2 \cdot N} \sum_{n=1}^{N} (\alpha \cdot F_n^R(p))^2 + (1 - F_n^O(p, \sigma_p))^2} \qquad (29)$$

Abhängigkeiten Wir können den Anteil des Overlaps, wie er in (23) eingeführt wurde, für einen Index $n = 1, ..N$ in folgender Darstellung anwendungsbezogen präsentieren:

$$F_n^O = \sqrt{\frac{2\sigma_{mess,n}\sigma_{sim,n}}{\sigma_{mess,n}^2 + \sigma_{sim,n}^2}} \exp(-\frac{\mu_{mess,n} - \mu_{sim,n}}{2(\sigma_{mess,n}^2 + \sigma_{sim,n}^2)}). \tag{30}$$

Wir sprechen dabei von $\sigma_{mess,n}$ als der gegebenen Varianz des Messwertes, von $\mu_{mess,n}$ als dem (Erwartungswert des) Messwert (sie entsprechen den bisher betrachteten d_i), von $\sigma_{sim,n}$ bzw. $\mu_{sim,n}$ als den durch lineare Propagation von σ_p erhaltenen Varianz bzw. Erwartungswerten des Modells. Da wir innerhalb des Gauss-Newton-Verfahrens die Ableitung des Zielfunktionals nach allen Parametern ermitteln müssen, müssen wir aber genauer betrachten:

$$\mu_{sim,n} = \mu_{sim,n}(p; y(p)),$$

sowie

$$\sigma_{sim,n} = \sigma_{sim,n}(p, \sigma_p; \frac{\partial y}{\partial p}(p)). \tag{31}$$

Die Abhängigkeit (31) impliziert, dass bereits zur Auswertung des Overlap-Funktionals $\mathbf{F}^O$ Informationen $\frac{\partial y}{\partial p}$benötigt werden, die im oben beschriebenen Standard-Verfahren erst bei der Aufstellung der Jacobi-Matrix notwendig werden. Das bedeutet, dass das Verfahren in der oben beschriebenen Form nicht direkt anwendbar ist, und die Aufstellung der nun benötigten Jacobi-Matrix die Aufstellung der Hesse-Matrix des Standard-Verfahrens benötigen wird.

5.2.2 Aufstellung der Jacobi-Matrix im Detail

Wie wir schon im Standardverfahren erwähnt haben, ist die Güte der Jacobi-Matrix J^O von entscheidender Bedeutung für das Verfahren. Daher müssen wir bei der Aufstellung sehr sorgfältig vorgehen und soweit wie möglich analytische Untersuchungen heranziehen. Dies haben wir im nachfolgenden Kapitel (6) getan. Sämtliche Ableitungen, die im folgenden benutzt werden, sind dort hergeleitet.

Die aufzustellende Jacobi-Matrix hat (unter Vernachlässigung des Gewichtungsfaktors α) die Form

$$J^O = \begin{pmatrix} \frac{\partial F^R}{\partial p} & \frac{\partial F^R}{\partial \sigma_p} \\ \frac{\partial F^O}{\partial p} & \frac{\partial F^O}{\partial \sigma_p} \end{pmatrix} \in M_{2N \times 2n}$$

1. Quadrant $\frac{\partial F^R}{\partial p} \in M_{N \times N}$: Dies entspricht der Jacobi-Matrix des Standard-Verfahrens, benötigt also die Berechnung der $\frac{\partial y}{\partial p} = \frac{\partial \mu_{sim}}{\partial p}$. Die Güte der Daten entspricht der bisher erzielten Güte durch das Standardverfahren.

2. Quadrant $\frac{\partial F^R}{\partial \sigma_p} \in M_{N \times N}$: ist identisch 0, da es keine Abhängigkeit der Zustände $y(t, p)$ von den σ_p gibt.

3. Quadrant $\frac{\partial F^O}{\partial p} \in M_{N\times N}$: Hier müssen wir vollständig betrachten

$$\frac{\partial F_n^O}{\partial p} = \frac{\partial F_n^O}{\partial \mu_{sim,n}} \cdot \frac{\partial \mu_{sim,n}}{\partial p} + \frac{\partial F_n^O}{\partial \sigma_{sim,n}} \cdot \frac{\partial \sigma_{sim,n}}{\partial p}, n = 1, ..N, \tag{32}$$

da sich der Overlap-Anteil bei Variation der Parameter sowohl über die Erwartungswerte $\mu_{sim,n}$ als auch über die $\sigma_{sim,n}$, die wiederum von den $\frac{\partial y}{\partial p}$ abhängen, ändert.

Auswertung Der erste Summand ergibt sich aus der analytischen Ableitung von (30) nach $\mu_{sim,n}$ (siehe (39)) und den schon berechneten Größen für die Jacobi-Matrix des Standardverfahrens. Für den zweiten Summanden muss man (ebenfalls analytisch) die Ableitung von (30) nach $\sigma_{sim,n}$ bestimmen (siehe (40)); zusätzlich ist die Änderung der Varianzen im Zustandsraum in Abhängigkeit von den Parametern zu betrachten. Es zeigt sich (42), dass man die zweite Ableitung der rechten Seite des Differentialgleichungssystems (24) (Hesse Matrix) benötigt.

