

Modellbildung & Simulation

Ein Überblick über die wichtigsten Themenbereiche,
basierend auf der Vorlesung von Prof. Troch und
Prof. Breiteneker im SS 2003

© Bernhard Kabelka
Version 1.02, 14. Mai 2004

Es sei ausdrücklich betont, dass

- (1) dieses Manuskript ohne das Wissen und die Mitarbeit von Prof. Troch und Prof. Breiteneker entstanden ist,
- (2) trotz großer Anstrengungen seitens des Autors, eine möglichst fehlerfreies Essay zu liefern, sich Fehler eingeschlichen haben könnten (sollte jemand einen Fehler entdecken, so bittet der Autor um Benachrichtigung per eMail an bernhard@kabelka.net),
- (3) die Lektüre dieses Manuskripts nur einen groben Überblick über den Stoff der Vorlesung „Modellbildung & Simulation“ liefern kann. Es ersetzt keinesfalls den persönlichen Besuch der Vorlesung bzw. das Studium des Skriptums.

Die aktuelle Version dieser Datei ist erhältlich unter:

<http://fsmat.at/~bkabelka/math/modsim/download/modsim.pdf>

<http://fsmat.at/~bkabelka/math/modsim/download/modsim.ps.gz>

Inhaltsverzeichnis

I	Modellbildung & Simulation	1
1	Grundlagen der Modellbildung	1
2	Grundlagen der Simulation	2
3	Arten der Modellbildung	2
3.1	Struktur- und Verhaltensmodelle	3
4	Zustandsereignisse	3
II	Numerische Grundlagen	5
1	Einschrittverfahren	5
1.1	Expliziter und impliziter Euler	5
1.2	Das Verfahren von Heun	6
1.3	Runge-Kutta-Verfahren	7
1.4	Runge-Kutta-Fehlberg-Algorithmen	7
2	Mehrschrittverfahren	8
2.1	AB- und AM-Verfahren	8
2.2	Nordsieck-Notation	8
3	Steife Systeme	9
4	Verfahren für implizite Systeme	10

Teil I

Modellbildung & Simulation

Grundsätzlich sind Modellbildung und Simulation eng miteinander verwoben. Die **Ziele** dabei sind unter anderem

- das Verständnis eines bestimmten Mechanismus' im untersuchten Prozess zu erhöhen
- bestimmte Vorhersagen über das Systemverhalten machen zu können (wo das sinnvoll und brauchbar erscheint)
- ein Kontrollsystem für einen bestimmten Prozess zu entwerfen oder zu testen
- bestimmte Untersuchungen zu ermöglichen, die in der Praxis zu aufwändig, zu teuer, oder zu gefährlich wären

1 Grundlagen der Modellbildung

Ein **Modell** ist eine vereinfachte (!) Repräsentation des zu untersuchenden Systems, um unsere Fähigkeit zu verbessern, das Systemverhalten zu verstehen, zu erklären, vorherzusagen oder zu kontrollieren. Im Allgemeinen werden bei der Modellbildung nur die wesentlichen Aspekte berücksichtigt, andernfalls ist das Erstellen des Modells sehr aufwändig und ineffizient.

Grundsätzlich gibt es hierzu zwei verschiedene Zugänge:

- **Top-Down:**
Berücksichtigung *aller* Gleichungen von Anfang an, langsames nachträgliches Vereinfachen (Linearisierung, Vernachlässigung kleiner Terme, etc.)
- **Bottom-Up:**
Beginn mit der einfachsten Gleichung, langsames Hinzunehmen anderer Einflüsse

Die drei wichtigsten **Grundlagen** bei der Modellbildung sind die folgenden:

- **Trennbarkeit:**
Nur ein Teil der Zusammenhänge wird berücksichtigt, d. h. das betrachtete System wird (so die Annahme) nicht zusätzlich „von außen“ beeinflusst.
- **Auswählbarkeit:**
Nur bestimmte Interaktionen zwischen dem betrachteten System und der Außenwelt werden als relevant ausgewählt.
- **Kausalität:**
Alles beruht auf Ursache und Wirkung (d. h. Ein- und Ausgänge sind kausal miteinander verknüpft).

