Name: Dominik Püllen Matrikelnummer 1912113 Computational Engineering 1.3.2013

Linearisierung am Ruhelage

1.) System auf 1. Ordnung bringen
2.) Ableitungen 0 setzen und auflösen -> Ergebnis = Ruhepunkte
3.) Ruhepunkte in Jacobi-Matrix einsetzen

Für Form des Vektors auf der linken Seite der Jacobi-Matrix

Herleitung:
- Sei s_0 eine Ruhelage
- Taylorentwicklung: $f(s_0+s) = f(s_0) + f'(s_0)s + \frac{1}{2}f''(s_0)s^2 + \dots$
- Näherung: $f(s_0+s) = f(s_0) + f'(s_0)s$
- $f(s_0) = 0$, da s_0 Ruhelage
- $\dot{s} = f'(s_0) \cdot s$

Einschränkungen der Linearisierung:
- Ruhezustand darf keine Sprung- oder Unstetigkeitsstelle sein
- Linearisierung nicht möglich, falls in Ruhelage eine Betragsfunktion ist

Stabilität

- Eigenwerte der Jacobi-Matrix berechnen: $\det(A - \lambda I) = 0$
- Realanteile betrachten:
  stabil: nur reelle, negative λ
  instabil: min ein $\lambda > 0$
  keine Aussage: min ein $\lambda == 0$
- (asymptotisch) stabil: Kurze Störung verursacht Abweichung aus Ruhelage. Dies geht aber gegen 0.

Steifheit

- explizite Integrationen brauchen unnötig viele Rechenzeiten -> besser implizite Verfahren
- wie steife Feder -> numerische Lösung von Kopplung v. steifem / weichem System ist schwierig
- Steifheit bei $\frac{T_{max}}{T_{min}} > 10^3 \dots 10^7$
- $T_{max} = \max(T_i)$ $T_{min} = \min(T_i)$
- Berechnung von T_i:
  - bei reellem λ: $T_i = \frac{1}{|\lambda_i|}$
  - bei imaginärem: $T_i = \frac{2\pi}{|\lambda_i|}$
  - bei konj. kompl. $T_i = \min\left\{\frac{1}{|Re(\lambda_i)|}, \frac{2\pi}{|Im(\lambda_i)|}\right\}$
- -> λ_i sind Eigenwerte

stabile DGL mit sehr unterschied. Zeitkonstanten

Simulationsdauer / Diskretisierungsschrittweite

- Stabiles System: $t_f \approx 5 \cdot T_{max}$ (bis zum annähernden Erreichen der Gleichgewichtslage)
- instabiles System: $|x(t_f)| \geq M$
- Diskretisierungsschrittweite:
  $h = \Delta t \leq \alpha \cdot T_{min}$
  $\alpha \approx \frac{1}{20}$ bis $\alpha \approx \frac{1}{5}$
  ↳ Diskretisierungsschrittweite für numerische Integration gebraucht

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

ε ist der relative Rundungsfehler
arithmetische Oper. $gl(x+y) = (x+y)(1+\varepsilon)$
Asso.-/Distributivgesetz gelten nicht
Auslöschung: Falsches Ergebnis bei Subtraktion fast gleich großer Zahlen

Transformation eines DGL-Systems auf 1. Ordnung

1.) Auflösen nach höchster Ableitung
2.) Für alle Variablen (außer höchste Ableitung) neue Variablen einsetzen
3.) als System schreiben

Bsp.: $\ddot{x} + \frac{c}{m}x = 0$

1) $\ddot{x} = -\frac{c}{m}x$
2.) $x_1 = x$ $x_2 = \dot{x}$
3.) $\frac{dx_1}{dt} = x_2$ $\frac{dx_2}{dt} = -\frac{c}{m}x_1$

$$\dot{x} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\frac{c}{m}x_1 \end{pmatrix}$$

- Variablen nur einführen, wenn Ordnung ≠ 1

Autonomisierung nichtautonomer DGL-Systeme

autonom: DGL-System hängt nicht explizit von t ab.

