Formelsammlung
Grundlagen der Modellierung und Simulation

<Marco.Moeller@macrolab.de>

Stand: 16.10.2005 - Version: 1.0.0
Erhältlich unter http://privat.macrolab.de

Diese Formelsammlung basiert auf der Vorlesung "Einführung in Computational Engineering (für Inf.) / Grundlagen der Modellierung und Simulation (für CE)" von Prof. Dr. Oskar von Stryk an der Technischen Universität Darmstadt im Sommersemester 2005.

Inhaltsverzeichnis

1 Einführung

1.1 Begriffsbildung

Abstraktes Modell formale Beschreibung, typischerweise (aber nicht nur) mit den Mitteln der Mathematik

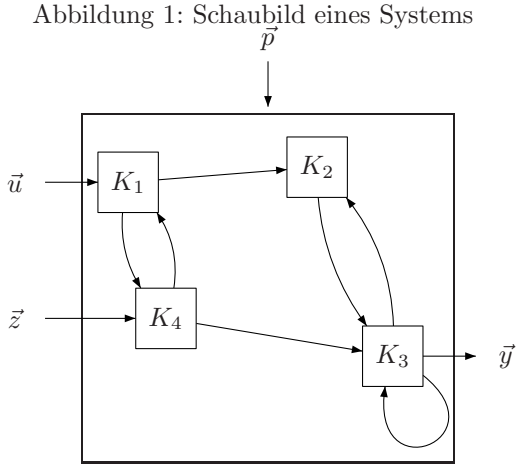
Mathematische Modellierung Prozess der formalen Herleitung und Analyse eines mathematischen Modells

Simulation Nachahmung von Eigenschaften eines technischen Systems oder naturwissenschaftlichen Phänomens auf einem Computer

1.2 Systemstruktur eines Modells

In Abbildung 1 sieht man exemplarisch ein System. Es hat folgende Komponenten:

Abbildung 1: Schaubild eines Systems



Systemzustand $\vec{x}$

- vollständige Charakterisierung des Systemverhaltens

Stellgrößen $\vec{u}$

- Systemeingang - steuerbar

Störgrößen $\vec{z}$

- Systemeingang - nicht kontrollierbar

Systemparameter $\vec{p}$

- konstant für einen Simulationslauf
- Wahl dieser Parameter anhand:
  - Bestimmung der Zeitkonstanten im Vergleich zur Analytischer Lösung / Realer Daten

  - Minimierung der Quadrate der Abweichung zur Analytischer Lösung / Realer Daten

Systemausgang $\vec{y}$

- mess- / beobachtbare Größen

1.3 Klassifikation von Modellen

1.3.1 Allgemein

diskret ein solches Modell nutzt diskrete / kombinatorische Beschreibung

kontinuierlich ein solches Modell nutzt kontinuierliche / reellwertige Beschreibung

deterministisch die Berechnungen im Modell enthalten keine Zufallsgrößen

stochastisch es sind Zufallsentscheidungen / größen im Modell involviert

1.3.2 Zeitachse

Kontinuierlich Achse hat keine Unterbrechungen

diskret äquidistant Achse ist in gleich große Stücke zerlegt

diskret nicht äquidistant Achse ist unterbrochen, aber in unterschiedlichen Schrittweiten

kontinuierlichdiskret es gibt Ausgezeichnete Punkte auf der Achse, aber der Zwischenraum ist vorhanden

zeitdiskret Die Zeitachse ist Unterteilt

ereignisdiskret Die Ereignisse lassen sich voneinander Trennen, diese ersetzen die Zeiachse sozusagen

1.3.3 Zustandsraum

diskrete Werteskala Zusandsmenge ist endlich, bzw. man kann nicht zwischen je zwei Zuständen noch einen Zwischenzustand finden

kontinuierliche Werteskala Zustandsmenge ist Reellwertig

räumlich verteilte Variablen Zustandsmenge ist isomorph zu $\mathbb{R}^n$

1.3.4 Art der Dynamik

instationär Objekt ist in Bewegung

stationär / quasi-stationär Objekt ist Ortsgebunden

Beschreibungsunschärfe

deterministisch Nächste Bewegung ist Vorhersagbar

stochastisch Nächster Zustand ist zufällig

fuzzy Kontinuierliche Logik

1.4 Aufgabenstellungen

Direktes Problem Eingänge und Systemmodell ist bekannt, Ausgang ist Unbekannt

Inverse Probleme Alles bekannt bis auf eines der folgenen: Systemmodell, Eingänge, Systemparameter

1.5 Teilschritte einer Simulationsstudie

1.5.1 Problemspezifikation

- Ausfgabenformulierung
- Kriterienfestlegung
- Datenerhebung

1.5.2 Modellbildung

- Strukturfestlegung
- Modellgleichungen
  - Festlegung der Zustandsvariablen ist der Ausgangspunkt jeder Modellbildung
- Modellvereinfachung

1.5.3 Implementierung

- Modellumsetzung
- Berechnungsverfahren
- Laufzeitoptimierung

1.5.4 Validierung

- Fehlersuche
  - Modellierungsfehler
    - vereinfachende Modellannahmen
    - ungenauigkeiten in Modellparametern
  - Approximationsfehler des iterativen Berechnungsverfahrens
  - Rundungsfehler (Rundung aufgrund der Zahlendarstellung im Computer)
  - Programmier-, Implementierungsfehler
- Konsistenzprüfung
- Daten-, Parameterabgleich

Gesamtfehler

- Fehlerquellen
  - Modellierungsfehler $|P - M|$
  - Diskretisierungsfehler $|M - D|$
  - Abbruchfehler $|D - L|$
  - Visualisierungsfehler $|L - V|$
- Gesamtfehler setzt sich aus Einzelfehlern zusammen

  $$|P - L| \leq |P - M| + |M - D| + |D - L| + |L - V|$$

- Akzeptanz einer Lösung L zu gegebenen Problem P, wenn in allen 4 Schritten vergleichbar kleine Fehler gemacht wurden

Validierung Sicherstellen, dass ein Simulationsmodell das reale System in Rahmen der Zielsetzung ausreichend gut abbildet.

Elemente einer Validierung

- Syntaktische Fehlerfreiheit der Implementierung (Debugging)
- Numerische Korrektheit der Implementierung (Reproduktion einer analytischen Lösung)
- Plausibilität der Simulationsergebnisse und Verfahrensparameter (“naturgetreues” Verhalten)
- Zulässigkeit der Modellannahmen, logische Konsistenz (Anwendbarkeit zur Problemlösung)
- Ausreichende Detailliertheit des Modells (korrekte Modellstruktur)
- Richtigkeit der Modellparameter (Parameteranpassung anhand realer Daten)

1.5.5 Simulationsläufe

- Parametervariation
- Strukturvariation
- Vorhersage und Optimierung

1.5.6 Ergebnispräsentation

- Datenauswertung
- Visualisierung und Animation
  - Zeitverlauf
  - Phasenraumplot (Auftragen der Zustandsgrößen gegeneinander)
  - Spezialvisualisierungen
  - Animation
- Dokumentation

2 Diskrete Modellierung und Simulation

2.1 Terminologie

System Ansammlung von Objekten, die zeitabhängig nach spezifizierten Regeln miteinander interagieren

Modell eine abstrakte, logische und mathematische Darstellung der Objekte und Wechselbeziehungen in einem System

Entität (entity) Objekt in einem System, das innerhalb des Modells explizit dargestellt wird (z.B. Bedieneinheit, Kunde, Maschine, Transporter, Werkstück)

Attribut (attribute) Variable, die den Zustand einer Entität beschreibt (z.B. Bearbeitungszustand, Belegung, Position)

Ereignis (event) Beschreibung der Aktualisierung des Modellzustands (z.B. Betreten eines Puffers, Maschinenbelegung) (Realisierung z.B. mit Ereignisroutine für jeden Ereignistyp)

Ereignisliste eine Liste mit Zeitpunkt und Typ der geplanten Ereignisse, nach Ereigniszeitpunkten aufsteigend geordnet

Aktivität (activity) zeitlich erstreckter Vorgang zwischen den initiierenden und abschließenden Ereignissen einer Operation, die den Zustand einer Entität transformiert (z.B. Aufenthalt in einem Puffer)

Simulationsuhr (simulation clock) eine Variable, die den aktuellen Stand der Simulationszeit angibt

Zeitführungsroutine (timingroutine) eine Prozedur zur Auswahl des nächsten Ereignisses aus der Ereignisliste und Vorstellen der Simulationsuhr auf den nächsten Ereigniszeitpunkt

Ergebnisroutine eine Prozedur zur Berechnung der statistischen Schätzwerte der Ergebnisvariablen anhand statistischer Zähler und zur Ausgabe des Ergebnisprotokolls am Ende des Simulationslaufs

Steuerprogramm ein Programmteil, der wiederholt die Zeitführungsroutine zur Bestimmung des nächsten Ereignistyps aufruft und die zugehörige Ereignisroutine aufruft, bis Simulationslauf beendet

2.2 DEVS

DEVS Discret Event Simulation

Beschreibung siehe Abbildung 2 auf der nächsten Seite

2.3 Warteschlangentheorie

2.3.1 Grundbegriffe Statistik

Zufallsgröße $\underline{x}$

Wahrscheinlichkeit $Pr[\underline{x} \leq x]$ das $\underline{x}$ kleiner oder gleich x

Verteilungsfunktion $F(x) = Pr[\underline{x} \leq x]$

- $F(a) - F(b) = Pr[a < \underline{x} \leq b]$
- $F(-\infty) = 0$
- $F(\infty) = 1$

Stetige Verteilung falls der Wertebereich von $\underline{x}$ überabzählbar unendlich ist, z.B. Reell.