Illustration Bei einer Änderung des Systems in einem Parameter p zu $p + \delta p$ sind die Zustände $y(t, p)$ betroffen; es ändert sich direkt der Erwartungswert $\mu_{sim,n}$ jedes simulierten Wertes (in folgender Abbildung 2 z.Bsp. von 30 auf 40).

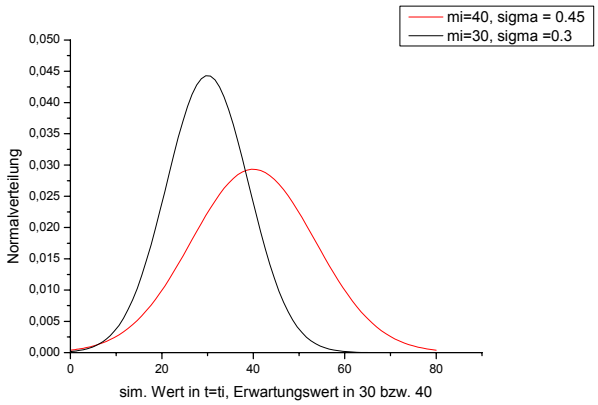

Abbildung 2: Änderung von p zu $p + \delta p$ führt zu einer Verschiebung und Verformung der Wahrscheinlichkeitsdichteverteilung des Modells in einem Punkt t

Von einer Änderung in p ist aber ebenso das $\sigma_{sim,n}$ (also die Form der Normalverteilung) betroffen. Der Overlap mit einer durch die Messpunktangaben fest definierten Kurve ändert sich also entsprechend:

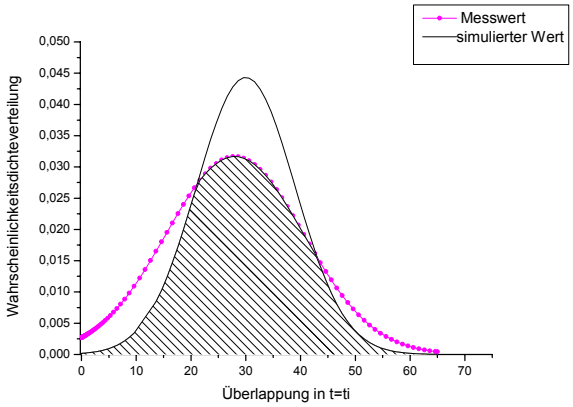

Abbildung 2a: Überlappung im Erwartungwert 30

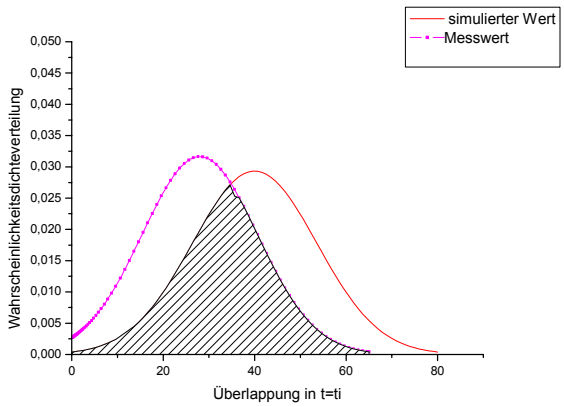

Abbildung 2b: Überlappung im Erwartungswert 40

Verschiedene Wahrscheinlichkeitsdichteverteilungen in einem Messzeitpunkt t_i (Abbildung 2a, 2b) bilden mit einer Messdatenverteilung (Strich-Punkt-Linie) verschiedene Überlappungen.

4. Quadrant $\frac{\partial F^O}{\partial \sigma_p}$: Da die σ_p nur im Zusammenhang mit der Propagation der Varianzen relevant sind, erhält man:

$$\frac{\partial F_n^O}{\partial \sigma_p} = \frac{\partial F_n^O}{\partial \sigma_{sim,n}} \cdot \frac{\partial \sigma_{sim,n}}{\partial \sigma_p} \tag{33}$$

Auswertung Wie beim 3. Quadranten muss man die Ableitung von (30) nach $\sigma_{sim,n}$ bestimmen. Die zweite partielle Ableitung $\frac{\partial \sigma_{sim,n}}{\partial \sigma_p}$ kann aus der Darstellung (41) bestimmt werden. Dieser Anteil kann vollständig analytisch bestimmt werden und hat keinen negativen Einfluss auf die Genauigkeit des Verfahrens.

Illustration Bei einer Änderung von σ_p zu $\sigma_p + \Delta\sigma_p$ bei festem Parameter p bleibt der Erwartungswert μ_{sim} des simulierten Wertes gleich, denn die Zustandsgleichungen sind von dieser Änderung nicht betroffen. Da sich aber die Form der Kurve (Wahrscheinlichkeitsdichteverteilung im Parameter p) ändert, ändert sich auch die Form der Wahrscheinlichkeitsdichteverteilung im Zustandsraum (siehe Abbildung 3), so dass sich insgesamt der Overlap verändert.