1.) Für t wird neue Variable eingeführt
2.) Ableitung der neuen Variable ist 1, da $\frac{dt}{dt} = 1$
3.) Ableitung wird in neue Zeile des Systems geschrieben

Bsp. $\dot{x} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} t^2 \cdot x_2 \\ \sqrt{t} \cdot x_1 \end{pmatrix}$

1.) $x_3 = t$
2.) $\dot{x}_3 = 1$
3.) $\dot{x} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} x_3^2 \cdot x_2 \\ \sqrt{x_3} \cdot x_1 \\ 1 \end{pmatrix}$

Ohne Beschränkung der Allgemeinheit:
DGL-System 1. Ordnung und autonom

Lösbarkeit von DGL-Systemen

Satz v. Picard-Lindelöf:
Ein autonomes DGL-System 1. Ordnung hat eine eindeutige Lösung, falls Lipschitz-Bedingung $\|f(x_1) - f(x_2)\| \leq L \|x_1 - x_2\|$ erfüllt ist

$L > 0$ (betragsgrößte Steigung v. f auf $[x_1, x_2]$)
Norm beliebig (z.B. Euklid)

Normalisierte Gleitpunktzahlen

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

B = Basis, E: Exponent, d_i = Ziffern

$z \neq 0 \Rightarrow d_1 \neq 0$, t = Länge der Mantisse

Zahl v. Komma gehört zur Mantisse

IEEE 754: 1 Bit Vorzeichen
8 Bit Exponent, 23 Bit Mantisse
Exponent = 127 + Exponent
Mantisse =
> 1: durch 2 teilen: Reste sind Zahl rückwärts
< 1: mit 2 mult.: ≥ 1 Diff. m. 1 nehmen und 1, sonst 0

DGL-Systeme

$$\begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \end{pmatrix} = \begin{pmatrix} -m & a \\ b & -n \end{pmatrix} \cdot \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} + \begin{pmatrix} c \\ d \end{pmatrix}$$

=> Wettrüsten

n, m > 0 -> Abrüstraten c, d. konst.
c, d > 0 -> Aufrüstraten Auf-/Abrüstraten

Kondition

- Abhängigkeit der Lösung eines Problems von der Störung der Eingangsdaten
- Konditionszahl: Maß für d. Abhängigkeit (hängen nur von f ab)
- -> Beträge der Verstärkungsfaktoren
  ↳ groß: schlecht konditioniert
  ↳ klein: gut konditioniert
- Konditionszahlen hängen nur von f ab
- Konditionszahlen = $\left\| \left(\frac{x_j}{f_i(x)} \cdot \frac{\partial f_i(x)}{\partial x_j} \right) \right\|$

1.) relative Fehler für jede Variable aufstellen.
2.) Verstärkungsfaktoren ausrechnen und untersuchen, wann sie groß/klein sind

Bsp.: Seien p, q die Variablen der Funktionen y_1, y_2

$$\varepsilon_1 = \frac{p_0}{y_1}\frac{\partial f_1}{\partial p}\varepsilon_p + \frac{q}{y_1}\frac{\partial f_1}{\partial q}\cdot\varepsilon_q$$

$$\varepsilon_2 = \underbrace{\frac{p}{y_2}\frac{\partial f_2}{\partial p}}\varepsilon_p + \underbrace{\frac{q}{y_2}\frac{\partial f_2}{\partial q}}\cdot\varepsilon_q$$

Verstärkungsfaktoren

- y_i ist gleich f_i

Numerische Stabilität

Sei $y = f(x)$ gut konditioniert. Berechnungsverfahren für f ist num. stabil, wenn relative Eingabefehler nicht verstärkt werden. num. instabil, falls große relative Fehler im Ergebnis erzeugt werden. num. stabil z.B. wenn Auslöschung (keine Subtraktion) vermieden wird. oder gr. Zwischenergebnisse. Ergebnis Lösungen bei Instabilität: höhere Genauigkeit

Fixpunktiteration

z.B.: $\phi(x) = x$
$\phi(x) = x + f(x)$ einfachster Fall

$x_{k+1} = \phi(x_k)$

- Konvergenz $\frac{\partial f}{\partial x}(x_s)$ -> Eigenwerte liegen im Einheitskreis. x_s ist ein Fixpunkt, der nah an x_0, der genauen Lösung liegt.
- Konvergenz nicht immer -> kann durch zusätzliche Matrix B verbessert werden
  $\phi(x) = x + B \cdot f(x)$
- optimal: $B = -\left[\frac{\partial f}{\partial x}(x_s)\right]^{-1}$