2 Grundlagen der Simulation

Im Allgemeinen werden bei der Simulation folgende Schritte durchlaufen:

- (1) Problemformulierung
- (2) Erstellen eines mathematischen Modells (Umsetzen des Problems in eine geeignete formale Beschreibung durch Gleichungen o. Ä.)
- (3) mathematische Umformungen (Transformation auf eine „computergerechte“ Form)
- (4) Erstellen des Computerprogramms
- (5) **Verifikation** (Überprüfung, ob das erstellte Programm mit der ursprünglichen mathematischen Formulierung konform ist)
- (6) Datensammlung und **Identifikation** (Bestimmung der Modellparameter anhand von realen Daten)
- (7) **Validierung** (Überprüfung, ob die Resultate des Modells mit den Daten aus dem realen Prozess überein stimmen)
- (8) Simulation (Durchführung der gewünschten Experimente)
- (9) Interpretation (Übertragung der erhaltenen Ergebnisse auf den realen Prozess)

Allerdings ist zu beachten, dass diese Schritte nicht notwendigerweise linear nacheinander ablaufen. Es könnte auch sein, dass Schleifen auftreten, und manche Schritte daher mehrfach durchlaufen werden.

3 Arten der Modellbildung

Ein mathematisches Modell kann in verschiedene Klassen eingeteilt werden:

- statisch oder dynamisch
- linear oder nichtlinear
- kontinuierlich, diskret oder hybrid
- konzentrierte oder verteilte Parameter
- stochastisch oder deterministisch
- etc.

3.1 Struktur- und Verhaltensmodelle

Beim **Strukturmodell** erfolgt die Modellierung durch **Deduktion**, d. h. durch Verwendung von Naturgesetzen (Massen- und Energieerhaltungssatz, Gravitationsgesetz, usw.), Anwenden des Analogieprinzips, etc.

Wird ein **Verhaltensmodell** (auch **Eingangs/Ausgangsmodell** oder **Black-Box-Modell** genannt) erstellt, so bedient man sich der **Induktion**: Es werden sinnvoll erscheinende Annahmen über den modellierten Prozess gemacht, die allerdings nicht auf Naturgesetzen beruhen (müssen). Meist wird einfach versucht, den Eingang auf den gewünschten Ausgang abzubilden (z. B. mittels Regressionsrechnung), d. h. das Modell ist im wesentlichen eine Black Box. Dafür müssen natürlich genügend Messdaten zur Verfügung stehen.

Letztere Art von Modellen wird natürlich meist dann verwendet, wenn es keine (oder nicht genug) Gesetze gibt, die die Erstellung eines Strukturmodells ermöglichen würden, also beispielsweise im Bereich der Soziologie oder der Psychologie.

Im technisch-naturwissenschaftlichen Bereich kommt meist eine Mischform zur Anwendung, da ja sehr wohl „verwertbare“ Naturgesetze existieren, manche andere Einflüsse aber nur induktiv modelliert werden können (z. B. Reibung, Federkennlinien, etc.).

In Hinblick auf die induktive Modellierung sei noch kurz auf den Einfluss des sogenannten **Gütemaßes** hingewiesen: Im Allgemeinen werden die bei der Identifikation entstehenden Gleichungen nicht eindeutig lösbar sein, sodass eine Ausgleichsrechnung erforderlich wird. In diesem Fall muss man sich entscheiden, mit welchem Gütemaß man den Fehler misst (Fehlerquadratsumme, gewichtete Fehlerquadratsumme, etc.). Je nach Wahl des Gütemaßes kann das Endergebnis (vor allem in Hinblick auf Prognosen für die Zukunft) deutlich differieren.

4 Zustandsereignisse

Oft tritt bei der Modellbildung der Fall ein, dass sich die Modellgleichungen (abhängig von gewissen Bedingungen) mehrmals ändern, wobei man grundsätzlich zwischen zwei Sorten von Ereignissen unterscheidet:

- Treten solche Änderungen nur an vorher definierten Zeitpunkten auf, so spricht man von **Zeitereignissen**.
- Ist der Umschaltzeitpunkt sowohl von der Zeit, als auch vom Wert der Zustandsvariablen abhängig (d. h. beispielsweise muss $g(t, \vec{x}(t)) = 0$ gelten), so spricht man von **Zustandsereignissen**.