Zusammenfassung: Güte der Jacobi-Matrix Die Aufstellung der Jacobi-Matrix ergibt sich in den 4 Blöcken, die wir gerade beschrieben haben, zu einer Blockmatrix der Gestalt $\hat{J}(p^l, \sigma_p^l) = \begin{pmatrix} \frac{\partial F^R}{\partial p} & 0 \\ \frac{\partial F^O}{\partial \sigma_p} & \frac{\partial F^O}{\partial \sigma_p} \end{pmatrix}$, wobei $\frac{\partial F^O}{\partial \sigma_p}$ nur aufwändig vollständig zu berechnen ist. Ein großer Anteil kann analytisch exakt bestimmt werden, es gibt aber Ungenauigkeiten durch

1. die Berechnung der Jacobi-Matrix des Standardverfahrens $\frac{\partial F^R}{\partial p}$ und

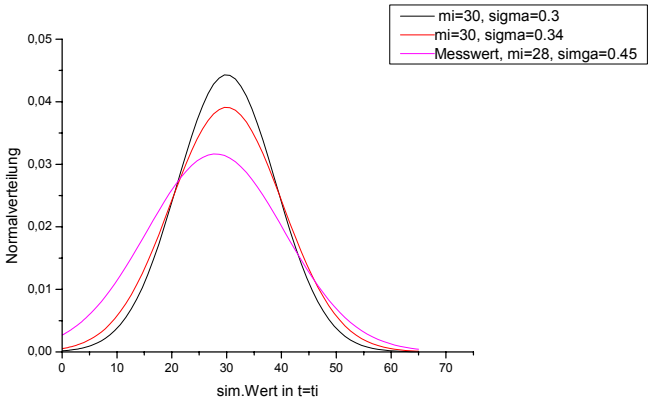

Abbildung 3: Wahrscheinlichkeitsdichteverteilungen in einem Messzeitpunkt ti, die sich in der Form unterscheiden, bilden mit der Messdatenverteilung verschiedene Überlappungen.

2. durch die Berechnung der Hesse-Matrix des Standardverfahrens,

jeweils über Differenzenquotienten. Diese Ungenauigkeiten, die in einer in diesem Projekt noch nicht abgeschätzten Größenordnung liegen, sind in Kauf zu nehmen.

5.2.3 Algorithmus und Aufwandsbetrachtung

Unter dem Gesichtspunkt der Einbettung in das zuvor beschriebene ausgefeilte Verfahren schlagen wir folgenden Algorithmus vor:

1. Startschritt: Bestimme $F_0 := \alpha F^R(p_0, y(t))$ wie im Standardverfahren, d.h. der Wert des Overlaps wird nicht ausgerechnet. Da davon auszugehen ist, dass der Overlap-Anteil erst bei fortgeschrittener Konvergenz wichtig wird, ist dies zulässig.

   Aufwand: benötigt eine Integration des Systems in p_0

   Einbettung: entspricht im Wesentlichen dem Startschritt des Standardverfahrens

2. Iterationsvorschrift: Angenommen, das Gauß-Newton-Verfahren befinde sich nun im l-ten Schritt mit aktuellem Parameter $\widehat{p} = (p^l, \sigma_p^l)$. Die Jacobi-Matrix $J(\widehat{p}^l)$ wird als Blockmatrix, wie oben beschrieben, erstellt. Sie benötigt nicht nur die Jacobi-Matrix des Standardverfahrens $J(p^l)$, sondern auch die Hesse-Matrix $H(p^l)$, die ebenfalls über Differenzenquotienten (mit einem vom aktuellen δp_k zur Bestimmung der Jacobi-Matrix abweichenden δp_k^H) ermittelt wird.

$$H(p^l) = \frac{J(p^l + \delta p_k^H) - J(p^l)}{\delta p_k}$$

Aufwand: Da für die Jacobi-Matrix des Standardverfahrens $J(p^l)$ nur ein einseitiger Differenzenquotient gebildet wird, ist pro Parameter p eine Integration von (24) nötig, die nur zur Bestimmung des Differenzenquotienten benutzt wird; $F^R(p^l)$ ist bereits vorhanden. Für die Bildung sämtlicher Differenzenquotienten für die Hesse-Matrix werden dann weitere $n \cdot n$ Integrationen durchgeführt.

Einbettung: Die Bildung der Jacobi-Matrix des Standardverfahrens entspricht dem Standardverfahren; gleichzeitig kann diese Matrix bei der Bildung der Differenzenquotienten für die Hesse-Matrix benutzt werden.

3. Korrektur: Bei der Korrektur der Parameter können wir eine (inexakte) Jacobi-Matrix bzgl. p und σ_p sowie eine aktuell ausgewertete Zielfunktion verwenden:

$$(\Delta p^{l+1}, \Delta\sigma^{l+1}) = -\lambda_l \tilde{J}(p^l, \sigma^l)^{-1} \cdot ((\alpha F^R(p^l; y(p^l)), (1-F^O(p^l, \sigma_p^l; y(p^l), \frac{\partial y}{\partial p}(p^l))))^T \tag{34}$$

(Alternativ könnte man ein Verfahren gestalten, bei dem die Auswertung der Zielfunktion im letzten Schritt genutzt werden muss und nicht die aktuelle Auswertung; das aber führt zu Ungenauigkeiten im Verfahren, die wir hier ausschliessen können.)