Wahrscheinlichkeitsdichte $f(x)$

- ist nur für stetige Verteilung definiert
- $f(x)\,dx = Pr[x < \underline{x} \leq x + dx]$
- $F(x) = \int_{-\infty}^{x} f(s)\,ds$

Moment das n-te Moment einer Verteilunsfunktion $E[\underline{x}^n] = \int_{-\infty}^{\infty} s^n dF(s)$

Erwartungswert $E[x] == \int_{-\infty}^{\infty} s dF(s)$

- Entspricht erstem Moment
- auch Mittelwert genannt

Varianz

$$\sigma^2 = E\left[(\underline{x} - E[\underline{x}])^2\right] = \int_{-\infty}^{\infty} (s - E[\underline{x}])^2\, dF(s)$$

- Auch Streuungsquadrat
- $E\left[(\underline{x} - a)^2\right] = \sigma^2(\underline{x}) + (E[\underline{x}] - a)^2$

Bedingte Wahrscheinlichkeit $Pr[A|B] = \frac{Pr[A \wedge B]}{Pr[B]}$

- Wahrscheinlichkeit das A eintritt wenn mann bereits weiss, das B eintritt
- $Pr[A \wedge B]$ Wahrscheinlichkeit das A und B zusammen eintreten

Unabhängigkeit zweier Ereignisse A und B, wenn $Pr[A|B] = Pr[A]$

negativ exponentielle Verteilung $F(x) = \begin{cases} 1 - e^{-\mu x} & x \geq 0 \\ 0 & x < 0 \end{cases}$

- Wahrscheinlichkeitsdichte $f(x) = \begin{cases} \mu e^{-\mu x} & x \geq 0 \\ 0 & x < 0 \end{cases}$
- Varianz $\sigma^2(\underline{x}) = \mu^{-2}$
- Dies ist die einzige Verteilungsfunktion die Gedächnislos ist. D.h. es gilt

$$Pr[\underline{x} \leq a + x | \underline{x} > a] = Pr[\underline{x} \leq x]$$

- Wenn im Mittel zwischen zwei Aufträgen t Zeiteinheiten vergehen, dann muss man im Mittel trotzdem noch t Zeiteinheiten warten, unabhängig davon, wieviel Zeit schon vergangen ist.

2.3.2 Warteschlange Allgemein

Anschauung siehe Abbildung 3 auf der nächsten Seite

Beschreibung siehe Abbildung 4 auf der nächsten Seite

Charakterisierung $A/B/c/N/K$

Kundenpopulation K

Ankunftszwischenzeitverteilung A

Warteschlangenlänge N

Bedienstationenanzahl c

Bedienzeitverteilung B

Abkürzungen (typische)

- M (negativ) exponentialverteilt
- D deterministisch oder konstant verteilt

Abbildung 2: Funktionsweise einer DEVS

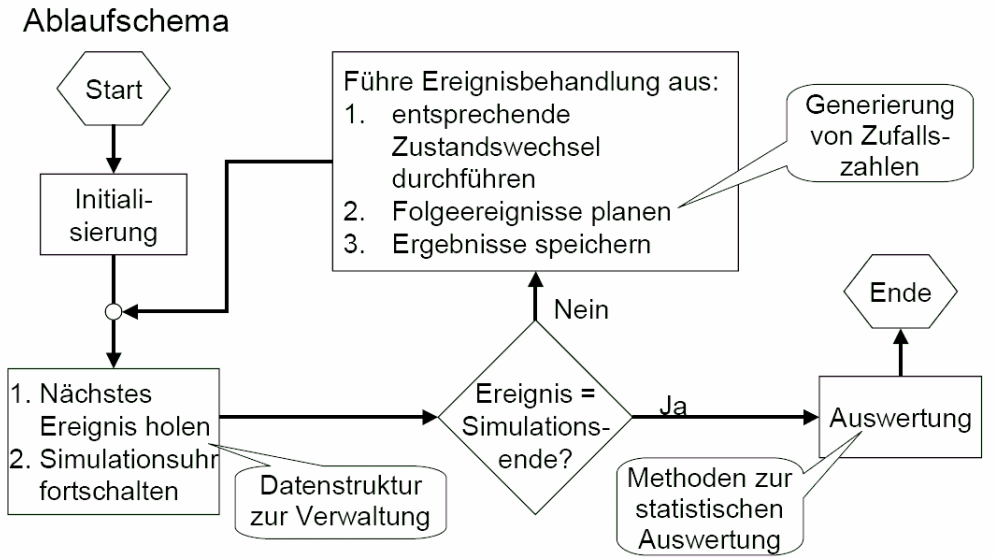


Abbildung 3: Warteschlangenmodell - Anschaulich

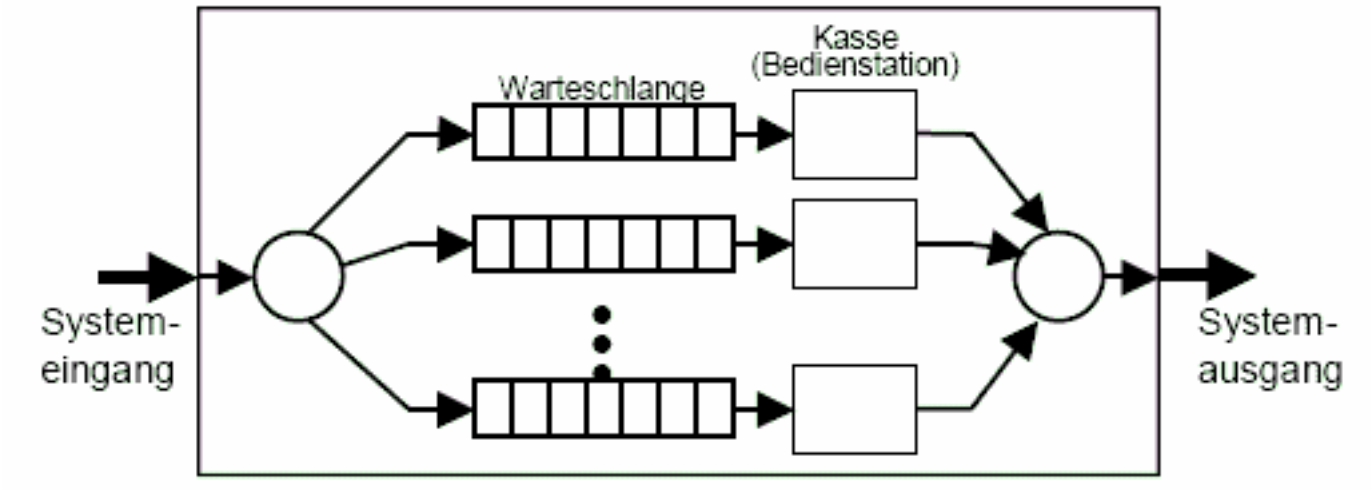
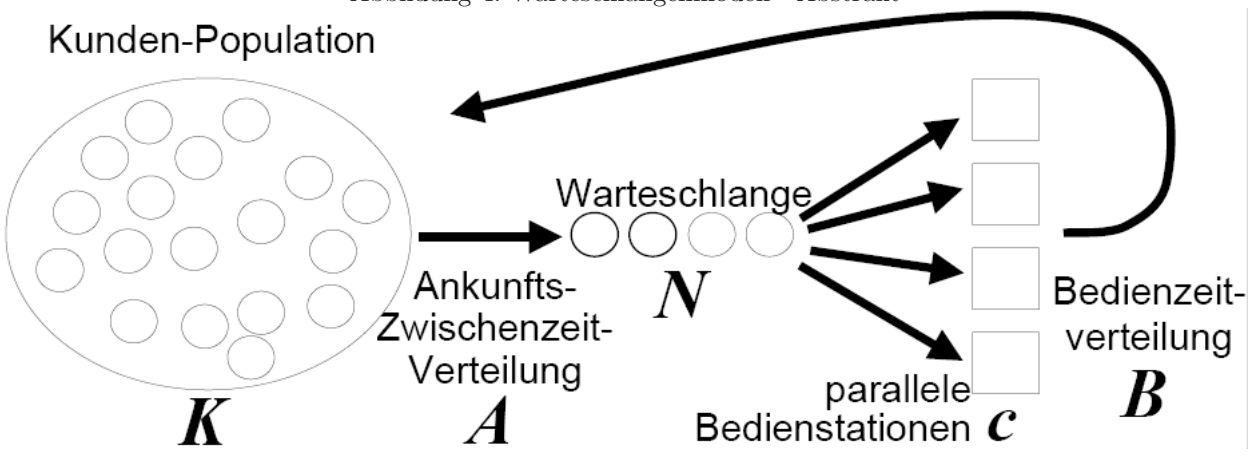