Ist die Zustandsgleichung

$$\dot{\vec{x}}(t) = \vec{f}(\vec{x}(t), \vec{u}, t, \vec{p})$$

gegeben, wobei \vec{u} den Eingang und \vec{p} den Parametervektor bezeichne, so unterscheidet man zwischen vier verschiedenen Arten von Zustandsereignissen:

- **ZE 1. Art:** unstetige Änderung in \vec{p}
- **ZE 2. Art:** unstetige Änderung in \vec{u}
- **ZE 3. Art:** unstetige Änderung in \vec{x}
- **ZE 4. Art:** unstetige Änderung in der Dimension (im Freiheitsgrad)

Das **Event Handling** erfolgt meist gemäß folgendem Schema:

- (1) **Detection:**
Beobachtung des Vorzeichens von $g(t, \vec{x}(t))$
- (2) **Localisation:**
Bestimmung der genauen Nullstelle \hat{t} durch
 - Iteration: Bisektionsverfahren, Sekantenverfahren
 - Interpolation: meist mit Polynomen
- (3) **Handling:**
Durchführen des letzten Integrationsschrittes $t_i \rightarrow \hat{t}$
Event Action
- (4) **Restart:**
Setzen von $t_0 := \hat{t}$, $t_1 := t_0 + h$

Unter Umständen erfolgt nach der Localisation eine „Auftrennung“, und das Handling wird nicht vom Simulator, sondern in der Runtime-Umgebung durchgeführt, was auf ein hybrides Modell führt.

Teil II

Numerische Grundlagen

Bei der Modellbildung treten oft Differentialgleichungen auf, die nicht (oder nur mit großem Aufwand) exakt gelöst werden können. Deswegen versucht man, die Lösung mit einem numerischen Verfahren zu bestimmen.

Hat man das Differentialgleichungssystem

$$\dot{x}(t) = f(t, x(t)), \quad x(t_0) = x_0, \quad t \in [t_0, t_e]$$

gegeben, so gilt

$$\int_{t_0}^t \dot{x}(s) ds = x(t) - x(t_0) = \int_{t_0}^t f(s, x(s)) ds$$

woraus

$$x(t) = x(t_0) + \int_{t_0}^t f(s, x(s)) ds$$

folgt.

Man wählt nun (im Allgemeinen äquidistante) Gitterpunkte t_i ($i = 1, \dots, n$) mit $t_0 < t_1 < t_2 < \dots < t_n = t_e$ und approximiert jeweils das Integral

$$\int_{t_i}^{t_{i+1}} f(s, x(s)) ds$$

um einen Näherungswert für $x(t_{i+1})$ zu berechnen. Die unterschiedlichen Methoden unterscheiden sich im Wesentlichen nur durch die Art der Approximation des Integrals.

1 Einschrittverfahren

Bei **Einschrittverfahren** geht nur ein „alter“ Wert x_i in die Berechnung von x_{i+1} ein.

1.1 Expliziter und impliziter Euler

Die einfachste Methode, obiges Integral abzuschätzen ist durch Rechteck mit Eckpunkten in $(t_i, 0)$, $(t_i, f(t_i, x_i))$, $(t_{i+1}, f(t_i, x_i))$ und $(t_{i+1}, 0)$. So erhält man das **explizite Euler-Cauchy-Polygonzugverfahren** (oder kurz **explizites Eulerverfahren**):

$$x_{i+1} = x_i + h \cdot f(t_i, x_i)$$

Der lokale Fehler e_i bei diesem Verfahren ist (wie man durch Taylorentwicklung leicht feststellen kann) von der Ordnung h^2 . Für den globalen Fehler E_n ergibt sich

$$E_n \leq n \cdot \max_{i=1}^n e_i + F_n = \frac{t_e - t_0}{h} \cdot \max_{i=1}^n e_i + F_n$$

wobei F_n der sogenannte Fortpflanzungsfehler ist. Verhält sich dieser ebenfalls wie $O(h^2)$, so strebt der globale Fehler mit h gegen Null, wenn h gegen Null geht, d. h. das explizite Euler-Verfahren ist ein Verfahren erster Ordnung.