Konstruktion einer konvergenten Fixpunktiteration ist (theoretisch) fast immer möglich

- Ordnung == 1
- Konvergenzhilfe mittels Relaxationsmatrix

Lösungen von DGLs

homogenes lineares DGL-System n. Ordnung:
$x^{(n)}(t) + a_{n-1}x^{(n-1)} + \dots + a_0 x(t) = 0$
charakteristisches Polynom:
$\lambda^n + a_{n-1}\lambda^{n-1} + a_1\lambda + a_0 = 0$
Fundamentalsystem: $\{e^{\lambda_j t}, te^{\lambda_j t}, \dots\}$
Lösung: $x(t) = \sum_{i=1}^{n} c_i \cdot e^{\lambda_i t}$

Newton-Verfahren

- spezielles Fixpunktverfahren
- $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$
- für Matrizen: $x_{k+1} = x_k - \left[\frac{\partial f}{\partial x}(x_k)\right]^{-1} \cdot f(x_k)$
- Ordnung == 2
- Startvektor (x_0) muss gut gewählt sein
- Schwierigkeiten: hoch nichtlinear und viele Nullstellen
- Quadratische Konvergenz nur für $\alpha = 1$, verdoppeln sich richtige Stellen
  ↳ In jedem Schritt
- $x_{k+1} = x_k - \alpha \frac{f(x_k)}{f'(x_k)}$
- spezielles Fixpunktverfahren ($\phi(x) = -\frac{f(x)}{f'(x)}$)
- hoher Rechenaufwand, wenn f nicht linear
- Genauer als Fixpunktverf.
- nur für einfache Nullstellen
- f muss 2mal stetig diffbar sein

Quasi-Newton-Verfahren

Jacobi-Matrix wird nur angenähert, um Rechenaufwand zu sparen
1.) Berechnung v. $f(x^{(k)})$
2.) Berechnung einer angenäherten Jacobi-Matrix
3) Berechnung der Korrektur $\Delta x^{(k)}$ durch Lösung des linearen Gleichungssystems

Man könnte die Jacobi-Matrix auch durch eine konstante, ähnliche Matrix ersetzen und das herkömmliche Verfahren anwenden.

Eulerverfahren

explizit: $x_{k+1} = x_k + h \cdot f(t_k, x_k)$
$t_k = t_0 + k \cdot h$
-> für steife DGL
implizit: $x_{k+1} = x_k + h \cdot f(t_{k+1}, x_{k+1})$
-> stabiler
$t_k = t_0 + k \cdot h$
↳ muss nach x_{k+1} zuerst umgeformt werden
- Ordnung == 1

Herleitung
- Ausgangsgleichung:

$$x(t_{k+1}) = x_k + \underbrace{\int_{t_k}^{t_{k+1}} f(x(\tau))d\tau}$$

entspricht Fläche, die hier mit Rechtecken angenähert wird

- Approximation: $x_k \approx x(t_k)$
- Rechenaufwand d. impl. Eulerverfahrens viel höher
- Approx.fehler $O(h^1)$ -> Fehler bzgl. Schrittweite

Sonstiges

Je höher die Konvergenzordnung, desto schneller nähert sich die Folge dem Grenzwert

	FPI	New.
lokale Konv.	−	++
globale Konv.	0	−
Rechenaufwand	++	−
Implement.aufwand	++	−

- Gesamtaufwand = Anzahl Iterationen · Aufwand pro Iteration

lokale Konvergenz: für alle Startpunkte in einer Umgebung konvergiert das Iterationsverfahren

globale Konvergenz: konvergiert für alle Startwerte

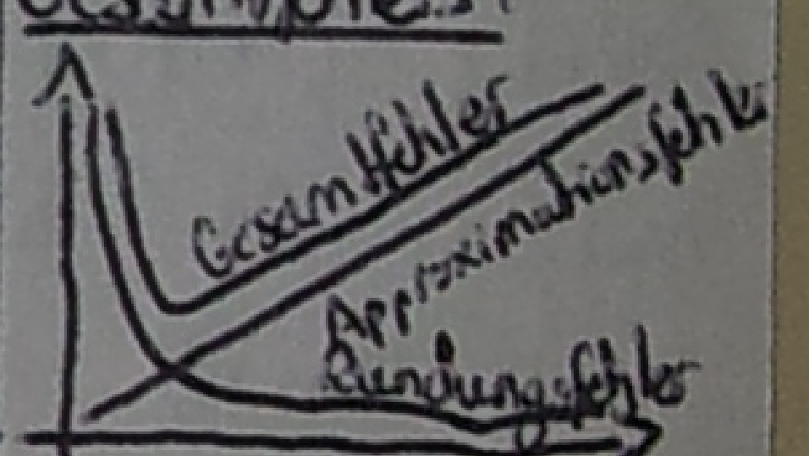
Approximation der j-ten Spalte der Jacobi-Matrix:

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

$\delta_j = \varepsilon(1 + |x_j|)$

$\varepsilon = \sqrt{\varepsilon_{mach}}$

Typischer Verlauf d. Gesamtfehlers.



Dünnbesetztheit der Jacobi-Matrix

Nur von 0 verschiedene Einträge werden approximiert

allgemeines Einschrittverfahren:

$x_{k+1} \approx x_k + h \cdot \Phi_k$
↳ bei Euler: $\Phi_k = f_k$

Numerische Integration

f muss min. so oft differenzierbar sein wie die Ordnung des Verfahrens

Blockorientierte Simulation in Simulink

- in System 1. Ordnung umformen

$\boxed{a} \to \boxed{\sin} \to \sin(a)$

Diskretisierungsfehler

lokaler: Fehler der innerhalb eines Verfahrensschrittes von x_{i-1} nach x_i auftritt

- an Stelle t_{k+1} ist der lokale Diskretisierungsfehler:

$$d_{k+1} = x(t_{k+1}) - x(t_k) - h\cdot\Phi(t_k, x(t_k), h, f)$$

- globaler an Stelle t_k:

$$g_k = x(t_k) - x_k$$

- Bestimmung der Fehlerordnung:

1) Taylorentwicklung der Terme mit k+1 (x_{k+1}, t_{k+1})
2) in d_{k+1} einsetzen
3.) höchste Potenz − 1 == Ordnung

Beispiel: Trapezregel

$d_{k+1} = x(t_{k+1}) - x(t_k) - h\cdot(\frac{1}{2}f(x_{k+1}) + \frac{1}{2}f(x_k))$

Taylorentwicklung:

$x(t_{k+1}) = x(t) + x'(t)h + \frac{1}{2}x''(t)h^2 + \frac{1}{6}x'''(t)h^3$

$\underbrace{f(x_{k+1}) = x'(t_{k+1})}_{!\ \dot{x} = f(x)\ !} = x'(t) + x''(t)h + \frac{1}{2}x'''(t)h^3$

Einsetzen ergibt: $-\frac{1}{12}x'''(t)h^3 + O(h^4)$

=> Ordnung == 2

Heun-Verfahren

- gehört zur Klasse der Runge-Kutta-Verfahren
- Mittelung erfolgt über Trapez
- führt eine Korrektur des expl. Eulerverfahrens durch
- Vorschrift:

$$x^P_{k+1} = x_k + h\cdot f(t_k, x_k)$$

$$x_{k+1} = x_k + \frac{1}{2}\cdot h(f(t_k, x_k) + f(t_{k+1}, x^P_{k+1}))$$

- globaler Fehler geht mit h^2 gegen 0 → Konvergenzordnung == 2

Runge-Kutta-Verfahren

- allgemeiner Ansatz

$s_1 = f(x_k)$
$s_2 = f(x_k + h\cdot\alpha_1\cdot s_1)$
$s_3 = f(x_k + h\cdot\beta_1\cdot s_1 + h\cdot\alpha_2\cdot s_2)$
$s_4 = f(x_k + h\cdot\gamma_1\cdot s_1 + h\beta_2 s_2 + h\alpha_3\cdot s_3)$
$x_{k+1} = x_k + h\cdot(\delta_1\cdot s_1 + \delta_2\cdot s_2 + \delta_3\cdot s_3 + \delta_4 s_4)$

- Problem: Bestimmung der Koeffizienten $\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \gamma_1, \delta_1, \delta_2, \delta_3, \delta_4$, sodass $\lim_{h\to 0}\Phi(t_k, x_k, x_{k+1}, h, f) = f(x_k)$ gilt und $\boxed{\delta_1 + \delta_2 + \delta_3 + \delta_4 = 1}$
- klassische Verfahren
  - 4. Ordnung:

$s_1 = f(x_k)$ $s_2 = f(x_k + \frac{h}{2}\cdot s_1)$
$s_3 = f(x_k + \frac{h}{2}\cdot s_2)$ $s_4 = f(x_k + h\cdot s_3)$
$x_{k+1} = x_k + \frac{h}{6}\cdot(s_1 + 2s_2 + 2s_3 + s_4)$

  - 3. Ordnung:

$s_1 = f(x_k)$ $s_2 = f(t_k + \frac{1}{2}h, x_k + \frac{h}{2}s_1)$
$s_3 = f(t_k + h, x_k - hs_1 + 2hs_2)$
$x_{k+1} = x_k + h(\frac{1}{6}s_1 + \frac{4}{6}s_2 + \frac{1}{6}s_3)$

  - 2. Ordnung: Heun-Verfahren

- bei gegebener Schrittweite h_k:
  → Approximationsfehler höherer Ordnung sehr viel kleiner
- Je höher die Genauigkeit sein soll, desto höher die Ordnung

Schrittweitensteuerung

- bei konstanten h_k
  - ungenau, wenn sich gesuchte Lösung lokal stark ändert
  - ineffizient, wenn gesuchte Lösung sich lokal wenig ändert
- Abhilfe mit Schrittweitensteuerung: h_k so groß wie möglich und so klein wie nötig
- übliche Varianten: ①

1) Berechnung von x_{k+1} mit h_k und einmal mit $h_k/2$
2) lokalen Fehler schätzen
3) Anpassung von h_k → Fehlerschätzung liegt unter vorgegebener Schranke
4) lokaler Fehler unter vorgegebener Schranke: akzeptiere Schritt (der mit $\frac{h_k}{2}$). Sonst neuen Schritt mit angepasster h_k

- Aufwand: Bei s-Stufen: s + 2s Auswertungen

② zweite Variante:

1) x_{k+1} mit zwei Verfahren unterschiedlicher Ordnung berechnen mit derselben Schrittweite
2.) lokalen Fehler schätzen

- nur s+1 Auswertungen
- bei geringer Ordnung ist erstes Verfahren schneller

Numerische Integration unstetiger dynamischer Systeme

- Ursachen für Unstetigkeiten
  - Approximation von Teilmodellen von f mit stückweise stetigen Funktionen
  - Hysterese: Teil von f hängt von Vorgeschichte von x ab
- Lösung: Beheben von Unstetigkeiten und Schaltfunktionen benutzen
- Schaltfunktionen hat an Schaltpunkten Nullstellen
  ↳ einfachste Schaltfunktionen:
  $q_1 = x - d_1$ $q_2 = x - d_2$
  d_1, d_2 == Schaltpunkt
- Bei numerischer Integration müssen Vorzeichen der Schaltfunktion beachtet werden → bei Wechsel: Schaltzeitpunkt (Nullstelle) bestimmen
- h_k muss klein genug sein, sodass kein doppelter Vorzeichenwechsel derselben Schaltfunktion stattfinden kann

⇒ numerische Int. von einem Schaltzeitpunkt zum anderen

- Kriterien für Integratoren
  - Rechenaufwand, Genauigkeit, Eignung für steife Systeme, Implementierungsaufwand

Teilschritte einer Simulationsstudie

Wasserfallmodell

Problemspezifikation
Modellierung ←
Implementierung ←
Validierung ←
Anwendung ←

Zustandsvariablen → Ausgangspunkt zur Modellbildung

- zeitabhängige Größen
- legen aktuelle Systemkonfiguration fest
- legen zukünftigen Verlauf fest
- sind nicht redundant