Aufwand: benötigt eine QR-Zerlegung in doppelter Dimension

Einbettung: entspricht der Korrektur des Standardverfahrens; wird das Verfahren der Reduzierten Richtungen benutzt, so kann es ohne Änderungen an dieser Stelle greifen.

4. Abbruchtest: Der Monotonietest wird wie im Standardverfahren ausgewertet; dazu wird $F^O(p^{l+1}, \sigma_p^{l+1}; y(p^{l+1}), \frac{\partial y}{\partial p}(p^{l+1}))$ ausgewertet, was im nächsten Schritt benutzt werden kann.

Aufwand: benötigt eine Integration des Systems in p^{l+1}, die aber direkt für den nächsten Schritt benutzt werden kann

Einbettung: entspricht dem Monotonietest des Standardverfahrens

Aufwandsbetrachtung Die Bestimmung der Hesse-Matrix des Standardverfahrens ist hier der aufwändigste Teil. Es wurde neben der Auswertung dieser Matrix in jedem Schritt ein Broyden-Verfahren implementiert, welches den Aufwand der Auswertung reduziert. Der Einfluss dieses Verfahrens auf die Güte der Gesamtmatrix konnte aber noch nicht abgeschätzt werden, so dass die bisherigen Probleme mit voller Hesse-Matrix durchgeführt wurden. Wir gehen daher nicht weiter auf die Implementierung des Broyden-Verfahrens ein.

Fazit: Es ist gelungen, ein Referenzverfahren für die Auswertung und Optimierung des Overlapfunktionals zu implementieren, welches alle Eigenschaften des bisherigen (Standard)Verfahrens übernommen hat. Jetzt ist es möglich, weitere Strategien zur Aufwandsminimierung zu implementieren und zu bewerten.

5.3 Reduzierte Richtungen als Hilfsmittel

Erfahrungen mit dem Gauss-Newton-Algorithmus zeigen, dass es zu Schwierigkeiten aufgrund korrelierter Parameter kommen kann. Bei direkt korrelierten Parametern ist anschaulich klar, dass es keine eindeutige Lösung des Problems geben kann. Innerhalb des Gauss-Newton-Verfahrens wird die aufzustellende Jacobi-Matrix nicht mehr maximalen Rang haben. Damit ist sie aber nicht mehr invertierbar, so dass eine Grundvoraussetzung für Gutartigkeit verletzt ist. Bei der Einführung von Varianz-Parametern σ_p in ein von Parametern p abhängendes System und eine Beschreibung der Zielfunktion in Abhängigkeit von p und σ_p, wie wir es mit dem Overlap-Funktional getan haben, müssen wir erwarten, dass (unabhängig von den bereits vorliegenden Korrelationen) weitere Korrelationen auftreten werden. Daher wurde ein zusätzliches Verfahren implementiert, welches innerhalb des Gauss-Newton-Algorithmus in jedem Iterationsschritt angewendet wird, wenn es um die Bestimmung der Parameteränderungen Δp^k geht (siehe (26)). Dieses Verfahren ist auch für das Gauss-Newton-Verfahren, angewendet auf das gewöhnliche Residuums-Zielfunktional, ausserordentlich wichtig; gerade in der tatsächlichen Anwendung ist es so, dass immer wieder, und gerade zu Beginn einer Modellentwicklung und Modelluntersuchung, durch ein zuviel an Systembeschreibung eine direkte oder auch versteckte Korrelation von Parametern eingeschleust wird. Das erste Anpassen von Parametern über eine Parameterschätzung erweist sich damit immer als schwierig. Insgesamt können wir dieses Verfahren direkt, d.h. ohne Fallunterscheidung (ob F^R oder F^O), in den Algorithmus integrieren, wie wir ihn in Kap. 5.1 beschrieben haben.

Wir wollen das Verfahren der „Reduzierten Richtungen" hier darstellen, zugleich aber auch auf das Dokument [44] verweisen, in dem die theoretischen Grundlagen sehr viel genauer dargestellt sind. Beispiele aus der realen Anwendung wurden innerhalb des Projektes bearbeitet und vorgestellt; sie sind in [47] aufgeführt.

Die Idee Die Idee des „Verfahrens der reduzierten Richtungen" ist es, zunächst die Anzahl essentieller Richtungen eines Systems, in unserem Fall des durch die Jacobi-Matrix dargestellten Systems, mit Hilfe einer Singulärwertzerlegung und eines darauf aufsetzenden Kriteriums festzustellen. Ist die Anzahl essentieller Richtungen kleiner als die Anzahl betrachteter Parameter (wir betrachten dies als Kriterium für das Vorliegen von korrelierten Parametern) soll das linearisierte Minimierungsproblem (36) auf einen niedriger dimensionierten Raum projiziert werden, innerhalb dessen die dann auftretenden Parameter unkorreliert sind. Ein in diesem Raum gefundener Lösungsvektor kann dann in den ursprünglichen Raum zurücktransformiert werden.