Ist f stetig und erfüllt es die Lipschitzbedingung

$$|f(t, x_1) - f(t, x_2)| < L \cdot |x_1 - x_2|$$

so gilt für den globalen Verfahrensfehler

$$E_n = \frac{c \cdot h}{2L} \cdot e^{L \cdot (x_n - x_0) - 1}$$

Gleichzeitig verhält sich doch der Rechenfehler R_n wie folgt:

$$R_n = \frac{1}{2^k \cdot h} \cdot e^{L \cdot (x_n - x_0) - 1}$$

Das bedeutet natürlich, dass die Schrittweite nicht zu klein gewählt werden sollte.

Eine andere Methode der Schätzung des Integrals ist durch ein Rechteck mit Eckpunkten in $(t_i, 0)$, $(t_i, f(t_{i+1}, x_{i+1}))$, $(t_{i+1}, f(t_{i+1}, x_{i+1}))$ und $(t_{i+1}, 0)$, was auf das **implizite Eulerverfahren** führt:

$$x_{i+1} = x_i + h \cdot f(t_{i+1}, x_{i+1})$$

Dabei müsste jedoch eine implizite Gleichung gelöst werden, was im Allgemeinen sehr aufwändig ist. Deswegen bedient man sich der **Prädiktor-Korrektor-Technik**: Mit Hilfe eines expliziten Verfahrens wird ein Prädiktorwert $x_{i+1}^{[P]}$ berechnet, der vom impliziten Eulerverfahren auf den Wert $x_{i+1}^{[K]}$ korrigiert wird:

$$x_{i+1}^{[P]} = x_i + h \cdot f(t_i, x_i), \quad x_{i+1}^{[K]} = x_i + h \cdot f(t_{i+1}, x_{i+1}^{[P]})$$

1.2 Das Verfahren von Heun

Man kann zur Approximation des Integrals natürlich auch die Trapezfläche mit den Eckpunkten $(t_i, 0)$, $(t_i, f(t_i, x_i))$, $(t_{i+1}, f(t_{i+1}, x_{i+1}))$ heranziehen, was auf ein weiteres implizites Verfahren führt:

$$x_{i+1} = x_i + \frac{h}{2} \cdot (f(t_i, x_i) + f(t_{i+1}, x_{i+1}))$$

führt. Berechnet man nun wieder einen Prädiktor mit Hilfe des expliziten Eulerverfahrens, so erhält man das **Verfahren von Heun**:

$$x_{i+1} = x_i + \frac{h}{2} \cdot (f(t_i, x_i) + f(t_{i+1}, x_i + h \cdot f(t_i, x_i)))$$

Dieses Verfahren ist ein Verfahren zweiter Ordnung.

1.3 Runge-Kutta-Verfahren

Alle obigen Verfahren sind spezielle **Runge-Kutta-Verfahren**. Allgemein besteht ein Runge-Kutta-Verfahren aus folgenden Schritten:

- (1) Durchführung eines expliziten Euler-Schrittes von t_i nach t_{i+1} mit der Steigung $k_1 = f(t_i, x_i)$ und Berechnung der Steigung $k_2 = f(t_{i+1}, x_{i+1})$
- (2) Verbesserung der Steigung k_1 mit Hilfe von k_2 zur Steigung k_3
- (3) Für $j = 3, 4, \dots, m - 1$:
 - (a) Durchführung weiterer expliziten Euler-Schrittes von t_i nach $t_\eta := t_i + \eta$ ($0 < \eta \leq 1$) mit Steigung k_j
 - (b) Berechnung der Steigung $k_{j+1} = f(t_\eta, x_\eta)$
 - (c) Verbesserung der Steigung k_j in (t_i, x_i) mittels k_{j+1}
- (4) Kombination aller Steigungen k_l für einen abschließenden Euler-Schritt

Formal lässt sich das wie folgt formulieren:

$$s_j = h \cdot f \left(t_i + \alpha_j \cdot h, x_i + \sum_{k=1}^m \beta_{jk} s_k \right) \quad \text{für } j = 1, 2, \dots, m$$
$$x_{i+1} = x_i + \sum_{k=1}^m a_j s_k$$

wobei die Parameter α_j , β_{jk} und a_j so zu wählen sind, das die Verfahrensordnung möglichst hoch ist.