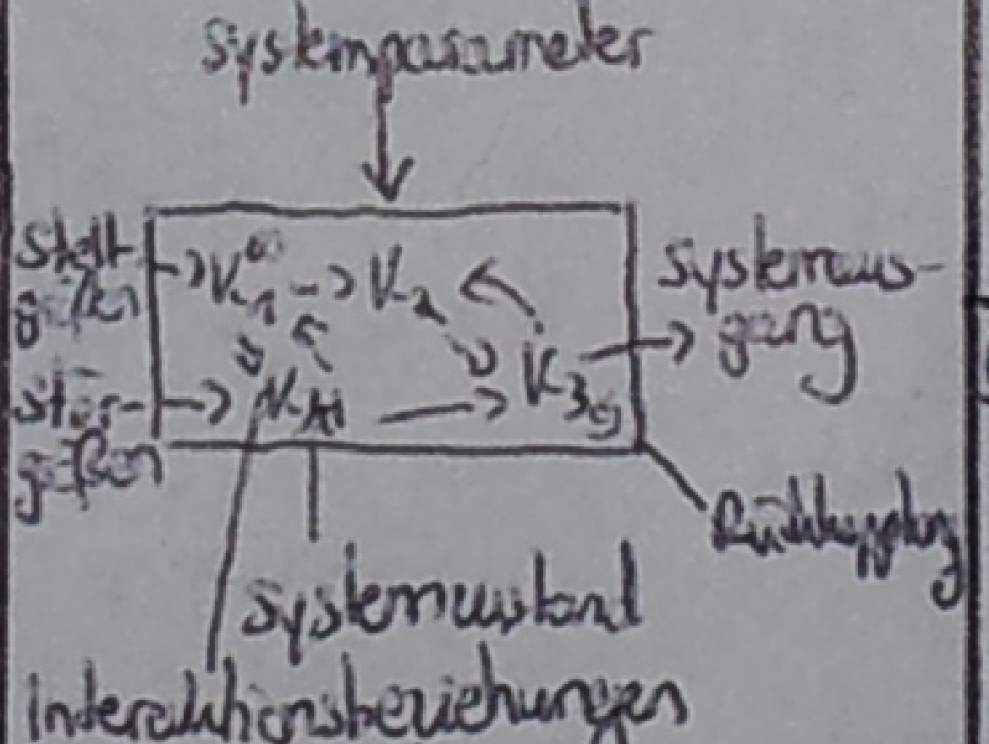
Interpretation & Validierung

- Verifikation: (meist mathematischer) Nachweis der Korrektheit, dass Programm vorgegebener Spezifikation entspricht
- in der Regel unmöglich, vollständige Korrektheit zu beweisen
- Validierung: Begründung der These, dass Modell innerhalb des Anwendungsbereiches ausreichende Genauigkeit hat
  - Elemente einer Validierung:
    - Implementierung von
    - Modell +
    - Berechnungsverfahren
- Tests auf Plausibilität und Konsistenz
  - Reproduktion von beobachtetem bzw. natürlichem Systemverhalten
  - Vorhersage von Verhalten
  - Verhaltensanomalien (große Unterschiede zw. Modell und Realität => Fehler)
  - Systemverhalten in Extremsituationen
  - Parametervariationen / Parametersensitivität
    ↳ Falls kein Parametersatz gefunden wurde, für den die Abweichungen klein sind

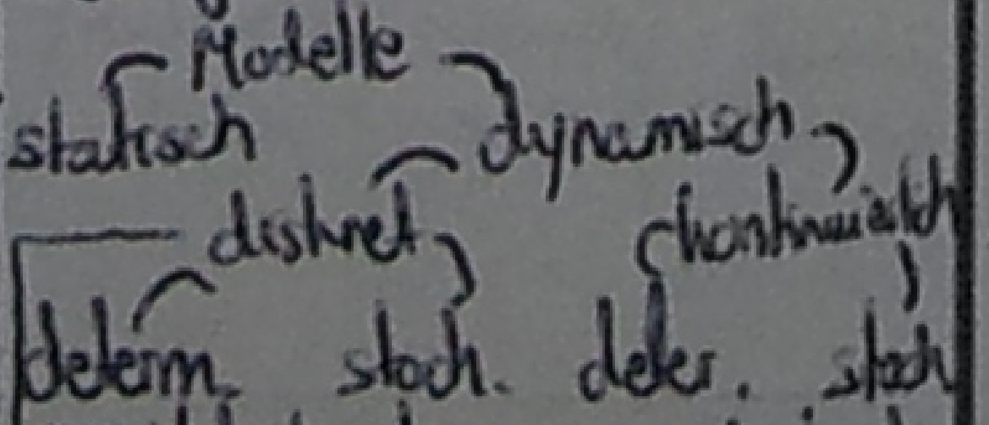
1) verwendete Optimierungsverfahren ungeeignet
2) experimentelle Daten nicht ausreichend
3.) Messfehler zu groß
4.) Modell nicht detailliert genug