Singulärwertzerlegung (Siehe auch [10], [16], [45]) Die Singulärwerte s_i einer Matrix $A \in M_{N\times n}$ sind die positiven Wurzeln der Eigenwerte der Matrix $A^T A \in M_{n\times n}$. Als Singulärwertzerlegung einer Matrix A bezeichnet man das Auffinden von unitären Matrizen $U \in M_{N\times N}$ und $V \in M_{n\times n}$, mit deren Hilfe eine Äquivalenztransformation von A auf Diagonalform möglich ist:

$$U^T AV = S := diag(s_1, s_2, ..s_k) \in M_{N\times n}, k := \min\{N, n\}$$

so dass die Diagonalwerte von S folgende Ungleichung erfüllen:

$$s_1 \geqslant s_2 \geqslant s_3 ... \geqslant s_r \geqslant s_{r+1} = ... = s_k = 0, \text{ mit } r = rang(A)$$

Bemerkung: Eine unitäre Matrix ist eine (komplexe) quadratische Matrix, deren Spalten zueinander orthonormal sind.

Singulärwertzerlegung und Lösung des Minimierungsproblems bei vollem Rang Für das ursprüngliche Minimierungsproblem

$$\|F(p)\| = \min \tag{35}$$

mit $F = (f_j), j = 1, ..N$, und $f_j = \frac{d_i - \Phi(t_i,p)}{\delta d_j}, p \in \mathbb{R}^n$. Es gelte $N > n$.

sei $J \in \mathbb{R}^{N \times n}$ die Jacobi-Matrix

$$J(p) = \frac{\delta F}{\delta p}(p)$$

Das linearisierte Problem lautet

$$\left\| J(p^k)\Delta p^k + F(p^k) \right\| = \min, \quad p^{k+1} = p^k + \Delta p^k \tag{36}$$

Es gilt bei Anwendung der Singulärwertzerlegung

$$J = USV^T \tag{37}$$

wobei $U \in \mathbb{R}^{N \times N}$ sowie $V \in \mathbb{R}^{n \times n}$ orthogonale Matrizen sind, $S \in \mathbb{R}^{N \times n}$ eine Diagonalmatrix mit den Singulärwerten, sortiert nach Größe, auf der Diagonalen. Die Matrix S hat denselben Rang wie J. Hat J den vollen Rang n, so kann die Lösung des Problems (36) mit Hilfe der Singulärwerte und aufgrund der Orthogonalität der Matrizen nach einigen Umformungen direkt angegeben werden (siehe [16]):

$$\begin{aligned} \left\| USV^T\Delta p - b \right\| &= \min \\ &\iff \left\| SV^T\Delta p - U^T b \right\| = \min \\ &\iff \left\| Sy - U^T b \right\| = \min \end{aligned}$$

wobei $b = F(p)$ sowie $V^T\Delta p = y \in \mathbb{R}^n$ gesetzt wurde. $y = (y_i)_{i=1,..n}$ kann aus dieser Darstellung direkt angegeben werden zu

$$y_i = \frac{1}{s_i} U_i^T b, s_i \neq 0$$

wobei die U_i die Spalten der Matrix U darstellen. Die Relation $n < N$ kann in unseren Beispielen vorausgesetzt werden.

Aus $V^T \Delta p = y$ bestimmen sich die Komponenten des Lösungsvektor Δp zu

$$(\Delta p)_l = \sum_{i=1}^{n} V_{li} \frac{1}{s_i} U_i^T b, l = 1, ..n \tag{38}$$

wobei nun die V_l die Spalten der Matrix V darstellen bzw. V_{li} das i-te Element der l-ten Spalte ist.

Eine QR-Zerlegung (wie in unserem Algorithmus erwähnt) muss hier nicht herangezogen werden.

Lösung des Minimierungsproblems bei nicht-vollem Rang Man kann zeigen, dass im Falle einer singulären Jacobi-Matrix die Singulärwertzerlegung dennoch eine Lösung Δp liefert, welche im Sinne einer least-squares-Minimierung, wie wir sie hier durchführen, optimal ist. In der Anwendung ist es aber häufig so, dass zwar die Jacobi-Matrix im mathematischen Sinn vollen Rang besitzt, aber die numerischen Werte so klein sind, dass sie praktisch als Null betrachtet werden können. In diesem Fall gelten die Resultate ihrer Singulärwertzerlegung weiterhin, aber der resultierende Lösungsvektor Δp aus (38) weist sehr große Komponenten auf. Dieses Verhalten führt uns direkt zum Begriff der Kondition:

Kondition Die Kondition der Matrix J ist bestimmt durch das Verhältnis des größten zum kleinsten Singulärwert:

$$cond_2(J) = \|J\| \left\|J^{-1}\right\| = \frac{s_{\max}}{s_{\min}}$$

Eine singuläre Matrix weist einen Konditionswert von ∞ auf, eine „schlecht-konditionierte" Matrix besitzt einen Konditionswert, der im Kehrwert etwa in der Größenordnung der Maschinengenauigkeit liegt.