Gilt $b_{jk} = 0$ für alle (j, k) mit $j \leq k$, so ist das Verfahren explizit, andernfalls liegt ein implizites Verfahren vor, das iterativ mit Hilfe der Prädiktor-Korrektor-Technik gelöst werden muss.

Zur Verfahrensordnung lässt sich sagen, dass man mit m Funktionsauswertungen ein Verfahren m -ter Ordnung konstruieren kann, sofern $m \leq 4$ ist. Ist $5 \leq m \leq 8$, so kann man ein Verfahren $(m - 1)$ -ter Ordnung konstruieren.

1.4 Runge-Kutta-Fehlberg-Algorithmen

RKF-Algorithmen werden zur Schrittweitensteuerung eingesetzt. Dabei wird zuerst mit einem Runge-Kutta-Verfahren der Ordnung m_1 ein Prädiktorwert $x_{i+1}^{[P]}$ berechnet, und anschließend wird ein Korrektorwert $x_{i+1}^{[K]}$ mit einem Runge-Kutta-Verfahren der Ordnung $m_2 > m_1$ berechnet. Diese beiden Werte werden nun zur Schätzung des lokalen Fehler herangezogen:

$$e_{i+1}^{[P]} \sim \left| x_{i+1}^{[P]} - x_{i+1}^{[K]} \right|$$

Ist der Fehler größer als die vorgegebene Genauigkeit, so wird die Schrittweite verkleinert; ist er signifikant kleiner, so wird die Schrittweite vergrößert.

Übrigens werden die beiden Runge-Kutta-Verfahren so gewählt, dass nur $m_2 + 1$ Funktionsauswertungen benötigt werden.

2 Mehrschrittverfahren

Bei **Mehrschrittverfahren** werden mehrere „alte“ Werte x_{i-s+1}, \dots, x_i zur Berechnung von x_{i+1} herangezogen, indem man $f(t, x(t))$ an diesen Stellen durch ein Polynom interpoliert, über das dann integriert wird. So ergibt sich ein Verfahren der Gestalt

$$x_{i+1} = x_i + \sum_{j=1}^{s \text{ bzw. } s+1} a_j \cdot f(t_{i-s+j}, x_{i-s+j})$$

Ein derartiges Verfahren (das s Vorwerte verwendet) hat eine globale Fehlerordnung von s .

2.1 AB- und AM-Verfahren

Die expliziten Verfahren dieser Gestalt nennt man **Adams-Bashforth-Verfahren**, die impliziten werden **Adams-Moulton-Verfahren** genannt.

Zur Schrittweitensteuerung kann man (wie bei den RKF-Verfahren) die Prädiktor-Korrektor-Technik einsetzen: Zuerst berechnet man mit Hilfe eines AB-Verfahrens einen Prädiktorwert $x_{i+1}^{[P]}$, der dann zur Berechnung des Korrektors $x_{i+1}^{[K]}$ mittels eines AM-Verfahrens (üblicherweise von gleicher oder von um eins höherer Ordnung wie das AB-Verfahren) herangezogen wird (mathematisch entspricht das einer Fixpunktiteration). Der lokale Fehler e_{i+1} lässt sich nun wieder wie folgt abschätzen:

$$e_{i+1} \sim \left| x_{i+1}^{[P]} - x_{i+1}^{[K]} \right|$$

Das würde (theoretisch) eine Schrittweitensteuerung erlauben, allerdings müssten bei einer Änderung der Schrittweite die neuen Vorwerte erst interpoliert werden, was natürlich etwas aufwändig ist. Darüber hinaus stellt sich (bei allen bisher erwähnten Mehrschrittverfahren) die Frage nach den Startwerten – kein Verfahren ist selbststartend.

