Systemsimulation

Jörg J. Buchholz

1999

Dieses Skript finden Sie auch im Internet unter

http://buchholz.hs-bremen.de/sysi/skript

1 Inhalt

2 Simulatoren

Simulation ist die Nachbildung in der Wirklichkeit vorkommender oder fiktiver (zukünftiger, geplanter) Prozesse. Ein Simulator ist die hardwaremäßige Realisierung dieser Nachbildung. In den meisten Simulatoren befindet sich der Mensch mit im Simulationskreis (pilot-in-the-loop). Diese Simulatoren müssen daher in Echtzeit arbeiten und möglichst viele der für den Menschen wichtigen Eindrücke darstellen (Sicht, Bewegung, Geräusche, Sprache, ...).

2.1 Klassen von Simulatoren

- Entwicklungs- und Systemsimulator (hardware-in-the-loop, iron bird)
- Trainings-, Schulungs- und Missionssimulator
- Experimental- und Fliegender Simulator
- Freizeit- und Demonstrationssimulator

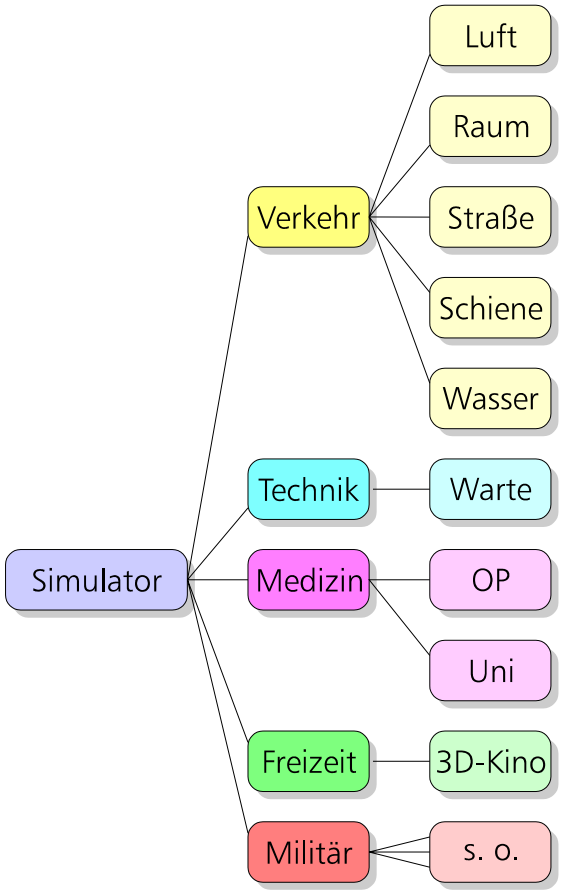

Bild 1 Einige Anwendungsgebiete von Simulatoren

3 Modellbildung

Modellbildung ist die Abbildung der Realität in Form mathematischer Beschreibungen. Dabei findet immer eine Vernachlässigung “unwichtiger” Systemteile statt.