Abbildung 4: Warteschlangenmodell - Abstrakt



- G beliebig verteilt
- z.B. $M/M/1 = M/M/1/\infty/\infty$
  $M/G/1 = M/G/1/\infty/\infty$

Ankunftsrate (mittlere) $\lambda = E[A]$

Bedienrate (mittlere) pro Station $\mu = E[B]$

Streuung Bedienrate $\sigma^2 = \mathrm{Var}(B)$

Kunden im System $L(t)$ zum Zeitpunkt t

Langzeitverhalten $L(t) \to L(\infty)$ stationäre Verteilung

- Ziel der Warteschlangentheorie ist es diese Größe zu bestimmen
- L Erwartungswert von $L(\infty)$
- p Langzeitausnutzung der Server
- w Langzeitaufenthaltzeit im System
- Gesetz von Little
  $L = \lambda \cdot w$

2.3.3 M/M/1

Bedingung für die Stationarität $E[A] = \lambda < \mu = E[B]$

Langzeitausnutzung des Servers $\rho = \frac{\lambda}{\mu}$

Kunden im System $L = \frac{\rho}{1-\rho}$

Langzeitaufenthaltszeit $w = \frac{1}{\mu-\lambda}$

Dichtefunktion von $L(\infty)$

$$d_{L(\infty)}(\xi) = P(L(\infty) = \xi) = (1-\rho)\cdot\rho^{\xi}$$

- dies ist die *geometrische Verteilung*

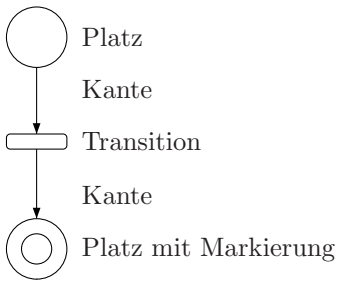
Eigenschaften

- Warteschlangen werden länger, wenn bei konstanten Erwartungswert die Streuung der Bedienzeiten wächst
- Jede Pufferlänge tritt mit gewisser Wahrscheinlichkeit einmal auf, d.h. Puffer laufen stets gelegentlich voll
- n Server mit einer gemeinsamen Warteschlange sind besser als n Server mit n Warteschlangen

2.3.4 M/G/1

- $\rho = \frac{\lambda}{\mu}$
- $L = \rho + \frac{\rho^2\left(1+\frac{\sigma^2}{\mu^2}\right)}{2(1-\rho)}$
- $w = \frac{1}{\mu} + \frac{\lambda\left(1+\frac{\sigma^2}{\mu^2}\right)}{2\mu^2(1-\rho)}$

Abbildung 5: Petrinetz Grundelemente



2.3.5 M/D/1:$\sigma = 0$

- $\rho = \frac{\lambda}{\mu}$
- $L = \rho + \frac{\rho^2}{2(1-\rho)}$
- $w = \frac{1}{\mu} + \frac{\rho}{2\mu(1-\rho)}$

2.3.6 M/M/c

- $\rho = \frac{\lambda}{c\mu}$
- $L = c\rho + \frac{\rho}{1-\rho} Pr[L(\infty) \geq c]$
- $w = \frac{L}{\lambda}$
- $P_0 = \left(\frac{(c\rho)^c}{(1-\rho)\cdot c!} + \sum_{i=0}^{c-1}\frac{(c\rho)^i}{i!}\right)^{-1}$
- $Pr[L(\infty) \geq c] = \frac{(c\rho)^c}{(1-\rho)\cdot c!} P_0$

2.4 Petrinetze

2.4.1 Definition

Petrinetz $PN = (P, T, A, M')$

- Gerichteter, bipartiter Graph - mit 2 Arten von Knoten (Plätzen und Transitionen)
- Zeichnung siehe Abbildung 5

Plätze $P = \{p_1, \ldots, p_n\}$

- beschreiben mögliche Zustände, Darstellung als Kreis
- Englisch: *places*

Transitionen $T = \{t_1, \ldots, t_m\}$

- beschreiben mögliche Ereignisse, Darstellung als Rechteck

gerichtete Kanten $A = \{(a_1, a_1'), \ldots, (a_k, a_k')\}$

- Verbinden Plätze und Transitionen
- Englisch: *directed edges*

Markierungen $M = (m_1, \ldots, m_n) \in \mathbb{N}^n$

- definieren den aktuellen Zustand des Petrinetzses mit
  m_i = Anzahl der Markierungen in Platz p_i
- werden in Plätzen gespeichert, Darstellung als schwarze Punkte im Kreis
- Anfangsmarkierung M'
- Englisch: *tokens*

dynamik eines Petrinetzes ist wie folgt charakterisiert

- Markierungen werden erzeugt, gelöscht oder verschoben durch Schalten von Transitionen (firing of transitions)
- Transition ist Schaltbereit (bzw. aktiviert) wenn alle ihre Eingangsplätze markiert sind (bzw. mind. eine Marke enthalten)
- Schaltbereite Transitionen können schalten (feuern), wobei eine Marke aus jedem Eingangsplatz weggenommen und in jedem Ausgangsplatz eine Marke hinzugefügt wird.
- Feuern ist im allgemeinen ein nichtdeterministischer Vorgang

Typen von Petrinetzen

Zeidiskrete (kausale) Modelle: Petrinetz beschreibt logisch, *was* in welcher Sequenz passiert

Zeitkontinuierliche Modelle: zusätzliche Vorhersage, *wann* ein Ereigniss auftritt.

Stochastische Petrinetze: zufallsverteilte Verzögerungsdauern der Schaltregeln

2.4.2 Mathematische Darstellung zeitdiskreter Petrinetze

Die Struktur eines Petrinetzes mit p Plätzen und t Transitionen kann durch zwei $t \times p$-Matrizen W^-und W^+ dargestellt werden. In W^- werden für alle Verbindungen eines Platze mit einer Transition (in dieser Reihenfolge / pre-weights) 1en und sonst 0en eingetragen. In W^+ werden für alle Verbindungen einer Transition mit einem Platz (in dieser Reihenfolge / post-weights) 1en und sonst 0en eingetragen. Die Inzidenzmatrix ergibt sich wie folgt

$$W = W^+ - W^-$$

2.4.3 Simulation zeitdiskreter Petrinetze

Markierungsvektor zum Zeitpunkt r

$$m(r) = (m_1(r), \ldots, m_n(r))^T$$

Kapazitätsvektor $k = (k_1, \ldots, k_n)^T$

- Es soll gelten $m_i(r) \leq k_i$ für $i \in \{1, \ldots, n\}$

Aktivierungsfunktion

$$u_j(r) = \begin{cases} 1 & \text{Transition } t_j \text{ aktiviert} \\ 0 & \text{sonst} \end{cases}$$

- In allen "Vor-Plätzen" müssen hinreichend viele Markierungen verfügbar sein
- In allen "Nach-Plätzen" darf die maximale Kapazität nach Schalten der Transition nicht überschritten werden

Schaltfunktion $m(r) = m(r-1) + W \cdot u(r)$

2.4.4 Charakteristika von Petrinetzen

Erreichbarkeit (Reachability)

- Ein Zustand heißt erreichbar von einem Anfangszustand, falls es eine Sequenz vom Anfangszustand zu diesem exisitert
- Der Erreichbarkeitsgraph (Graph mit möglichen zuständen des Petrinetzes als Knoten und mit Transitionen beschrifteten gerichteten Kanten dazwischen.
  - ob gewünschte Zustände erreicht, bzw. unerwünschte Zustände nicht erreicht werden können
  - ob Vorgägner gefährlicher Zustände umgangen werden können
- ist NP Hart in Speicher und Zeit

Beschränktheit

- Ein Petrinetz heißt beschränkt (bounded), falls in keinem Platz (zu keinem Zeitpunkt) mehr als eine gewisse Maximalanzahl von k_i Markierungen vonhanden ist.
- Falls $k_i = 1$, nennt man das Petrinetz *sicher* (*safe*)

Verklemmung (Deadlock)

- Eine Verklemmung (deadlock) eines Petrinetzes ist eine Transition (oder eine Menge von Transitionen), die nicht schalten können.
- Diese können *nie* schalten.

Lebendigkeit (Liveness)

- Eine Transition heißt tod (dead, non-live), falls sie bei keiner Folgemarkierung mehr aktivierbar ist.
- Diese kann *nicht mehr* schalten.

3 Zeitkontinuierliche Simulierung und Simulation

3.1 Einleitung

3.1.1 Örtlich konzentrierter Systemzustand

Die bestimmende Gleichung einer Funktion einer unabhängigen Veränderlichen heißt gewöhnliche Differentialgleichung, d.h. die Gleichung hängt nur von der Funktion selber und ihren Ableitungen ab.

Systeme mit örtlich konzentrierten Systemzustand werden durch gewöhnliche Differentialgleichungen beschrieben.

3.1.2 Örtlich verteilter Systemzustand

Die bestimmende Gleichung einer Funktion von mehr als einer unabhängigen veränderlichen heißt partielle Differentialgleichung.

Systeme mit örtlich verteilten Systemzustand werden durch partielle Differentialgleichungen beschrieben.