Im Zusammenhang mit dem Gauss-Newton-Schritt bedeutet eine schlechte Kondition aufgrund der auftretenden großen Komponenten des Lösungsvektors einen unerwünscht großen Schritt in eine bestimmte Richtung. Ein Dämpfungsverfahren muss diese Richtungen dann reduzieren. Erfahrung mit vielen Parameterschätz-Beispielen aus den verschiedensten Bereichen der Chemie belegen, dass Jacobi-Matrizen mit einem Konditionswert kleiner als 100 noch ein gut lösbares Problem darstellen. Werte oberhalb von 1000 signalisieren nahezu immer Korrelationen in den Parametern. Als Konsequenz wäre wünschenswert, dass der kleinste Singulärwert, der betrachtet werden muss, einen Wert größer als $s_{\max}/100$ besitzt.

Essentielle Richtungen Um das Problem der korrelierten Parameter zu lösen und damit auch die numerischen Probleme mit „kleinen" Zahlen zu vermeiden, definieren wir die essentiellen Richtungen einer Matrix (in unserem Fall der Jacobi-Matrix) als die Richtungen, die unter einer Singulärwertzerlegung zu „großen" Singulärwerten gehören, in der Hoffnung, dass große Singulärwerte mehr Charakteristik der Matrix

tragen als kleine Singulärwerte, und eine Vernachlässigung der kleinen Singulärwerte keinen nennenswerten Informationsverlust bedeutet. Untersuchungen haben gezeigt, dass man in manchen Problemen in der Liste der Singulärwerte $(s_i)_{i=1,..k}$ tatsächlich einen eindeutigen Index feststellen kann, ab dem die Werte der Singulärwerte nahezu Null werden oder aber signifikant kleiner sind als die größeren Singulärwerte. Diesen Index bezeichnen wir als den *gap* und würden ihn gerne zur Festlegung der Anzahl essentieller Richtungen heranziehen. Die Bestimmung dieses Index ist aber nicht immer ohne Aufwand möglich bzw. in manchen Problemen läßt sich ein solcher gap nicht finden (Beispiele dazu sind auch in [30] zu finden). Mit Blick auf die Kondition der Jacobi-Matrix bestimmen wir daher für ein gegebenes Problem die Anzahl essentieller Richtungen *willkürlich* so, dass ein Konditionswert von etwa 100 unterschritten wird, d.h. i_{ess} ist derjenige Index, für den gilt: $\frac{s_{\max}}{s_i} \leq 100$ für alle $i \leq i_{ess}$.

Projektion des Raumes Mit der gefundenen Anzahl i_{ess} essentieller Richtungen wird das ursprüngliche Problem (36) in einen Raum projiziert, in welchem die dann auftretenden Parameter unkorreliert sind. Die Projektionsmatrix hat die Form

$$P = \begin{Bmatrix} 1 & & & 0 \\ & 1 & & \\ & & .. & \\ & & 0 & \\ 0 & & & 0 \end{Bmatrix} \in \mathbb{R}^{n \times n}$$

wobei die 1 auf der Diagonalen bis zum Index i_{ess} eingetragen ist. Wir bilden die Lösung

$$\begin{aligned} \Delta p^{k+1} &= -(U(PSV^T))^{-1}F \\ &= -(PSV^T)^{-1}UF \\ &= -(V^T)^{-1}PS^{-1}UF \end{aligned}$$

wobei im letzten Schritt eine QR-Zerlegung die Invertierung von V^T vermeidet. Δp^{k+1}stellt den Lösungsvektor im ursprünglichen Raum dar.

Projektion des Lösungsvektors Alternativ zur Projektion des Raumes und im Sinne der bereits existierenden Software haben wir einen weiteren Ansatz implementiert: Dabei wurde zunächst im k-ten Schritt des Gauss-Newton-Verfahrens der Lösungsvektor Δp^{k+1} über eine QR-Zerlegung bestimmt. Darüberhinaus wurde die Anzahl essentieller Richtungen i_{ess} wie oben mit Hilfe einer Singulärwertzerlegung festgelegt. Der Lösungsvektor Δp^{k+1} des Ursprungsraumes wurde dann durch Transformation mit der Matrix V in den Raum der Eigenvektoren transformiert ($V^T \Delta p^{k+1}$), dort wurden die Komponenten oberhalb des Indexes i_{ess} auf Null gesetzt, was einer Projektion auf die wesentlichen Richtungen entspricht, und der so resultierende Vektor $\Delta \widehat{p}^{k+1}$ wurde in den ursprünglichen Raum zurücktransferiert (löse ${V^T}^{-1} \Delta \widehat{p}^{k+1}$ $= \Delta p_{neu}^{k+1}$).

Aufgrund der Konstruktion stimmt der Vektor Δp_{neu}^{k+1} mit dem Ergebnisparametervektor aus der Projektion des Raumes überein. Diese Alternative hat aber den Vorteil,

dass der bisherige Algorithmus ohne große Änderungen bestehen bleiben kann und (orthogonal dazu implementiert) nur der ermittelte Parametervektor durch Transformation, Projektion und Rücktransformation verändert wurde.