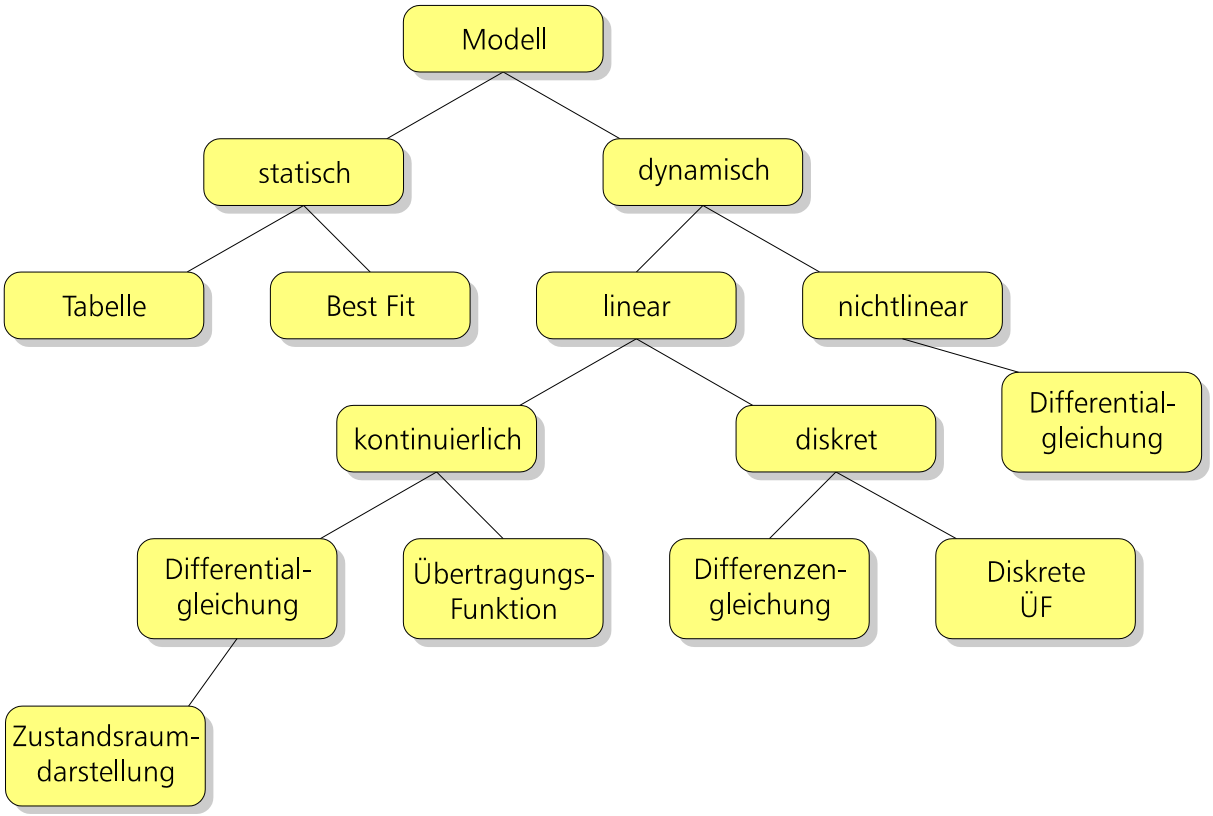

Bild 2 Unterschiedliche Modellbeschreibungen

3.1 Fragen bei der Modellbildung

- Wie erhalte ich das Modell? (physikalisches Verständnis, Messen, Identifikation, ...)
- Welche Systemteile kann und darf ich vernachlässigen?
- Was muß ich unbedingt mitmodellieren?
- Wie kann ich das Modell möglichst effektiv beschreiben, darstellen, abspeichern und abrufen? (Speicherplatz, Rechenzeit, Kosten, ...)

4 Tabelle, Kennlinie, Look Up Table

Tabellen sind immer dann sinnvoll, wenn nur sehr wenige, dafür aber recht zuverlässige Daten vorhanden sind, z. B. aus dem Windkanal. Es ist dann erforderlich, eine Kurve zu suchen, die genau durch die vorhandenen Datenpunkte hindurchgeht und dazwischen plausible Schätzungen liefert. Die Tabelle ist ein statisches System ohne eigene Dynamik; zu jedem Eingangswert gehört genau ein Ausgangswert.

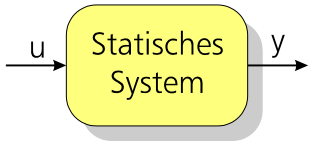

Bild 3 Ausgangsvektor y ist nur von Eingangsvektor u abhängig

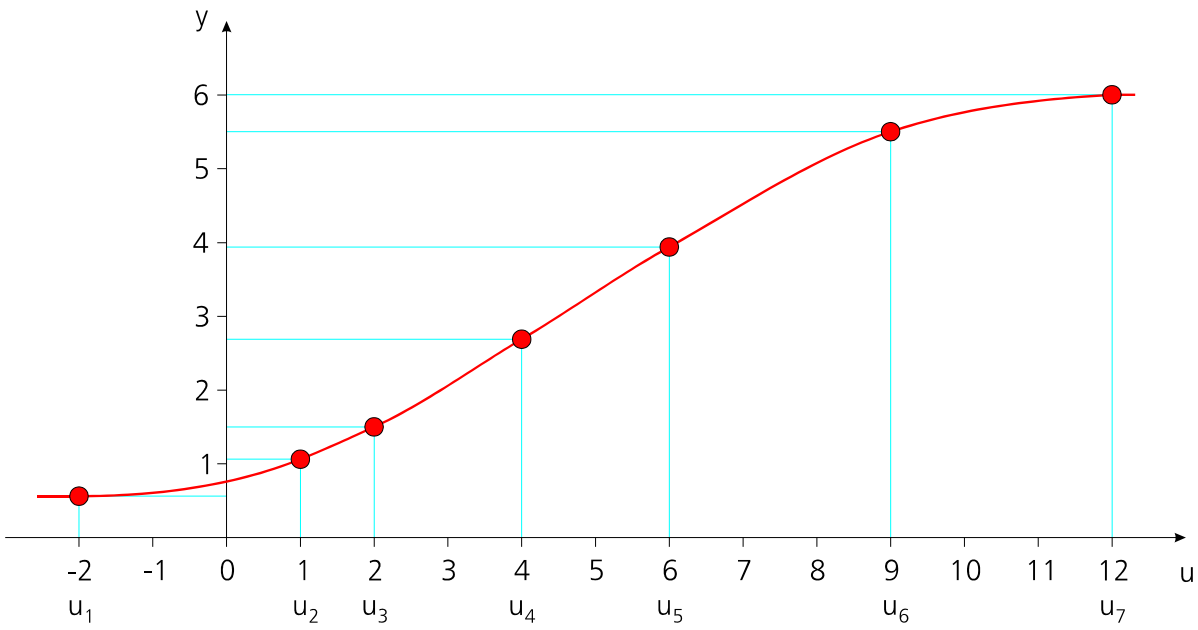

Bild 4 Eindimensionale Kennlinie mit nicht äquidistanten Stützstellen

4.1 Tabelle aufbauen

Der Stützstellenvektor $u_S = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \\ \vdots \\ 12 \end{bmatrix}$ und der Datenvektor $y_S = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} 0.56 \\ \vdots \\ \vdots \\ \vdots \end{bmatrix}$ müssen die gleiche Länge haben. Der Stützstellenvektor braucht dabei nicht äquidistant zu sein. Wenn er aber äquidistant ist, reichen drei der vier Größen u_1, Δu, u_N und N.