Begriffe

System: Ausschnitt der realen Welt



- Lösungsansätze für mathematische Modelle
  - analytisch
  - heuristisch
  - direkt-numerisch
  - approximativ-numerisch
- Modell: Ersatzsystem, gebildet unter Annahmen und Idealisierung, vereinfachtes Abbild der Realität
  ↳ diskret: diskrete Beschreibung
  ↳ kontinuierlich: kontinuierliche Beschreibung
  ↳ deterministisch
  ↳ stochastisch
- Arten der Zustandsübergänge



↳ zeitdiskret, ereignisdiskret

DEVS (Discrete Event Simulation)

- Modellierung diskreter Event-Systeme
- Ereignisalgorithmus:

```
1.) Initialisierung:
    L={(t_1,E_1)}; t:=0;
    InitializeSystem();
2.) Ereignisschleife:
    while t < t_max
    begin
        t:=t_1; E:=E_1;          // nächstes Ereignis
        L:=L\(t_1,E_1);          // Ereignis aus Liste entfernen
        EventRoutine(E);         // Ereignisbehandlung
        sort(L);                 // Liste nach t_i aufsteigend sortieren
```

```
- InitializeSystem()
  begin
      N:=0;
      S:=idle;
  end;

- EventRoutine(E)
  begin
      switch(E)
          A: ArrivalRoutine();
          D: DepartureRoutine();
      end;
  end;

- ArrivalRoutine()
  begin
      L:=L ∪ (t+exp(λ), A);
      if(S==busy)
          N:=N+1;
      else
      begin
          S:=busy;
          L:=L ∪ (t+norm(μ,σ), D);
      end;
  end;

- DepartureRoutine()
  begin
      if N==0
          S:=idle;
      else
      begin
          N:=N-1;
          L:=L ∪ (t+norm(μ,σ), D);
      end;
  end;
```

- exp (Exponentialverteilung) ist gedächtnislos

Grundbegriffe der Statistik

Beschreibung eindimensionaler x' durch Verteilungsfunktion $F(x)$:

- $F(x) = P(x' < x)$
- $F(b) - F(a) = P(a \le x' \le b)$
- $F(-\infty) = 0$, $F(\infty) = 1$
- $F(x_k) = p_1 + p_2 + \dots + p_k$ x = diskret verteilt
- Wahrscheinlichkeitsdichte: x = stetig verteilt
  $f(x)\cdot dx = P(x < x' \le x + dx)$
  $F(x) = \int_{-\infty}^{x} f(s)\cdot ds$

Alle Teilintervalle gleicher Länge haben die gleiche Wahrscheinlichkeit.

- Erwartungswert diskreter reeller Zufallsvariabel:

$$E(x) = \sum_{i\in I} x_i\, P(X = x_i)$$

↳ x_i ist ein Ereignis und $P(x = x_i)$ die dazugehörige Wahrscheinlichkeit

- Varianz (stets größer gleich 0) (Streuung der Werte relativ zum Erwartungswert → hängt nicht vom Zufall ab)

$$Var(x) = \sum (x - E(x))^2 \cdot P(X = x)$$

- Standardabweichung: Wurzel der Varianz
- Binomialverteilung: $E = n\cdot p$, $\sigma = \sqrt{np(1-p)}$

Petrinetze

- zur Modellierung ereignisdiskreter Systeme mit nebenläufigen Eintritten von Ereignissen
- Zustand: $M = \{m_1, m_2, \dots, m_i\}$
  m_i := Anzahl der Marken auf p_i
- 2 Typen von Petrinetzen:
  - zeitdiskrete Modelle: Was passiert in welcher Reihenfolge
  - zeitkontinuierliche Modelle: Wann tritt welches Ereignis auf?
- Transition kann nur schalten, wenn alle Eingänge belegt sind.
- W^+: 1, für Verbindung die aus Transition in Zustand geht.
- W^-: 1 für Verbindung, die aus Zustand raus und in Transition rein geht.
- $W = W^+ - W^-$
  ↳ 1: aus Transition, in Zustand
  −1: aus Zustand, in Transition
- Berechnung des Zustands nach mehrmaligem Feuern:

$$m = m_0 + W\cdot u = m(r-1) - W^- u(r) + W^+ u(r)$$

↳ m_0 = Markierungsvektor am Anfang
↳ W = Inzidenzmatrix
↳ u = Aktivierungsfunktion (Vektor, der dort 1 hat, wo gefeuert wird)