3.2 Beschreibung zeitkontinuierlicher Systeme

3.2.1 Allgemein

Beschreibung der Begriffe siehe Abschnitt 1.2 auf Seite 2.

- Im Nachfolgenden werden Vektorpfeile der Einfachheit halber weggelassen.

Zustandsgleichung / Systemfunktion

$$\begin{aligned}\dot{x}(t) &= \frac{dx(t)}{dt} \\ &= \begin{pmatrix} \dot{x}_1(t) \\ \vdots \\ \dot{x}_n(t) \end{pmatrix} \\ &= \begin{pmatrix} f_1(x(t), u(t), t) \\ \vdots \\ f_n(x(t), u(t), t) \end{pmatrix} \\ &= f(x(t), u(t), t)\end{aligned}$$

Ausgangsgleichung

$$\begin{aligned}y(t) &= g(x(t), u(t), t) \\ &= \begin{pmatrix} g_1(x(t), u(t), t) \\ \vdots \\ g_m(x(t), u(t), t) \end{pmatrix}\end{aligned}$$

Zustandsgrößen $x = (x_1, \ldots, x_n)^T$

Stellgrößen $u = (u_1, \ldots, u_l)^T$

Ausgangsgrößen $y = (y_1, \ldots, y_m)^T$

3.2.2 Verschiedene Systemdynamiken

Höhere Ordnung Jedes System von Differentialgleichungen höherer Ordnung kann auf ein System 1. Ordnung transformiert werden.
Hierzu wird bei einem System k-ter Ordnung $\left(x_1 = x,\ x_2 = \dot{x},\ x_3 = \ddot{x}, \ldots,\ x_k = x^{(k)}\right)^T$ als Systemzustandmit entsprechender Systemfunktion genommen.

autonomes System Ein System das nicht explizit von t abhängt wird als *autonom* bezeichnet.
Jedes System lässt sich durch Hinzunahme von t zu den Systemzuständen mit $\dot{t} = 1$ in ein autonomes System transformieren.

lineare Systemdynamik hat folgende Gestalt

$$\begin{aligned}\dot{x} &= f(x, u) = Ax + Bu \\ y &= g(x, u) = Cx + Du\end{aligned}$$

mit konstanten (oder rein zeitabhängigen) Koeffizientenmatrizen

$$\begin{aligned} A &\in \mathbb{K}^{n \times n} & B &\in \mathbb{K}^{n \times l} \\ C &\in \mathbb{K}^{m \times n} & D &\in \mathbb{K}^{m \times l}\end{aligned}$$

3.3 Modellanalyse

3.3.1 Lösbarkeit

Lösungstrajektorie Man zeichne in ein x, t Diagramm (x kann Dimension größer 1 haben, dann dafür mehrere Achsen verwenden) in jeden Punkt einen Pfeil mit der Steigung $f(x, t) = \dot{x}$ ein (*Richtungsfeld*). Wenn man nun im Punkt $x(0)$ (n Integrationskonstanten) startet und immer den Pfeilrichtungen folgt, erhält mein eine *Lösungstrajektorie*.

eindeutige Lösung $x(t)$ mit $t > 0$ für ein autonomes DGL-System 1ter Ordnung existiert falls für alle (zulässigen) $x_1, x_2 \in \mathbb{K}^n$ die folgende Lipschitzbedingung erfüllt ist. Es exisitert ein $L \in \mathbb{K}$ so dass für alle (zulässigen) $x_1, x_2 \in \mathbb{K}^n$ gilt

$$\|f(x_1) - f(x_2)\| \leq L \|x_1 - x_2\|$$

mit einer belibigen Norm $\|.\|$.

3.3.2 Gleichgewichtslösung

Außer der Existenz einer (eindeutigen) Lösung von

$$\dot{x} = f(x,u),\ x(0) = x_0 \in K^n$$

ist die Existenz eines *stationären Zustandes* $x(t) \to x_s$ für $t \to \infty$ für konstantes $u(t) = u_s$ von Interesse.

Notwendig $0 = f(x_s, u_s)$

Lineares System $Ax_s = -Bu_s$ eindeutig Lösbar mit $x_s = -A^{-1}Bu_s$ wenn $\det(A) \neq 0$

- Lösbar sobald $\text{Rang}(A) = \text{Rang}(A|Bu_s)$

Nichtlineares System ist zudem notwendig das die Determinante der Jakobimatrix im Lösungsumfeld (x_s) nicht 0 ist.

$$\exists\varepsilon : \forall \|x - x_s\| < \varepsilon : \det \frac{\partial f(x, u_s)}{\partial x} \neq 0$$

- Es kann keine, eine, mehrere, unendlich viele oder keine Lösung geben.

3.3.3 Jacobi-Matrix

Definition

$$\frac{\partial f}{\partial x} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

Vorwärtsdifferentenquotient

- Approximation der j-ten Spalte der Matrix

$$\frac{\partial f}{\partial x_j} \approx \frac{1}{\delta(x_j)} \left(f(x + e_j\delta(x_j)) - f(x)\right)$$

- e_j ist der j-te Einheitsvektor
- Schrittweite $\delta(x_j) = \varepsilon(1 + |x_j|)$
- Wahl der Toleranz ε bei "gut skalierter" Funktion f: $\varepsilon = \sqrt{\varepsilon_{mach}}$ mit ε_{mach} relative *Maschienengenauigkeit*, z.B. $\varepsilon_{mach} \approx 2.2 * 10^{-16}$ bei double precision
- Aufwand: Approximation erfordert im wesentlichen $n+1$ Auswertungen von f
- Genauigkeit:

$$\left|\frac{\partial f_i(x)}{\partial x_j} - \frac{1}{\delta(x_j)}\left(f_i(x + e_j\delta(x_j)) - f_i(x)\right)\right| \leq |\delta(x_j)| \cdot \left|\frac{\partial^2 f_j(x)}{\partial x_j^2}\right|$$

  d.h. maximal die Hälfte der gültigen Dezimalstellen von f

Besetztheisstruktur einer Jacobi-Matrix gibt wieder, in welchen Stellen der Matrix von 0 verschiedene Werte stehen.

- Dies gibt die Kopplungen der DGL wieder
- sehr große Systeme können nur durch Ausnutzung der (hoffentlich dünnen) Besetztheitsstruktur der Jakobimatrix effizient numerisch simuliert werden.

Schrittweise Aufdatierung ("Updates") einer Approximation der Jacobi-Matrix (sogenanntes "Quasi Newton Verfahren")

- im Allgemeinen: Rang 1 Verfahren z.B. nach Broyden:
  Approximation der Jacobimatrix $J^{(k)} \approx \frac{\partial f}{\partial x}\left(x^{(k)}\right)$ durch schrittweise Aufaddierung einer $n \times n$ Matrix vom Rang 1 (rank 1 update)

$$J^{(k)} = J^{(k-1)} + \frac{1}{\left\|\alpha^{(k-1)} \cdot \Delta x^{(k-1)}\right\|} \cdot \underbrace{f\left(x^{(k)}\right) - f\left(x^{(k-1)}\right)}_{n\times 1 \text{ Spaltenvektor}} \cdot \underbrace{\alpha^{(k-1)} \cdot \Delta x^{(k-1)\,T}}_{1\times n \text{ Zeilenvektor}}$$

- $\alpha^{(k-1)} \cdot \Delta x^{(k-1)} = x^{(k)} - x^{(k-1)}$
- Bei symmetrischer Jacobi-Matrix: Rang 2-Verfahren

Symbolisches Differenzieren durch systematisches Anwenden von Ketten-, Produkt etc. Regeln

- z.B. mit Maple, Mathematica, Macsyma
- Vorteil: kein Approximierungsfehler, nur Rundungsfehler
- Nachteil: liefert lange, unübersichtliche Formeln; hoher Berechnungsaufwand

Automatisches Differenzieren sind spezielle Algorithmen zur Transformation eines C-, Fortran oder Matlab- Programms von $f(x)$ in Programm zur Auswertung der Jacobi-Matrix

- Vorteil: kein Approximatinsfehler, nur Rundungsfehler
- Schwierigkeiten u.a. bei Fallunterscheidungen (if-then-else)

3.3.4 Linearisierung um die Ruhelage

$$(\Delta x)^{\bullet} \approx \underbrace{\left.\frac{\partial f}{\partial x}\right|_{x_s,u_s}}_{A} \cdot \Delta x + \underbrace{\left.\frac{\partial f}{\partial u}\right|_{x_s,u_s}}_{B} \cdot \Delta u$$

- x_s, u_s sind Koordinaten der jeweiligen Ruhelage um die linearisiert wird
- Fällt ein Ruhezustand mit einer Sprung- oder Knickstelle (oder anderen Ausnahmestelle) zusammen, so kann man um ihn nicht linearisieren (und auch nicht in einer "engeren" Umgebung).
- Es lässt sich auch um eine Referenztrajektorie linearisiern. Formeln sehen genauso aus wie hier. Interpretierbar als Linearisierung um zeitveränderliche Ruhelage