Der Stützstellenvektor sollte monoton sein: $u \neq \begin{bmatrix} 1 \\ 5 \\ 3 \\ 8 \end{bmatrix} \Rightarrow u = \begin{bmatrix} 1 \\ 3 \\ 5 \\ 8 \end{bmatrix}$.

Beim Sortieren des Stützstellenvektors u den Datenvektor y mitsortieren lassen.

4.2 Tabelle auswerten

Abhängig von der zu interpolierenden Kurve kann jedes Interpolationsverfahren Vorteile gegenüber den anderen haben. Ein Sinus läßt sich besser durch die "weichen" Splines annähern als durch Geraden. Eine Rechteckschwingung wiederum kann nur durch einen linearen Polygonzug exakt beschrieben werden. Polynome höherer Ordnung neigen häufig zu wilden Schwingungen zwischen den Tabellenpunkten. Gebrochen rationale Funktionen haben aufgrund ihres Nenners, der ja auch Null werden kann, immer dann Vorteile, wenn Singularitäten nachgebildet werden sollen.

Extrapolation klappt fast nie! Schon wenige Stützstellenabstände außerhalb des Interpolationsbereichs liegen die meisten Extrapolationsfunktionen weitab vom wahren Wert.

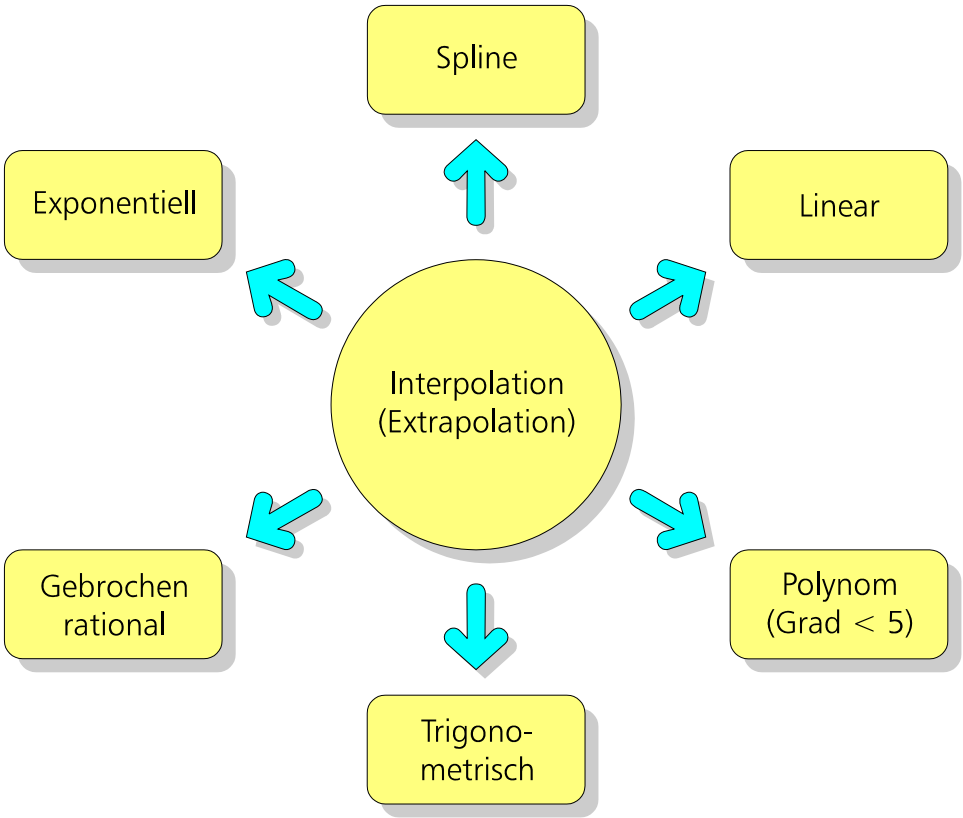

Bild 5 Unterschiedliche Interpolationsverfahren

4.2.1 Lineare Interpolation

Bei der linearen Interpolation werden die Tabellenpunkte durch Geraden (Polygonzug) miteinander verbunden. Diese einfachste Interpolationsart führt, wenn die Stützstellen nicht allzu weit auseinander stehen, in den meisten Anwendungen zu durchaus akzeptablen Ergebnissen.