- Kapazitäten: Wie viele Markierungen auf einem Zustand? → k_i
- Komplexität des Erreichbarkeitsproblems ist exponentiell hart im Zeit-/Speicheraufwand
- beschränkt: Auf p_i sind nie mehr als k_i Markierungen
- sicher: $k_i = 1$
- tote Transition: bei keiner Folgemarkierung mehr aktivierbar
- nicht lebendiges Petrinetz = min. eine tote Transition

Zufallszahlen

- lineare Kongruenzgeneratoren

x_0 := Startwert (seed)

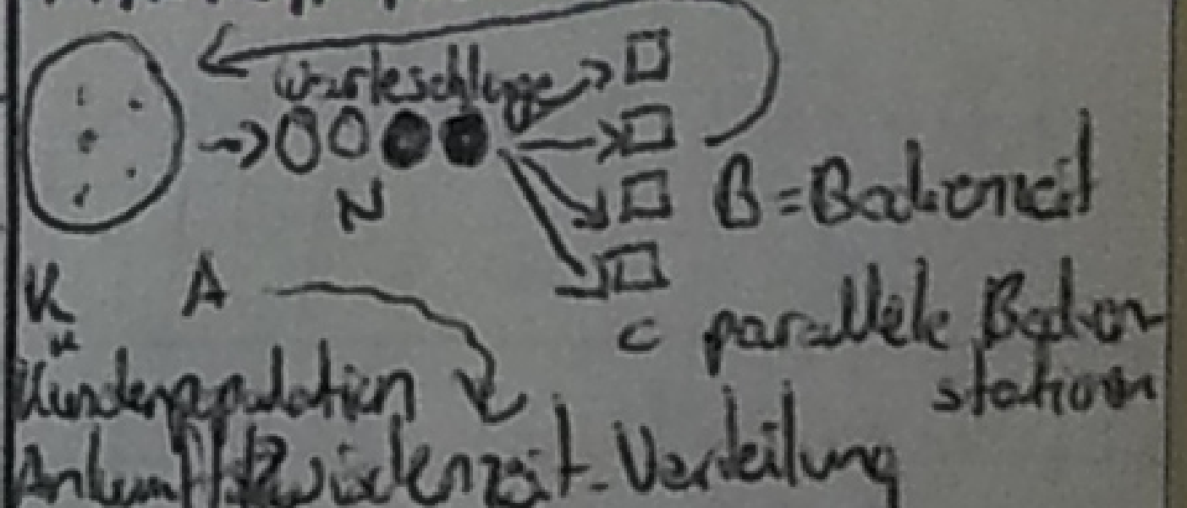
$x_{k+1} = (a\cdot x_k + c) \bmod m$

a: multiplier c: increment
m: modulus (Periode)

- kontinuierliche Zufallszahlen mit anderen Verteilungen über Transformation

Allgemeines Warteschlangenmodell

A/B/c/N/K



Typische Verteilungen für A, B:

M: (negativ) exponentiell (markovsch)
D: deterministisch
G: beliebig

Bezeichnungen:

λ = E(A) mittlere Ankunftsrate
μ = E(B) mittlere Bedienrate
σ^2 = Var(B) Streuung der Bedienrate
ρ: Langzeitauslastung der Server
ω: Langzeitaufenthaltszeit im System

Gesetz von Little (z.B. für Validierung einer Simulation)

$$L = \lambda \cdot \omega$$

L: mittlere Anzahl der Aufträge im System
ω → mittlere Aufenthaltszeit der Aufträge

- Bedingung für Stationarität $\lambda < \mu$
- andere Größen: $\rho = \frac{\lambda}{\mu}$ $L = \frac{\rho}{1-\rho}$
- Warteschlangen werden länger, wenn bei fixem Erwartungswert die Streuung der Bedienzeit wächst
  $\omega = \frac{1}{\mu - \lambda}$
  n Server mit 1 Warteschlange deutlich besser als [illegible] n Server