Zusammenfassung Das Verfahren wird seit jetzt 3 Jahren in mehreren Programmpaketen der Firma CiT zur Simulation von chemischen Prozessen verschiedenster Art benutzt. Es zeigt sich, dass bei sämtlichen Problemen der Parameterschätzung, in denen der Algorithmus ohne das Verfahren der Reduzierten Richtungen zur Konvergenz gelangte, auch der Algorithmus mit Reduzierten Richtungen konvergierte und zwar gegen dieselbe Lösung. Bei nicht-eindeutigen Lösungen, die aufgrund von korrelierten Parametern auftreten, kann der einfache Algorithmus aufgrund der auftretenden Notwendigkeit zur Dämpfung keine Lösung angeben, der Algorithmus mit Reduzierten Richtungen dagegen kann eine der Lösungen finden. In der überwiegenden Zahl der Fälle reicht dafür die Belegung des *Konditionsschwellwertes* zur Bestimmung der Anzahl essentieller Richtungen mit 100 aus, in manchen Fällen ist es nötig, diesen Wert nach oben zu korrigieren, damit nicht zuviele Richtungen abgeschnitten werden. Ein Beispiel zur Effektivität dieses Verfahrens findet sich im anschliessenden Kapitel.

5.4 Erweiterung der Software

Die Tatsache, dass der Code von Presto-Kinetics zur Verfügung steht, ermöglicht es, die Erweiterungen (Reduzierte Richtungen, Overlap-Auswertung und Overlap in der Parameterschätzung) im Kontext eines Programms zu implementieren, welches das effiziente Aufstellen und Validieren von Modellen u.a. der chemischen Reaktionskinetik ermöglicht, sowie die Verwaltung von Messdaten und ein Parameterschätzmodul zur Verfügung stellt, mit dessen Hilfe beliebig viele Parameter des Modells bzgl. beliebig vieler, auch unter verschiedenen Versuchsbedingungen aufgenommener, Messdaten gefittet werden können. Dies alles geschieht unter einer benutzerfreundlichen Oberfläche, so dass Änderungen am Modell und auch an den Einstellungen zur PE leicht und transparent durchzuführen sind, und ist seit über 10 Jahren durch viele Anwendungen validiert.

Die Erweiterung von bestehender Software hat den Vorteil, dass bereits Datenstrukturen vorhanden sind, auf die man direkt zugreifen kann oder die man möglicherweise durch kleine Erweiterungen in einen gewünschten Zustand bringen kann, hat aber den Nachteil, dass man eben diese bereits bestehenden Datenstrukturen nutzen muss, auch wenn manche vorhandenen Funktionalitäten nicht direkt durchschaubar zu sein scheinen. Gerade bei Software, die in der echten Anwendung genutzt wird und die mit dem Ziel erweitert wird, diese Erweiterung auch tatsächlich in der Praxis zu nutzen, ergeben sich besondere Anforderungen:

1. Aufwärtskompatibilität in der Funktionalität: die gesamte bereits vorliegende Funktionalität muss erhalten bleiben, d.h. insbesondere, Simulationen und Parameterschätzungen müssen auch nach einer Änderung/Erweiterung des Codes noch identische Ergebnisse liefern
2. Aufwärtskompatibilität in den Dateien: die Syntax in Modellen oder benötigten Dateien muss gültig bleiben, d.h. das erweiterte Programm muss Dateien lesen

und korrekt interpretieren können, die von älteren Versionen des Programms geliefert werden

3. Abwärtskompatibilität so weit möglich: es ist sehr hilfreich, wenn ältere Versionen des Programms Dateien lesen und korrekt interpretieren kann, die von der neuen Programmversion erstellt wurden.

Im vorliegenden Fall ist es so, dass Presto-Kinetics nicht nur eine modulare Oberfläche für den Benutzer bietet, sondern selbst in modularen Teilen erstellt ist. Das ermöglicht es zum einen, das Programm nach Benutzeranforderungen zu konfigurieren (dazu werden manche Teile der Software nicht compiliert), zum anderen aber auch, Module und damit Funktionalität in Gänze für andere Programme und damit auch für andere Problemklassen (Polymerchemie, Biokinetik, Pharmakokinetik) zu übernehmen! Ziel ist daher natürlich, die erforderlichen Erweiterungen ebenso in modularer Form zuzufügen.

5.4.1 Reduzierte Richtungen

Für das Verfahren der reduzierten Richtungen ist eine Erweiterung in modularer Form relativ leicht, vor allem, wenn man die Alternative mit der Projektion des Lösungsvektors (siehe 5.3, "Projektion des Lösungsvektors") benutzt und dies bei der Bestimmung der Parameteränderung (siehe (26)) nutzt. Hier reicht die Einführung einer Unterfunktion. Für Testzwecke wurden natürlich auch die Alternativen mit Projektion der Raumes implementiert; dann wird die gesamte Bestimmung des Änderungsvektors inklusive der Projektion an eine Unterfunktion übergeben.

Mehrere Einstellungen können vor Start einer PE gemacht werden (siehe Abbildung 4):

1. Wahl des Projektionsverfahrens
2. Einstellungen zur Bestimmung der essentiellen Richtungen (SVD: singular value decomposition):
   (a) SVD, Anzahl essentieller Richtungen wird über den Konditionswert ermittelt, der hier auch abweichend von 100 eingegeben werden kann
   (b) SVD, Anzahl essentieller Richtungen wird über einen Schwellwert ermittelt, der an den aktuellen Dämpfungsfaktor gekoppelt ist (nur testweise)
   (c) SVD, max. Anzahl essentieller Richtungen wird hier fest vorgegeben
   (d) Benutzereingabe: nach jedem Schritt der PE werden dem Benutzer die Singulärwerte gezeigt und er wird gefragt, welche Anzahl er nutzen will

Während einer Parameterschätzung werden die Anzahl essentieller Richtungen sowie die Quotienten der ermittelten Singulärwerte ausgegeben.