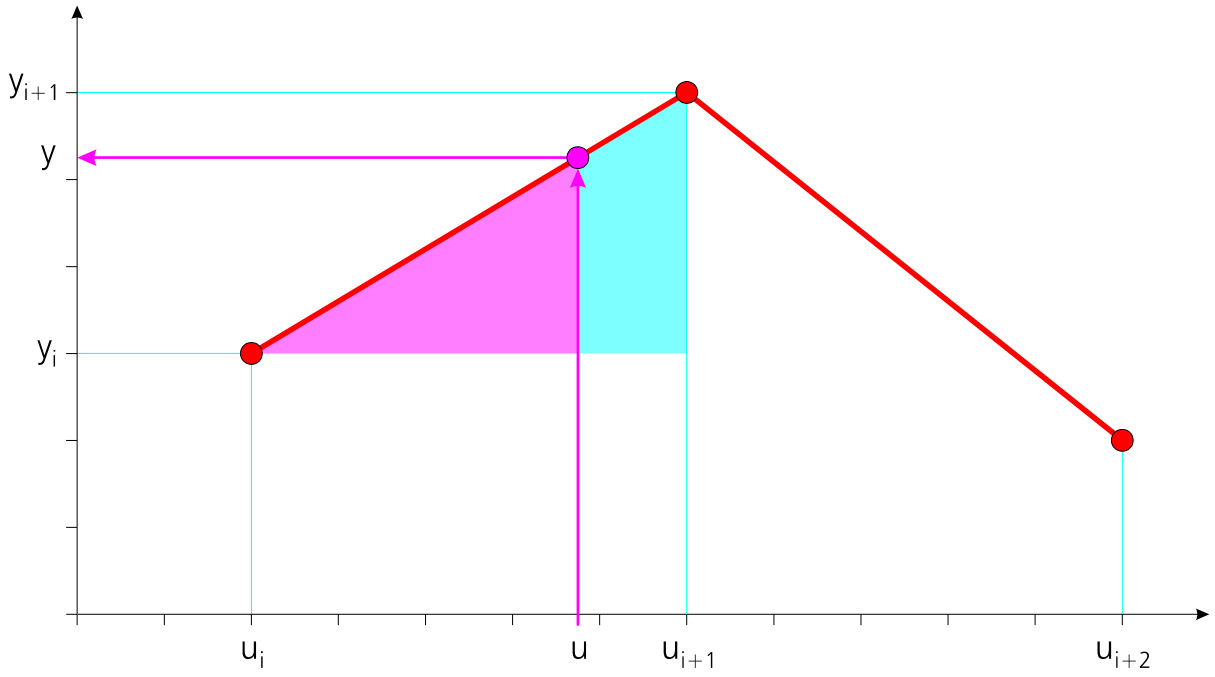

Bild 6 Lineare Interpolation

Ähnliche Dreiecke: $\frac{y - y_i}{u - u_i} = \frac{y_{i+1} - y_i}{u_{i+1} - u_i} \quad \Rightarrow \quad y = \frac{y_{i+1} - y_i}{u_{i+1} - u_i}(u - u_i) + y_i$

4.2.2 Kubische Splines

Kubische Splines sind eine besondere Art von Polynomen dritten Grades, deren Koeffizienten so gewählt (berechnet) werden, daß an jedem Tabellenpunkt nicht nur die Kurve selbst, sondern auch die erste und sogar die zweite Ableitung stetig sind. Auf diese Weise entstehen sehr “weiche” und “runde” Interpolationskurven, die bei ebenfalls “ruhigen” Originalkurven oft erstaunlich gute Approximationen darstellen. Die Herleitung und Formulierung der Koeffizientenbedingungen für Splines ist etwas lästig. Üblicherweise stellt aber jede gute numerische Bibliothek wenigstens einen numerisch ausgereizten Spline-Interpolationsalgorithmus zur Verfügung.