3.3.5 Lösen einer linearisierten (linearen) DGL

Gegeben $\dot{x} = A \cdot x$

Lösungsansatz $x(t) = c \cdot e^{\lambda t}$

- $x, c \in \mathbb{C}^n \ \lambda \in \mathbb{C}$

Zwischentherm $(\lambda \cdot E - A) \cdot c = 0$

- Eigenwerte von A als Lösungen für λ bestimmen. n komplexe Lösungen für folgende Gleichung:

$$\det(\lambda_i \cdot E - A) = 0$$

Lösung $x(t) = \sum_{i=1}^{n} c_i e^{\lambda_i t}$

- λ_i sind Eigenwerte - n Stk.
- c_i sind Eigenvektoren - Anhand von Anfangsbedingungen genauer zu bestimmen - n Stk.
- Nur falls A diagonalisierbar. Ansonsten Jordanketten Bilden:
  - Hauptvektoren

$$(\lambda E - A)\, v^{(k)} = v^{(k-1)}$$

$$e^{\lambda_i t}\left(c_1 v_1 + c_2\left(v_2 + t v_1\right) + c_3\left(v_3 + t v_2 + t^2 v_1\right)\right)$$

3.3.6 Stabilität

Für den Fall das A eine rein reelle Matrix war gilt:

aperiodische Dämpfung wenn alle Eigenwerte reell und negativ sind

- Stabil

aperiodisch ungedämpft wenn alle Eigenwerte reell und mind. einer positiv ist

- instabil

gedämpfte Oszilation wenn alle Realanteile der Eigenwerte negativ sind und komplexe Eigenwerte immer konjungiert nocheinmal auftauchen.

- stabil

ungedämpfte Oszillation wenn mindestens ein Realanteil eines Eigenwertes positiv ist und komplexe Eigenwerte immer konjungiert nocheinmal auftauchen.

- instabil

instabil bedeutet das bei einer begrenzen Eingabe eine theoretisch unbegrenzte Ausgabe erfolgen kann.

3.3.7 Zeitcharakteristik

Zeitcharakteristika T_i

- sind für aperiodische Vorgänge die Dauer, bis sich der Wert um den Faktor e (bzw. $\frac{1}{e}$) verändet hat
- sind für periodische Vorgänge die Periodendauer
- können mit Hilfe der Eigenwerte λ_i bestimmt werden

reeller EW $T_i = \frac{1}{|\lambda_i|}$

imaginärer EW $T_i = \frac{2\pi}{|\lambda_i|}$

konjungiert komplexer EW

$$T_i = \min\left\{\frac{1}{|\Re(\lambda_i)|}, \frac{2\pi}{|\Im(\lambda_i)|}\right\}$$

Minmale Zeitcharakteristik

$$T_{min} = \min\{T_i | i = 1, \ldots, n\}$$

Maximale Zeitcharakteristik

$$T_{max} = \max\{T_i | i = 1, \ldots, n\}$$

Simulationsdauer für nichtlineares System (Faustregel)

- stabiles System: $t_f = 5 \cdot T_{max}$
- instabiles System bis z.B. ein gewünschter Wert über-/unterschritten wurde

Diskretisierungsschrittweite $h = \Delta t \leq \alpha \cdot T_{min}$

- mit $\alpha \in \left[\frac{1}{20}, \frac{1}{5}\right]$

3.3.8 Steife DGL

Definition Eine DGL wird steif genannt wenn sie sehr unterschiedliche Zeitcharakteristika enthält

- $s > 10^3 \dots 10^7$

Steifheitsmaß $s = \frac{T_{max}}{T_{min}}$

- numerische Lösung benötigt Rechenzeit Proportional zu s

Integration von steifen DGLn am besten mit implizieten Verfahren

3.3.9 Unstetige Rechte Seite

Die rechte Seite f der Differentialgleichung $\dot{x} = f(x, u)$ (oder eine ihrer Ableitungen) kann *unstetig* sein, z.B. wegen

- unstetiger Steuerung $u(t)$, z.B. Ventil auf/zu
- modellinhärenter Unstetigkeiten (z.B. Kollision von Roboterarm, Abtrennung einer Raketenbrennstufe)
- Einige numerische Methoden benötigen (mehrfach) stetig ableitbare rechte Seite. Hier Vorsicht walten lassen bzw. berücksichtigen.
- Verbesserung der numerischen Simulation durch Detektion der Unstetigkeit und abschnittsweiser Berechnung.

3.4 Grundlagen der numerischen Simulation

3.4.1 Zahlendarstellung

Normalisierte Gleitpunktzahl Zahl bei dem genau eine Stell vor dem Komme ungleich 0 ist. Die entsprechende Kommaverschiebung wird im Exponenten des Faktors untergebracht.

$$\underbrace{3.001109}_{Mantisse} \cdot \underbrace{10}_{Basis}^{\overbrace{9}^{Exponent}}$$

$$z = \pm \left(d_1 B^0 + d_2 B^{-1} + \ldots + d_t B^{-t+1} \right) B^E$$

Basis B feste Zahl, z.B. 10 (Dezimal) oder 2 (Binär)

- bei Basis $B = 2$ braucht d_1 nicht expliziet gespeichert werden, da es durch die Normalisierung 1 sein *muss*

Exponent E ganze Zahl und beschränkt $E_{min} \leq E \leq E_{max}$

- mehr Bits für den Exponenten ergibt größeren Zahlenbereich

Ziffern $d_i \in \{0, 1, 2, \ldots, B-1\}$

- z normalisiert wenn $d_1 \neq 0$

Mantisse $M = d_1 d_2 \dots d_t$

- Länge der Mantisse t (konstant)
- mehr Bits für die Mantisse ergibt größeren (relative) Genauigkeit

Eigenschaften

- Es gibt eine größte und eine kleinste darstellbare Zahl
- Auf heutigen Computer ist $B = 2$ quasi Standard
- bei Basis $B = 2$ braucht d_1 nicht expliziet gespeichert werden, da es durch die Normalisierung 1 sein *muss*
- Es wird noch ein weiteres Bit für das Vorzeichen gebraucht: s

Binär

- $z = (-1)^s \cdot (1 + M) \cdot 2^E$
- Einige besondere Belegungen sind reserviert für Unendlich und co: siehe IEEE 754

relative Maschienengenauigkeit (machine precision) heißt der Abstand der Zahlen im Intervall $[1, 2]$

- IEEE single precision $eps = 2^{-23} \approx 1,19 \cdot 10^{-7}$
- IEEE double precision $eps = 2^{-52} \approx 2,2 \cdot 10^{-16}$

3.4.2 Rundungsfehler

Rundung $\mathrm{rd}(g)$ Abbilden einer Zahl auf die "am nächsten liegende" darstellbare Zahl

$$\forall g : |x - \mathrm{rd}(g)| \leq |x - g|$$

- Es gibt unterschiedliche Rundungsarten nach IEEE 754
  - R1 immer aufrunden (nach rechts)
  - R2 immer abrunden (nach links)
  - R3 Runden durch abschneiden (nach Null)

  - R4 Runden zur nächsten Zahl (zum Nächsten)

relative Rundungsfehler $\varepsilon = \frac{\mathrm{rd}(x)-x}{x}$

- $\mathrm{rd}(x) = x \cdot (1+\varepsilon(x))$
- mit $|\varepsilon(x)| \leq \mathrm{eps}$
- Für die elementaren arithmetischen Operationen $(+,-,\cdot,)$ gilt zumindest näherungsweise $f_{gerundet}(x,y) \approx f(x,y) \cdot (1+\varepsilon_f)$ mit $|\varepsilon_f| \leq \mathrm{eps}$

3.4.3 Fortpflanzung von Rundungsfehlern

- Assoziativ- und Distributivgesetz gelten nicht für Gleitpunktarithmetik:

  $$\mathrm{gl}(\mathrm{gl}(x+y)+z) \neq \mathrm{gl}(x+\mathrm{gl}(y+z))$$

  (gl ist Gleitpunktimplementierung der entsprechenden Funktion)
- In allen Ausdrücken

  $$\mathrm{gl}(x \diamond y) = (x \diamond y) \cdot (1+\varepsilon)$$

  sukzessive ersetzen.
  Daraus Ermittlung einer Darstellung

  $$\mathrm{gl}(f) = f \cdot (1 + (\ldots)\varepsilon_1 + (\ldots)\varepsilon_2 + \ldots)$$

  mit von Eingangsdaten abhängigen Verstärkungsfaktoren

3.4.4 Kondition

Gegeben $f : D \subseteq \mathbb{R}^n \to \mathbb{R}^m$ sei mindestens einmal stetig differenzierbar

absuluter Fehler in x

$$\Delta x_j = |\tilde{x}_j - x_j|$$

absuluter Fehler in y

$$\Delta y_j \quad = \quad |\tilde{y}_j - y_j| = |f_i(\tilde{x}) - f_i(x)|$$

relativer Fehler in x $\varepsilon_{x_j} = \frac{\Delta x_j}{x_j}$

relativer Fehler in y

$$\varepsilon_{y_j} = \frac{\Delta y_j}{y_j} = \sum_{k=1}^{n} \underbrace{\frac{x_k}{f_j(x)} \frac{\partial f_j(x)}{\partial x_k}}_{Konditionszahlen} \cdot \varepsilon_{x_k}$$

Konditionszahlen (siehe vorherige Formel) nennt man die Verstärkungsfaktoren des relativen Fehlers in den Eingangsdaten.