2.2 Nordsieck-Notation

Das Problem der fehlenden Startwerte kann man mit Hilfe der sogenannten Nordsieck-Notation in den Griff bekommen. Für diesen Zweck schreibt man das Interpolationspolynom $p(t)$ wie folgt an:

$$p(t) = b_0 + b_1 \cdot (t - t_i) + b_2 \cdot (t - t_i)^2 + \dots + b_{s-1} \cdot (t - t_i)^{s-1}$$

Da dieses Polynom die Funktion $f(t, x(t))$ approximiert, gilt

$$p^{(k)}(t) \sim f^{(k)}(t, x(t)) = x^{(k+1)}(t)$$

und damit

$$b_k = \frac{p^{(k)}(t)}{k!} \sim \frac{f^{(k)}(t, x(t))}{k!} = \frac{x^{(k+1)}(t)}{k!}$$

Man speichert nun die b_k in dem sogenannten **Nordsieck-Vektor** \vec{n}_i , wobei mit den passenden Potenzen von h multipliziert wird:

$$\vec{n}_i = \begin{pmatrix} x_i \\ h \cdot \dot{x}_i \\ h^2 \cdot \ddot{x}_i \\ \vdots \\ \frac{h^s}{(s-1)!} \cdot x_i^{(s)} \end{pmatrix} = \begin{pmatrix} x_i \\ h \cdot f_i \\ h^2 \cdot \dot{f}_i \\ \vdots \\ \frac{h^s}{(s-1)!} \cdot f_i^{(s-1)} \end{pmatrix} = \begin{pmatrix} x_i \\ h \cdot b_0 \\ h^2 \cdot b_1 \\ \vdots \\ h^s \cdot b_{s-1} \end{pmatrix}$$

Aus den Komponenten des Nordsieck-Vektors lässt sich nun der Wert x_{i+1} gemäß des oben angeführten Integrationssschrittes berechnen.

Eine Schrittweitensteuerung ist nun sehr einfach, da der Nordsieck-Vektor nur mit den passenden Potenzen des Faktors, um den die Schrittweite verkleinert oder vergrößert wird, multipliziert werden muss.

Auch die Startwerte können so ermittelt werden: Man kann nämlich bei einem Integrationsschritt von x_i auf x_{i+1} – anstatt die Information über f_{i-s+1} „wegzuschmeißen“ – den Nordsieck-Vektor einfach vergrößern. Man startet also im Allgemeinen mit einem Verfahren der Ordnung 1 oder 2, und kann dann sukzessive die Ordnung erhöhen.

3 Steife Systeme

Durch Linearisieren eines Differentialgleichungssystems mit Hilfe der Jacobi-Matrix J kann man Aussagen über die Stabilität von Lösungen gewinnen. Je nach Stabilitätseigenschaften muss man die Schrittweite im Integrationsalgorithmus passend wählen. Als Faustregel gilt, dass die Schrittweite ein gewisser Bruchteil der kleinsten Zeitkonstante des Systems (das ist üblicherweise der Kehrwert des größten Eigenwerts der Jacobi-Matrix) ist. Mit anderen Worten, das Produkt aus Schrittweite und Eigenwert sollte in einem bestimmten Bereich liegen.

Beim expliziten Eulerverfahren gilt beispielsweise für den globalen Fehler

$$\vec{E}_{i+1} = (I + h \cdot J(\eta)) \cdot \vec{E}_i + \vec{e}_{i+1}$$

Dieser strebt nur dann gegen Null (für $h \rightarrow 0$), wenn gilt $\|I + h \cdot J(\eta)\| < 1$ für eine beliebige Matrixnorm $\|\cdot\|$. Das ist genau dann der Fall, wenn für den größten Eigenwert λ von J gilt:

$$|1 + h \cdot \lambda| < 1 \iff 0 < h \cdot \lambda < 2$$

So ergeben sich für jeden Algorithmus unterschiedliche Stabilitätsgebiete. Explizite Runge-Kutta-Verfahren und explizite Mehrschrittverfahren weisen übrigens beschränkte Stabilitätsgebiete in der linken Halbebene auf.