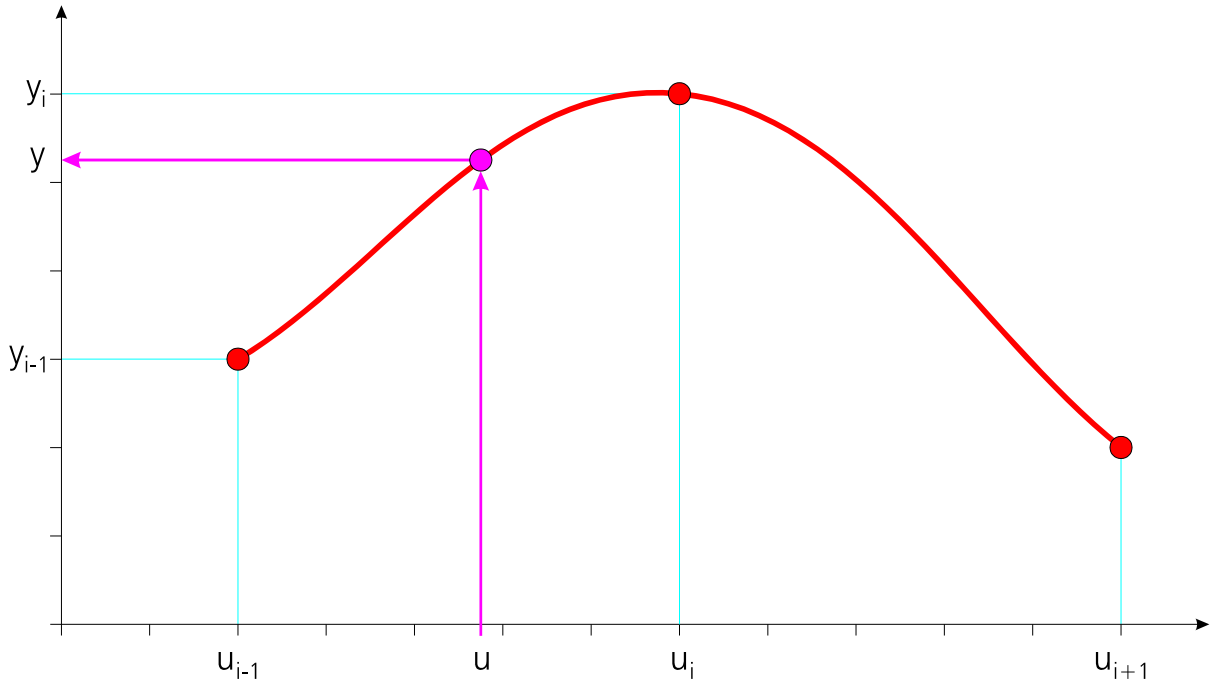

Bild 7 Kubische Splines

Kurve (nullte Ableitung) stetig: $y\big|_{u_{i-0}} = y\big|_{u_{i+0}}$

Steigung (erste Ableitung) stetig: $\frac{dy}{du}\Big|_{u_{i-0}} = \frac{dy}{du}\Big|_{u_{i+0}}$

Krümmung (zweite Ableitung) stetig: $\frac{d^2y}{du^2}\Big|_{u_{i-0}} = \frac{d^2y}{du^2}\Big|_{u_{i+0}}$

4.3 Zweidimensionale Tabelle

Eine zweidimensionale Tabelle besteht aus zwei möglicherweise verschieden langen Stützstellenvektoren $u_S = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 7 \end{bmatrix}$ und $v_S = \begin{bmatrix} -3 \\ 1 \\ 8 \end{bmatrix}$, die als unabhängige Variablen die zumeist rechtwinklige Grundfläche aufspannen, und einer Datenmatrix $Y = \begin{bmatrix} -102 & 2.93 & 5.81 \\ 4.27 & \cdots & \cdots \\ 10.3 & \cdots & \cdots \\ 9.42 & \cdots & -2.4 \end{bmatrix}$, die die Ordinatenwerte als Höhe über der Grundfläche darstellt.

Beispiel für Datenwerte an den Stützstellen: $y(u = 2, v = -3) = 4.27$
$y(u = 7, v = 8) = -2.4$

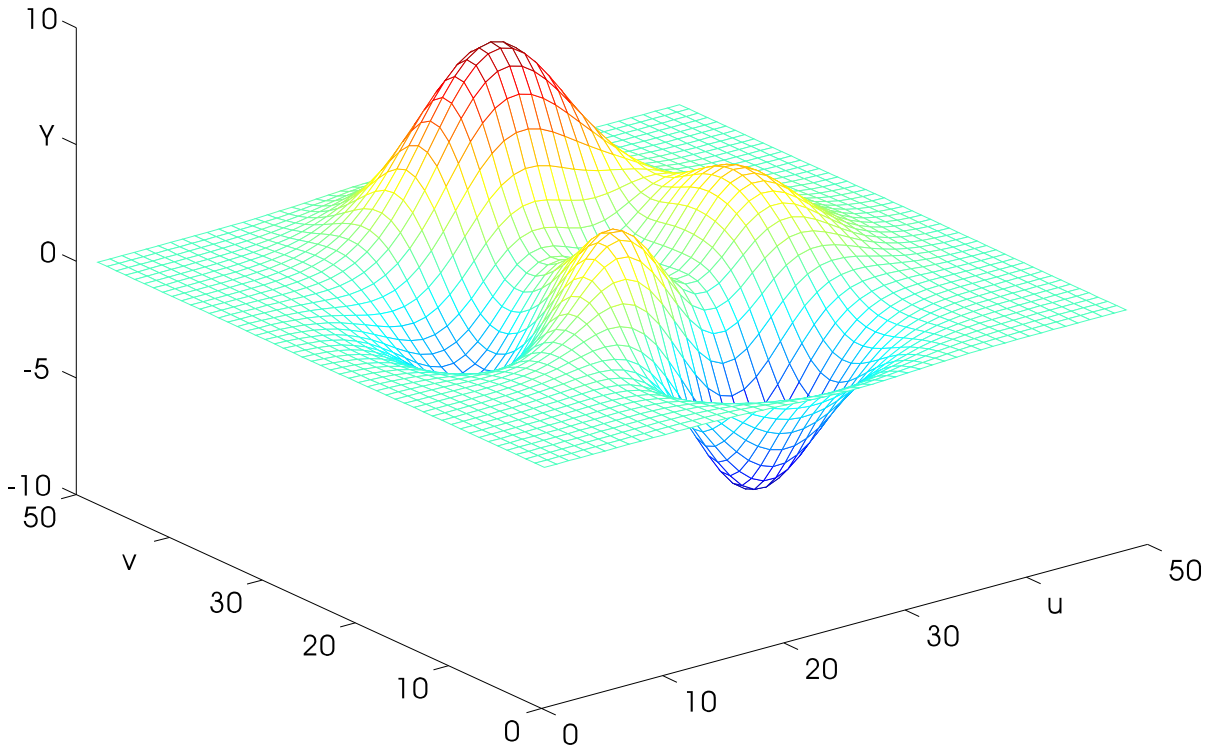

Bild 8 Graphische Darstellung einer zweidimensionalen Tabelle

Zur graphischen Veranschaulichung der Datenmatrix einer zweidimensionalen Tabelle muß verständlicherweise die dritte Dimension bemüht werden. Viele Numerikpakete stellen solche 3D-Plotroutinen standardmäßig zur Verfügung. Auch drei- oder vierdimensionale Tabellen können Sie unter Ausnutzung von Farbe oder als Film über der Zeit noch halbwegs übersichtlich darstellen.