- sind diese groß ist das Problem schlecht Konditioniert
- sind diese klein ist das Problem gut Konditioniert
- hängen *nicht* von der Implementierung der Funktion ab, sondern nur von deren Eigenschaften

  Multiplikation $\varepsilon_\cdot = \varepsilon_{x_1} + \varepsilon_{x_2}$

  Division $\varepsilon_/ = \varepsilon_{x_1} - \varepsilon_{x_2}$

  Addition / Subtraktion

  $$\varepsilon_\pm = \frac{x_1}{x_1 \pm x_2}\varepsilon_{x_1} - \frac{x_2}{x_1 \pm x_2}\varepsilon_{x_2}$$

  - Subtraktion (speziell von gleich großen Zahle) ist extrem schlecht Konditioniert: *Auslöschung*

  Wurzel $\varepsilon_{\sqrt{\cdot}} = \frac{\varepsilon_{x_1}}{2}$

Numerische Stabilität ein gut Konditioniertes Problem kann bei schlechter Implementierung trotzdem extreme Fehlerverstärkung haben

- Formel Umformen! Umständlich kann besser als direkt sein! Subtraktionen vermeiden!

3.5 Berechnung nichtlinearer Gleichgewichtslösungen

3.5.1 Newton Verfahren - Basisversion

Gesucht x_s mit $0 = f(x_s)$

Start $(k=0)$ Wahr eines Startvektors $x^{(0)}$

- z.B. aus Kentnissen über das dem Modell zugrundelegene physikalische Modell
- Ausprobieren
- Herantasten von der bekannten Lösung eines einfacheren Problems (*Fortsetzungsmethode*)

Iterationsschritt $(k \to k+1)$

- Berechnung von $f\left(x^{(k)}\right)$
- Berechnung der Jakobimatrix $\frac{\partial f}{\partial x}\left(x^{(k)}\right)$
  - siehe Abschnitt 3.3.3 auf Seite 10
  - oder Jakobimatrix durch *konstante Matrix*, z.B. $\frac{\partial f}{\partial x}\left(x^{(0)}\right)$ bzw. deren Approximation ersetzen
    - Verfahren nicht mehr quadratisch, höchstens linear konvergent
    - Wenn konvergent, dann heufig schneller als “normales” Newtonverfahren

- Berechnung der Korrektur (Suchrichtung) $\Delta x^{(k)}$ durch lösen des linearen Gleichungssystems

$$\frac{\partial f}{\partial x}\left(x^{(k)}\right) \cdot \Delta x^{(k)} = -f\left(x^{(k)}\right)$$

- Berechnung der neuen Näherung $x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$

Terminierungstest Falls "nahe genug" an der Lösung oder "kein Fortschritt" *Stop*, ansonsten k um 1 erhöhen und nächsten Iterationsschritt.

- "Nahe genug" an der Lösung
  - keine nennenswerte Änderung in $x^{(k+1)}$

$$\left\|\Delta x^{(k)}\right\| \leq \varepsilon_1 \cdot \left(1 + \left\|x^{(k)}\right\|\right)$$

  - Funktionswerte klein genug

$$\left\|f\left(x^{(k+1)}\right)\right\| \leq \varepsilon_2$$

- "kein Fortschritt" mehr
  - keine nennenswerte Abnahme mehr

$$\left\|f\left(x^{(k+1)}\right)\right\| \geq \left\|f\left(x^{(k)}\right)\right\| - \varepsilon_3$$

  - zu viele Iterationsschritte

$$k + 1 \geq k_{max}$$

- Für $\varepsilon_j, j = 1, 2, 3$ geeignete Toleranzen z.B. $\varepsilon_j = 10^{-4}, \ldots, 10^{-8}$

3.5.2 Problembereiche

Einzugsbereich bei mehreren Lösungen

- Es können einige Nullstellen durch andere in Ihrere Umgebung versteck sein, so dass sie nur gefunden werden wenn der Startwert in einem (kleinen) Bereich liegt.

Divergenz und Singularitäten

- Singularitäten in der Nähe von Nullstellen können diese "Verstecken"

hoch Nichtlineare Probleme

- Divergenz obwohl Startwert "nahe" an Nullstelle

Abhilfe durch "kleineren" Schritt zur Berechnung von $x^{(k+1)}$

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} \cdot \Delta x^{(k)}$$

- mit $0 < \alpha_{min} \leq \alpha^{(k)} \leq 1$

- Bestimmung der Schrittweite durch Näherungsweise Minimierung von

$$\begin{aligned} \varphi(\alpha) &= \left\|f\left(x^{(k)} + \alpha \cdot \Delta x^{(k)}\right)\right\|_2^2 \\ &= \sum_{i=1}^{n} \left(f_i\left(x^{(k)} + \alpha \cdot \Delta x^{(k)}\right)\right)^2 \end{aligned}$$

- Fausregel: α nur so "klein wie nötig" und "so groß wie möglich", denn nur für $\alpha \approx 1$ quadratische Konvergenz

3.6 Numerische Lösung der nichtlinearen Zustandsdifferentialgleichungen

3.6.1 Einleitung

Bahn (Pfad, Weg) kontinuierlich, räumliche Punktfolge

Trajektorie Bahn mit "zeitlichen Beschränkungen"

- z.B. außer Position auch Geschwindigkeit (und Beschleunigung) an jedem Punkt der Bahn (evtl. Impliziet über t Achse) gegeben

Einsatzgebiet der numerischen Lösung überall dort wo nichtlinearitäten vorliegen bzw. keine expliziete Lösung gefunden wird (werden kann)

Klassen Numerischer Integrationsverfahren jeweils explizit und implizit:

- Einschrittverfahren - nur diese sind hier erwähnt
- Mehrschrittverfahren
- Extrapolationsverfahren

Kriterien für Integratorenwahl

- Rechenaufwand (Berechnungseffizienz)
- Genauigkeit (Approximationsfehler)
- Eignung für steife Systeme (Stabilität). Bei steifen DGL impliziete Verfahren bevorzugen!
- Implementierungsaufwand (Eigenprogrammierung oder Verwendung einer Bibliothek)

Ordnung des Verfahrens (n)

- Approximationsfehler bei gegebener Schrittweite h_k für Methoden höhere Ordnung ist sehr viel kleiner: $O\left(h^n\right)$
- Berechnungsaufwand dafür höher: n Funktionsauswertungen von f
- Rechte Seite muss mindesten n mal stetig differenzierbar sein.
  - Wenn es nur $m < n$ mal stetig differenzierbar ist bleibt der Berechnungsaufwand gleich, aber der Approximationsfehler liegt nur noch bei $O\left(h^m\right)$
- Diese Genauigkeitsaussagen gelten nicht für sehr große oder sehr kleine Schrittweiten. Bei sehr kleinen dominieren Rundungsfehler.

3.6.2 Einschrittverfahren

Gegeben $x_k \approx x\left(t_k\right)$

Gesucht $x_{k+1} \approx x\left(t_{k+1}\right)$

Ansatz für Einschrittverfahren

$$x_{k+1} = x_k + h \cdot \Phi\left(t_k, x_k, x_{k+1}, h, f\right)$$

Konsistenzbedingung für Einschrittverfahren

$$\lim_{h \to 0} \Phi\left(t_k, x_k, x_{k+1}, h, f\right) = f\left(x_k\right)$$

3.6.3 Explizites Euler Verfahren

$$x_{k+1} = x_k + h \cdot f_k$$

- Rechenaufwand gering, nur 1 Auswertung pro Schritt. Für Echtzeitsysteme interessant
- Genauigkeit: Approximationsfehler $O\left(h^1\right)$
  - Für hohe Genauigkeiten kleine Schrittweite nötig $\Rightarrow$ Stabilitätsprobleme
- Schrittweite $h = \Delta t$
- Gesamtfehler $\varepsilon_k = \left|x_k - x\left(k \cdot h\right)\right|$
  - Bei zu kleiner Schrittweite dominieren Rundungsfehler
  - Bei zu großer Schrittweite dominieren Einzelschritt- und Fortpfanzungsfehler
  - $\Rightarrow$ Notwendigkeit für Adaptive Schrittweitensteuerung
- Einzelschritt- und Fortpflanzungsfehler

$$\varepsilon'_k \leq \sum_{i=1}^{k} \varepsilon_i^E + \sum_{i=1}^{k-1} \varepsilon_i^F$$

  - ε_i^E Einzelschrittfehler
  - ε_i^E Fortpflanzungsfehler

3.6.4 Implizites Euler Verfahren

$$x_{k+1} = x_k + h \cdot f_{k+1}$$

- Ein Verfahrensschritt erfordert die (näherungsweise) Lösung der nichtlinearen Gleichung

$$0 = x_{k+1} - x_k - h_k \cdot f\left(x_{k+1}\right)$$

  nach x_{k+1}
- Rechenaufwand: sehr hoch
- Genauigkeit: Approximatinsfehler $O\left(h^1\right)$
  - Für hohe Genauigkeit kleine Schrittweite nötig, aber bessere Stabilitätseigenschaften als explizites Euler Verfahren