Es würde den Rahmen dieser Arbeit sprengen, wenn alle Eingabe-Möglichkeiten hier diskutiert werden würden. Als Ergebnis der Untersuchungen läßt sich aber folgendes festhalten:

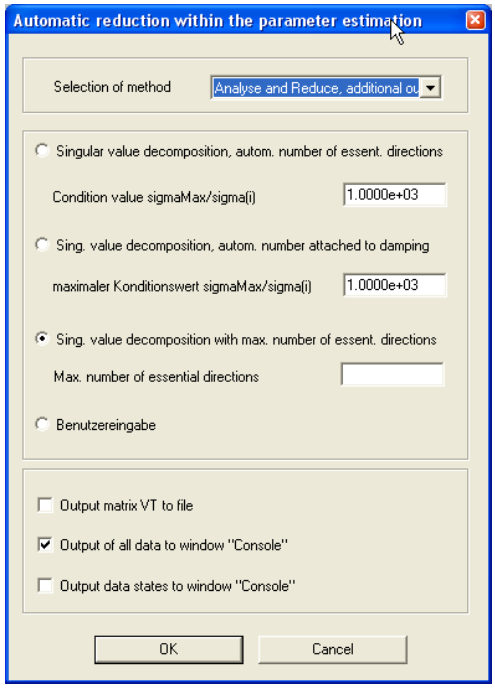

Abbildung 4: Mögliche Einstellungen zum Verfahren

1. die Ermittlung von optimalen Werten einer PE gelingt immer in höchstens sovielen Schritten, wie sie eine PE in bisheriger Form auch benötigt hat, d.h. das Verfahren ist bei Problemen, in denen keine korrelierten Parameter auftauchen, nicht schlechter als das bisherige Verfahren

2. es gibt Probleme, in denen das bisherige Verfahren aufgrund schlechter Startdaten keinerlei Fortschritt erzielt, das Verfahren mit reduzierten Richtungen aber (initiiert durch eine manuell herbeigeführte Reduktion) einen Weg und eine optimale Lösung findet

3. bei Vorliegen von korrelierten Parametern, aber ausreichend guten Startwerten, findet das Verfahren der reduzierten Richtungen eine der Lösungen.

Beispiele für die Effektivität des Verfahrens finden sich in [47].

5.4.2 Overlap-Auswertung

Die Erweiterung von Presto-Kinetics durch die Overlap-Auswertung und die erweiterte Parameterschätzung kann nicht in vollständig modularer Form durchgeführt werden, da an mehreren Stellen innerhalb des Algorithmus Änderungen eingebaut werden müssen. Insbesondere die Bestimmung der erweiterten Jacobi-Matrix benötigt Fallunterscheidungen an verschiedenen Stellen der Implementierung.

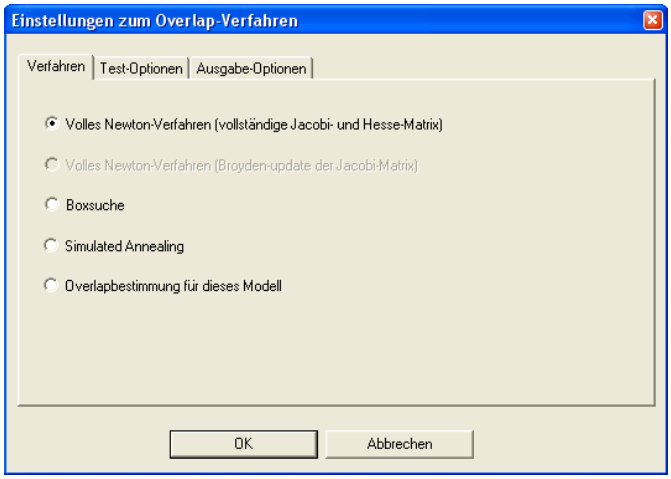

Abbildung 5: Mögliche Einstellungen zur Wahl des Optimierungsverfahrens

Erfolgreich implementiert wurden folgende wählbare Optionen:

1. Wahl des PE-Verfahrens: vollständiges Gauss-Newton-Verfahren, Boxsuche oder Simulated Annealing für das OverlapFunktional stehen neben der reinen Auswertung des Overlap-Funktionals (für gegebene Parameter und Parametervarianzen) zur Verfügung

2. Test-Optionen sind einstellbar: Tests der Hesse-Matrix auf Symmetrie, Einstellen des Parameters α, der in (29) erwähnt wurde.

Es ist gelungen, die Erweiterung so zu implementieren, dass Standardverfahren und erweitertes Overlap-Verfahren in einer ausführbaren Anweisung *.exe genutzt werden können. Damit können Ergebnisse von nötigen Vergleichen zwischen den Verfahren genau auf das Verhalten der Verfahren zurückgeführt werden, da die restliche Programmierumgebung identisch ist.