Wie bei der eindimensionalen gibt es auch bei der mehrdimensionalen Interpolation zahlreiche Verfahren mit unterschiedlichen Eigenschaften. Die zweidimensionalen Pendants heißen hier: Bilineare Interpolation, Bikubische Interpolation, Bikubische Splines ...

4.3.1 Bilineare Interpolation

Über den Ecken einer rechteckigen Grundfläche sind vier Stützstellen (grün) definiert, die im Regelfall nicht alle in einer Ebene liegen. Gesucht ist nun der interpolierte Funktionswert (magenta), den ein Punkt hat, der über zwei Achsenwerte (blau) festgelegt ist.

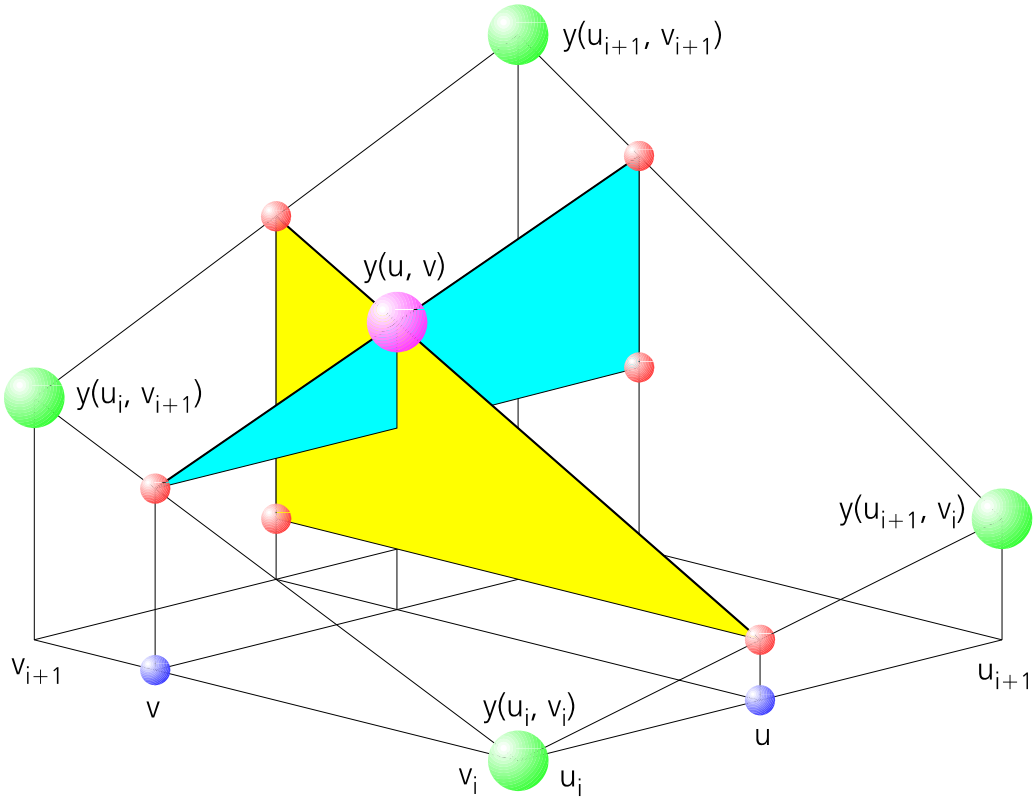

Bild 9 Bilineare Interpolation

Relative Abstände am Rande der Grundfläche: $r_u = \frac{u - u_i}{u_{i+1} - u_i}$ $r_v = \frac{v - v_i}{v_{i+1} - v_i}$

Ordinatenwert des zu interpolierenden Punktes:

$$y(u,v) = (1-r_u)(1-r_v)y(u_i,v_i) + r_u(1-r_v)y(u_{i+1},v_i) + (1-r_u)r_v y(u_i,v_{i+1}) + r_u r_v y(u_{i+1},v_{i+1})$$

4.3.2 Veranschaulichung

- ✓ Stellen Sie sich vor, die grünen Stützstellenpunkte sind durch Drähte mit ihren jeweiligen nächsten Nachbarn verbunden. Im Regelfall bilden diese Drähte dann kein Rechteck.
- ✓ Bilden Sie aus zwei weiteren Drähten ein rechtwinkliges Kreuz, in dessen Mitte der gesuchte lila Interpolationspunkt liegt.
- ✓ Legen Sie dieses Kreuz so auf das Stützstellendrahtgestell, daß die Kreuzdrähte parallel zu den Achsen der Grundfläche ausgerichtet sind.
- ✓ Durch achsenparalleles Verschieben des Kreuzes können Sie jetzt jeden Punkt über der Grundfläche erreichen.
- ✓ Interessanterweise berührt dabei das Kreuz bei achsenparalleler Ausrichtung an allen vier roten Punkten das Drahtgestell, obwohl doch eigentlich schon drei Punkte zur Lagedefinition des Kreuzes im Raum ausreichen würden. Warum eigentlich?