3.6.5 Heun Verfahren

- Ist ein Prädiktor-Korrektor Verfahren
  - Prädiktor Schritt:

$$x_{k+1}^P = x_k + h_k \cdot f_k$$

  - Kottektor Schritt: Mittelung des Gradienten

$$x_{k+1} = x_k + h_k \cdot \frac{1}{2}\left(f_k + f\left(x_{k+1}^P\right)\right)$$

- Insgesamt:

$$\begin{aligned} s_1 &= f\left(x_k\right) \\ s_2 &= f\left(x_k + h_k \cdot s_1\right) \\ x_{k+1} &= x_k + \frac{h_k}{2}\left(s_1 + s_2\right) \end{aligned}$$

- Ist ein zweistufiges Einzelschrittverfahren - Rechenaufwand zwei Funktionsauswertungen von f
- Genauigkeit: Approximationsfehler $O\left(h^2\right)$
  - Heun Verfahren ist 2. Ordnung

3.6.6 Runge-Kutta-Verfahren 4. Ordnung

Allgemeiner Ansatz

$$\begin{aligned} s_1 &= f\left(x_k\right) \\ s_2 &= f\left(x_k + h \cdot \alpha_1 \cdot s_1\right) \\ s_3 &= f\left(x_k + h \cdot \beta_1 \cdot s_1 + h \cdot \alpha_2 \cdot s_2\right) \\ s_4 &= f\left(x_k + h \cdot \gamma_1 \cdot s_1 + h \cdot \beta_2 \cdot s_2 + h \cdot \alpha_3 \cdot s_3\right) \\ x_{k+1} &= x_k + h \cdot \underbrace{\left(\delta_1 \cdot s_1 + \delta_2 \cdot s_2 + \delta_3 \cdot s_3 + \delta_4 \cdot s_4\right)}_{\Phi} \end{aligned}$$

man versucht nun die unbekannten Koeffizienten

$$\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \gamma_1, \delta_1, \delta_2, \delta_3, \delta_4$$

so zu bestimmen, dass

- die Konsistenzbedingung

$$\lim_{h \to 0} \Phi\left(t_k, x_k, x_{k+1}, h, f\right) = f\left(x_k\right)$$

  erfüllt ist:

$$\Rightarrow \delta_1 + \delta_2 + \delta_3 + \delta_4 = 1$$

- der Approximationsfehler möglichst klein wird (d.h. $O\left(h^4\right)$

klassisches RK-Verfahren 4. Ordnung

$$\begin{aligned} s_1 &= f\left(x_k\right) \\ s_2 &= f\left(x_k + \frac{h}{2} \cdot s_1\right) \\ s_3 &= f\left(x_k + \frac{h}{2} \cdot s_2\right) \\ s_4 &= f\left(x_k + h \cdot s_3\right) \\ x_{k+1} &= x_k + \frac{h}{6} \cdot \left(s_1 + 2 \cdot s_2 + 2 \cdot s_3 + s_4\right) \end{aligned}$$

- Rechenaufwand: vier Auswertungen von f
- Genauigkeit: Approximationsfehler $O\left(h^4\right)$

3.6.7 Schrittweitensteuerung

Konstante Schrittweite h_k

- ungenau, wenn gesuchte Lösung sich lokal sehr stark ändert
- ineffizient, wenn gesuchte Lösung sich lokal sehr wenig ändert

Idee der Schrittweitensteuerung

- Wahl von h_k "so groß wie möglich" und "so klein wie nötig" und das neu von Schritt zu Schritt
- Wahl von h_k, so dass globaler Approximationsfehlers nach einem Schritt unterhalb einer Fehlerschranke liegt

Einschrittverfahren der Ordnung p

- Berechnung von 2 Näherungen für x_{k+1}: einmal für Schrittweite h_k und einmal für Schrittweite $\frac{h_k}{2}$
- Daraus Schätzung des lokalen Fehlers
- Anpassung von h_k so dass lokale Fehlerschätzung unter Schranke liegt.
- Aufwand: $3p$ Funktionsauswertungen

zwei Verfahren unterschiedlicher Ordnung

- z.B. p und $p+1$
- Berechnung von 2 Näherungen für x_{k+1} für dieselbe Schrittweite h_k
- Bei geeigneter "Einbettung" der beiden Verfahren, z.B. Runge-Kutta-Verfahren 4. und 5. Ordnung zusammen nur $p+1$ Funktionsauswertungen nötig
- Ansatz viel effizienter als "Einschrittverfahren der Ordnung p"

3.6.8 Klassifikation zeitkontinuierlicher Simulationswerkzeuge

Level 0

- Direkte Programmierung
  - Fortran
  - C
  - Pascal
  - Matlab

Level 1

- Simulationssprachen
  - ASCL
  - VHDL
  - Dare-P
  - Desire
- Simulationsframeworks
  - Simulink
  - C++ Klassenbibliothek

Level 2

- Graphische Modellierung
  - Simulink
  - WorkingModel
  - Aspen
  - Stella
- Spezialsimulatoren
  - KSIM

Level 3

- Multidisziplinäre Modellgenerierung
  - Modellica
  - VHDL-A

3.7 Integration von Zustands-DGLn mit Unstetigkeiten

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

Annahme

- $f(x, u, t)$ sei abschnittsweise mehrfach stetig differenzierbar
- An den Übergängen der Abschnitte ist f möglicherweise unstetig, bzw. nicht häufig genug stetig differenzierbar
- Die Übergänge werden durch formulierbare Ereignisse (events) ausgelöst, die zustandsabhängig oder zeitgesteuert sein können

Schaltfunktion Die (Um-) Schaltpunkte (events) $t_{s,i}$, $i = 1, \ldots, n_S$ werden im Allgemeinen als (einfache) Nullstellen reellwertiger Schaltfunktionen

$$q_k(x(t_{s,i}), t_{s,i}) = 0,\ k \in \{1, \ldots, n_q\}$$

charakterisiert. Bei der numerischen Integration müssen nun die Vorzeichen der Schaltunktionen beobachtet werden, d.h.

- wenn zwischen der letzten berechneten Näherung $x_k = \tilde{x}(t_k)$ und der neuen Näherung $x_{k+1} = \tilde{x}(t_{k+1})$ ein Vorzeichenwechsel in (mind.) einer Schaltfunktion stattfindet,
- muss der dazwischenliegende erste Schaltzeitpunkt bestimmt werden (und zwar in der Genauigkeitsanforderung des Benutzers und der Ordnung des Verfahrens implizit gegebenen Genauigkeit)
  - Kompination von numerischer Integration und (eindimensionaler) Nullstellensuche
- Vorraussetzung: Schrittweite h_k klein genug, so dass kein doppelter Vorzeichenwechsel in derselben Schaltfunktion stattfindet.

4 Modulare Modellbildung und Simulation komplexer Systeme

4.1 Modulare Modellbildung

4.1.1 Modul

- Darstellung siehe Abbildung 6 auf der nächsten Seite.
- Zerlegung des zu modellierenden Systems in Module
- Jedes Modul ist über Schnittstellen S_i mit übrigen Modulen verbunden
- Schnittstellen zeigen aus Modulen heraus
- Export der Beschreibungsvariablen des Moduls über Schnitstellen S_i
- für jedes Modul eigene Modellgleichungen, die die Schnittstellenvariablen verknüpfen
- Verschaltung von Modulen durch Verknüpfungen von Schnittstellen in einem Knoten
  - dabei Unterscheidung zwischen Fluß- und Potentialvariablen

Flußvariablen Summe ist 0 in einem Knoten

$$\sum_{k=1}^{n} i_k = 0$$

Potentialvariablen Der Wert der Variablen ist gleich

$$u_1 = u_2 = \ldots = u_m$$

Konventionen

- Pro Schnittstelle je eine Potential und Flussvariable möglich
- Schnittstellen können zu Multiplexkanälen gebündelt werden
  - z.B. x, y, z Position (Potentialvariablen) und Kräfte (Flussvariablen)

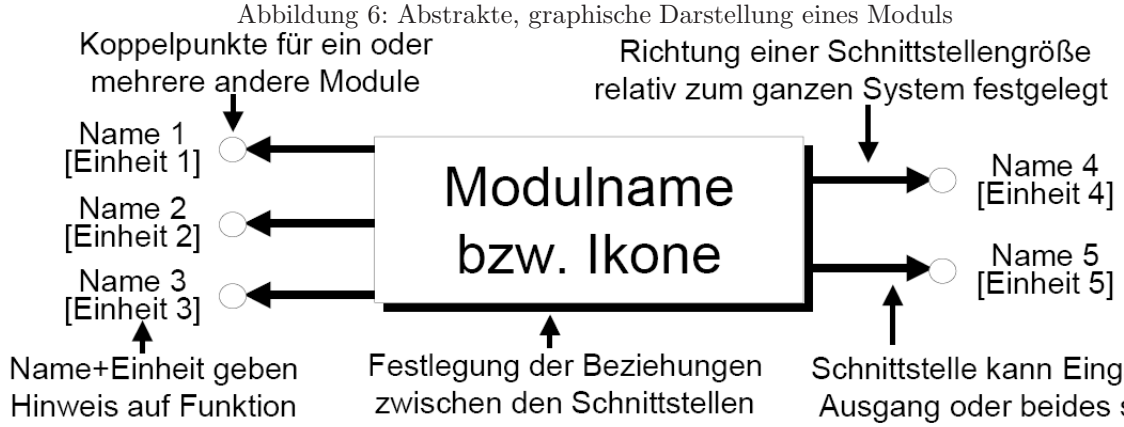
4.1.2 Automatische Gleichungsgenerierung