Sind die Eigenwerte eines Systems alle in der selben Größenordnung, so spielen diese beschränkten Stabilitätsgebiete eigentlich keine Rolle. Probleme treten

dann auf, wenn die Größenordnungen der Eigenwerte deutlich unterschiedlich sind (was bei sogenannten **steifen Systemen** auftritt). Die Schrittweite muss dann nämlich dem größten Eigenwert angepasst werden, allerdings ändern sich manche Komponenten kaum, wodurch es zu Auslöschungsfehlern kommt.

Ziel ist es nun, die Stabilitätsgebiete der Integrationsalgorithmen zu vergrößern. Man stellt fest, dass implizite Verfahren deutlich größere Stabilitätsgebiete haben (das implizite Eulerverfahren beispielsweise $\mathbb{C} \setminus \overline{B}(0, 1)$), und versucht daher, implizite Verfahren mit möglichst großem Stabilitätsgebiet zu konstruieren.

Ein Beispiel derartiger Verfahren sind die **Gear-Verfahren**, die gewisse implizite Mehrschrittverfahren darstellen. Zur Lösung kann man allerdings nicht die Prädiktor-Korrektor-Technik verwenden (da explizite Mehrschrittverfahren ein beschränktes Stabilitätsgebiet haben), also verwendet man das Newton-Verfahren zur Lösung des Nullstellenproblems

$$\vec{N}(\vec{x}_{i+1}) = -\vec{x}_{i+1} + \vec{x}_i + \vec{v}_i + \beta \cdot \vec{f}(\vec{x}_{i+1})$$

Die Iterationsvorschrift lautet dann

$$\vec{x}_{i+1}^{[k+1]} = \vec{x}_{i+1}^{[k]} - \left(N' \left(\vec{x}_{i+1}^{[k]} \right) \right)^{-1} \cdot N \left(\vec{x}_{i+1}^{[k]} \right)$$

wobei

$$N'(\vec{u}) := \frac{\partial \vec{N}}{\partial \vec{u}} \quad \Rightarrow \quad N' \left(\vec{x}_{i+1}^{[k]} \right) = -I - \beta \cdot h \cdot J \left(\vec{x}_{i+1}^{[k]} \right)$$

Die verwendete Jacobi-Matrix wird übrigens üblicherweise nicht in jedem Integrationsschritt neu berechnet; es wird nur überprüft, ob der Grad der Änderungen eine Neuberechnung unbedingt erforderlich macht, wodurch der Rechenaufwand etwas geringer ist.

Gear-Verfahren sind natürlich (mittels der Nordsieck-Notation) auch zur Ordnungssteuerung fähig.

Im Allgemeinen geht man nun so vor:

- regelmäßiger Test auf Steifheit (an Hand der Jacobi-Matrix)
- Verwendung eines geeigneten Algorithmus
- Schrittweiten- und Ordnungssteuerung

Auf diesem Weg kann man viele Probleme adäquat lösen.

4 Verfahren für implizite Systeme

Die wenigsten Simulatoren können implizite Systeme direkt behandeln, weshalb eine andere Beschreibung vonnöten ist. Man versucht daher meistens, den Integrationsschritt auf ein Nullstellenproblem zu transformieren, das mit den bekannten Algorithmen gelöst werden kann.

Auf diesem Weg kann sogar eine zusätzliche (durch eine Gleichung beschriebene) Zwangsbedingung behandelt werden.

Index

Adams-Bashforth-Verfahren, 8
Adams-Moulton-Verfahren, 8

Black-Box-Modell, 3
Bottom-Up, 1

Deduktion, 3

Eingangs/Ausgangsmodell, 3
Einschrittverfahren, 5
Euler-Cauchy-Polygonzugverfahren
 explizites, 5
Eulerverfahren
 explizites, 5
 implizites, 6

Gütemaß, 3
Gear-Verfahren, 10

Heun, 6

Identifikation, 2
Induktion, 3

Mehrschrittverfahren, 8

Nordsieck-Vektor, 9

Prädiktor-Korrektor-Technik, 6

RKF-Algorithmen, 7
Runge-Kutta-Verfahren, 7

steifes System, 10
Strukturmodell, 3

Top-Down, 1

Validierung, 2
Verfahren von Heun, 6
Verhaltensmodell, 3
Verifikation, 2

Zeitereignis, 3
Zustandsereignis, 3