5 Best Fit, Least Squares, Regression, Approximation

Eine Regression ist immer dann sinnvoll, wenn Sie sehr viele, aber verrauschte Daten vorliegen haben und diese durch einen möglichst einfachen Zusammenhang (Gerade, Parabel, e-Funktion, ...) beschreiben wollen. Während also bei der Tabelle die Interpolationskurve genau durch die Tabellenpunkte hindurchgeht, fordern Sie hier nur, daß der Fehler, den Sie mit der glatten Kurve unweigerlich machen, im Mittel über alle Punkte möglichst klein ist.

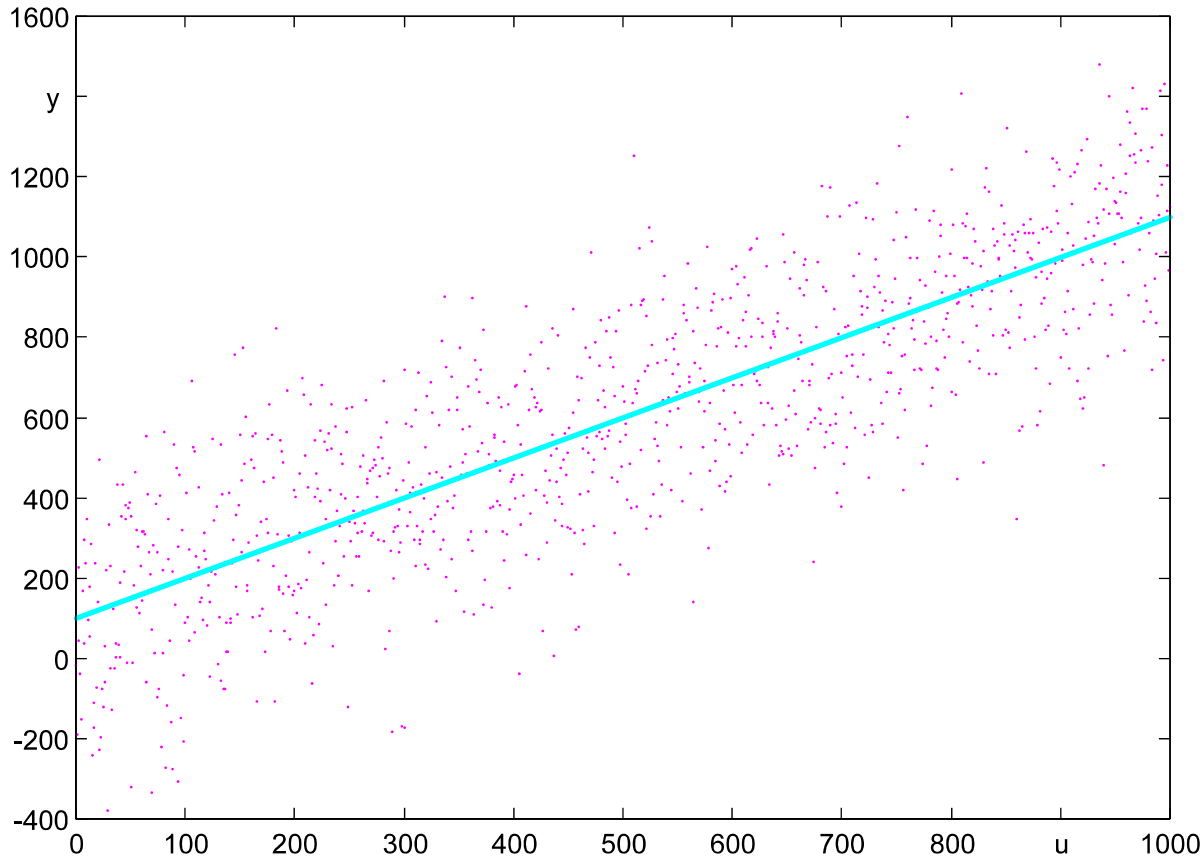

Bild 10 Regressionsgerade

Zwei lange Vektoren $u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix}$ und $y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$

beschreiben eine große Anzahl Punkte, durch die eine “optimale” Kurve (z. B. eine Regressionsgerade) gelegt werden soll.