- Name der Globalen Variablen ist "Modulname.InternerName" bzw. "Modulname.SchnittstellenName"
- Export der Modellgleichung des Moduls mit Globalen Valiablennamen
- Hinzuname der Kopplungsgleichungen (für Potential und Flussvariablen)
- Das entstandene System nennt sich *Differentialalgebrasiches System* - DAE

4.1.3 Differentialalgebrasiches System

- Ein so automatisch aufgestelltes DAE System enthält im allgemeinen noch redundante Gleichungen. Dieses kann durch algebraische Umformungen noch reduziert und vereinfacht werden
- sehr große Zahl von Gleichungen entsteht
- viele Kopplungen leicht eliminierbar

Abbildung 6: Abstrakte, graphische Darstellung eines Moduls



- nichtlineare Gesetze unter Umständen nicht eleminierbar
- resultierendes DGL-System nach (theoretischer) vollständiger Elimination sehr unhandlich
  - hierzu z.B. einige Hilfsgleichungen (Modellgleichungen) Ableiten und ineinander einsetzen. ⇒ "Rumtricksen"
- Allgemeines DAE-System

$$\begin{aligned} \dot{x}_d &= f(x_d, x_a) \\ 0 &= h(x_d, x_a) \end{aligned}$$

  - x_d differentielle Variablen
  - x_a algebraische Variablen
  - $\dim(f) = \dim(x_d)$
    $\dim(h) = \dim(x_a)$

4.1.4 Prinzipien der modularen Modellierung

- *Zerlegung* des Systems in Module mit Ein/Ausgangsschnittstellen
- Zuordnung einer *Potential/Flussvariablen* pro Schnittstelle
- Formulierung von *Gleichungen* pro Modul unter ausschließlicher Benutzung von Schnittstellen/Konstanten (evtl. interner temporärer Variablen - Zwischenergebnisse)
- *Kopplung* zweier Module über (kompatible) Schnittstellen induziert eine Potential- und eine Flussgleichung

4.2 Numerische Integration von Zustands-DAEs

4.2.1 Problemstellung

- Bisher betrachtete Bewegungsgleichungen eines dynamischen Systems
  - basieren auf einem minimalen Satz von Zustandsvariablen (*Minimalkoordinaten*)
  - führen auf Systeme gewöhnlicher DGLn (ordinary differential equations, *ODE*s), die numerisch effiziernt integriert werden können
  - aber nicht alle Systeme sind leicht in Minimalkoordinaten formulierbar
- Rechnergestützte, modulare Modellierung
  - führt auf Systeme differential-algebraischer Gleichungen (DAEs)
  - Integration von DAEs ist aufwendiger als von ODEs

4.2.2 Eigenschaften von DAEs

- DGLn mit Nebenbedingungen

$$\begin{aligned} \dot{x}_d &= f(x_d, x_a) \\ 0 &= h(x_d, x_a) \end{aligned}$$

- benötigt konsistente Anfangswerte, d.h.

$$\begin{aligned} x_d(0) &= x_{d,0} \\ x_a(0) &= x_{a,0} \\ 0 &= h(x_{d,0}, x_{a,0}) \end{aligned}$$

- Totale Differentiation der Nebenbedingung nach der Zeit t

$$0 = \frac{\partial h(x_d, x_a)}{\partial x_d} \cdot \underbrace{\frac{dx_d}{dt}}_{f(x_d, x_a)} + \frac{\partial h(x_d, x_a)}{\partial x_a} \cdot \frac{dx_a}{dt}$$

- Wenn Jakobimatrix von h nach x_a invertierbar ist, kann formal nach x_a aufgelöst werden

$$\dot{x_a} = -\left(\frac{\partial h(x_d, x_a)}{\partial x_a}\right)^{-1} \cdot \frac{\partial h(x_d, x_a)}{\partial x_d} \cdot f(x_d, x_a)$$

- Mit dieser und der Gleichung $\dot{x_d} = f(x_d, x_a)$ erhält man so wieder eine ODE die wie bisher gelöst werden kann

4.2.3 Index einer DAE

- Kriterium für die Schwierigkeit der numerischen Lösung einer DAE ist der sogenannte *Index*
- Es gibt verschiedene Index-Definitionen. Wichtigster und häufigster ist:
  Differentieller Index j einer DAE ist die kleinste Anzahl der nötigen Differentiationen, so dass

$$\frac{d^j}{dt^j} h\left(x_d, x_a\right)$$

  explizit nach $\dot{x_a}$ auflösbar ist.
- Je größer des Index ist, um so schwieriger ist die numerische Integration!
- Möglichkeit der *Bifurkation* (Verzweigung) der Lösung

4.2.4 Lösung mit ODE Löser und implizieten Gleichungslöser

- x_a als Funktion von x_d auffassen

$$0 = h\left(x_d\left(t\right), x_a\left(t\right)\right)$$

  und $x_d\left(t\right)$ bekannt $\Rightarrow x_a = x_a\left(x_d\right)$
- ODE-Integrationsverfahren kombiniert mit nichtlinearen Gleichungslöser
- Besser: Spezailverfahren für DAEs

4.2.5 Drift

- Bei numerischer unvermeidlicher Abweichung von exakter Lösung für x_a "driftet" die numerische Lösung ab
- Abhilfe: (impliziter) Prädiktor/Korrektor-Ansatz

4.2.6 Impliziter DAE-Lösungsansatz

- Impliziter Ansatz (z.B. Euler)

$$\begin{aligned} \frac{x_d(t+\Delta t)-x_d(t)}{\Delta t} &= f\left(x_d\left(t+\Delta t\right), x_a\left(t+\Delta t\right)\right) \\ 0 &= h\left(x_d\left(t+\Delta t\right), x_a\left(t+\Delta t\right)\right) \end{aligned}$$

- Numerische Lösung mit Newton-Verfahren nach $x_d\left(t+\Delta t\right), x_a\left(t+\Delta t\right)$

$$\begin{aligned} 0 &= \Delta t \cdot f(x_d(t+\Delta t), x_a(t+\Delta t)) - x_d(t+\Delta t) - x_d(t) \\ 0 &= h(x_d(t+\Delta t), x_a(t+\Delta t)) \end{aligned}$$

- Benötigt Invertierbarkeit der entsprechenden Jakobimatrix:

$$\begin{pmatrix} \Delta t \frac{\partial f}{\partial x_d} - E & \Delta t \frac{\partial f}{\partial x_a} \\ \frac{\partial h}{\partial d} & \frac{\partial h}{\partial a} \end{pmatrix}$$

  - Vorraussetzung: Index 1
  - DAEs vonhöherem Index müssen auf Index 1 reduziert werden

4.2.7 Weiterer Ansatz für DAE-Integratoren

- Einbettung in steifes ODE-System für ε sehr klein

$$\begin{aligned} \dot{x}_d &= f\left(x_d, x_a\right) \\ \varepsilon \cdot \dot{x}_a &= h\left(x_d, x_a\right) \end{aligned}$$

- Untersuchung von (implizieten) Runge-Kutta und Extrapolationsverfahren für Grenzfall $\varepsilon = 0$ liefern effiziente und zuverlässige DAE-Integratoren
- Sehr verbreitet sind auch implizite Mehrschrittverfahren: z.B.: BDF-Verfahren, DASSL

4.2.8 Visualisierung algebraischer Abhängigkeiten

- Hierbei handelt es sich um einen bipartiten Graphen mit gerichteten und ungerichteten Kanten
  - Für jede Gleichung wird das Gleichungsymbol in einen Kreis geschrieben und als Knoten verwendet
  - Für jede in den Gleichungen vorkommende Variable gibt es einen Knoten (aber ohne Umkreisung)
- Die gerichteten Kanten zwischen den Variablen und Gleichungen deuteten an, dass die Gleichung nach dieser Variablen auflösbar ist
- Falls eine Variable zwar in einer Gleichung vorkommt, diese aber nicht nach ihr auflösbar ist, wird eine ungerichtete Kante verwendet
- Alle Variablen, von denen mindestens zwei Formeln abhängen und mindestens eine Formel nach auflösbar ist lassen sich durch Gleichsetzen oder Einsetzen eliminieren
- Interessante Fragestellung: Lässt sich diesem Graphen der Index ansehen?

4.2.9 Prinzipielle Bearbeitungsschritte bei einer automatischen Modellgenerierung

1. System"schaltbild"
   - Gleichungsgenerierung
2. DAE-System
   - Elinination der Potential und Flussgleichungen
3. vereinfachtes DAE-System
   - Indexreduktion
4. Index-1-DAE-System
   - Partition und Schleifenreduktion

5. vereinfachtes Index1-DAE-System
    - ODE- bzw. DAE-Löser
6. numerische Lösung

Index