Die Geradengleichung: $y = mu + n$

für jeden Punkt angesetzt, ergibt ein lineares, überbestimmtes Gleichungssystem:

$$\begin{aligned} y_1 &= mu_1 + n \\ y_2 &= mu_2 + n \\ &\vdots \\ y_N &= mu_N + n \end{aligned}$$

das sich auch in Matrizenschreibweise formulieren läßt: $\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} u_1 & 1 \\ u_2 & 1 \\ \vdots & \vdots \\ u_N & 1 \end{bmatrix} \begin{bmatrix} m \\ n \end{bmatrix}$

Mit den Definitionen: $A = \begin{bmatrix} u_1 & 1 \\ u_2 & 1 \\ \vdots & \vdots \\ u_N & 1 \end{bmatrix} \qquad x = \begin{bmatrix} m \\ n \end{bmatrix} \qquad b = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$

ergibt sich ein allgemeines, überbestimmtes, lineares Gleichungssystem: $Ax = b$

Die Koeffizientenmatrix A ist rechteckig schlank, ergo singulär und läßt sich daher nicht invertieren. Das Gleichungssystem kann also nicht nach $x = A^{-1}b$ aufgelöst werden. Es gibt im allgemeinen kein x, das das überbestimmte Gleichungssystem $Ax - b = 0$ exakt löst.

Gesucht ist daher die Näherungslösung der kleinsten Fehlerquadrate (Least Squares), also die Gerade, die im Mittel von allen Punkten den kleinsten Abstand hat:

$$(Ax - b)^2 \rightarrow \text{Minimum}$$

5.1 Singular Value Decomposition (SVD)

Die Singulärwertzerlegung ist das State-of-the-art-Lösungsverfahren für alle Arten von linearen Gleichungssystemen (bestimmt, überbestimmt und unterbestimmt). Die der SVD zugrundeliegenden numerischen Verfahren sind für den Ingenieur, der sich nicht eingehend mit numerischer Mathematik beschäftigen möchte, etwas schwer verdaulich. Nichtsdestotrotz kann und sollte die SVD in Form einer black-box-Routine quasi als Universalwerkzeug zur Behandlung linearer Gleichungssysteme unbedingt eingesetzt werden, da sie numerisch ausgesprochen stabil ist.

Das Grundprinzip ist die Zerlegung der Koeffizientenmatrix A in das Produkt dreier besonderer Matrizen, die sich einzeln alle leicht invertieren lassen.

$$\underbrace{\begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \vdots & \vdots & \vdots \\ \bullet & \bullet & \bullet \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \vdots & \vdots & \vdots \\ \bullet & \bullet & \bullet \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{bmatrix}}_{S} \underbrace{\begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{bmatrix}}_{V^T} \quad \Rightarrow \quad A = USV^T$$

S: Diagonalmatrix der Singulärwerte (singular values) s_i, leicht invertierbar: $S^{-1} = S(\frac{1}{s_i})$

U, V^T: Unitäre (orthogonale) Matrizen der Singulärvektoren (singular vectors),

auch leicht invertierbar: $U^{-1} = U^T \qquad V^{-1} = V^T$

Die (Pseudo-)Inversion der Gesamtmatrix A erfolgt dann über die Inversion des Matrizenproduktes und führt auf das Produkt der Einzelinversen in umgekehrter Reihenfolge.

Pseudoinverse: $A^+ = \left(USV^T\right)^{-1} = \left(V^T\right)^{-1} S^{-1} U^{-1} = VS(\frac{1}{s_i})U^T$

Die Lösung des Gleichungssystems ist jetzt trivial. Wenn die Matrix **A** regulär (nichtsingulär) ist, wenn es also genau eine Lösung des dann bestimmten Gleichungssystems gibt, liefert die SVD die exakte Inverse der Koeffizientenmatrix, die dann nur noch mit der rechten Seite multipliziert werden muß, um den unbekannten Lösungsvektor **x** zu erhalten.

Das Gleichungssystem ist jetzt im Least Squares-Sinne lösbar: $x = A^+ b = VS(\frac{1}{s_i})U^T b$

Wenn die Koeffizientenmatrix hingegen schlank ist, weil mehr Gleichungen als Unbekannte da sind oder wenn sie fett ist, weil es mehr Unbekannte als Gleichungen gibt oder wenn sie einfach nur singulär ist, weil sie interne lineare Abhängigkeiten beinhaltet, dann liefert die SVD eine Pseudoinverse zurück, die, wenn sie mit der rechten Seite multipliziert wird, das unbestimmte Gleichungssystem freundlicherweise automatisch mit dem minimalen quadratischen Fehler löst.

6 Lineares dynamisches System

In einem dynamischen System existiert kein statischer Zusammenhang zwischen Ein- und Ausgangsgröße mehr. Der zeitliche Verlauf der Ausgangsgröße wird vielmehr durch die dynamischen Eigenschaften und den Zustand der im System vorhandenen zumeist rückgekoppelten Speicher (Integratoren) bestimmt. Im Gegensatz zu statischen Systemen, die durch algebraische Gleichungen dargestellt werden können, werden dynamische Systeme durch Differentialgleichungen beschrieben, in denen die Zeit als unabhängige Variable auftritt.

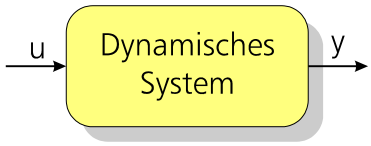

Bild 11 Dynamisches System beinhaltet Speicher (Integratoren)

Lineare Systeme sind in vieler Hinsicht angenehmer als nichtlineare Systeme. Sie lassen sich im allgemeinen wesentlich einfacher analysieren, regeln und simulieren. Zur Überprüfung, ob ein allgemeines nichtlineares System der Form $y = g(u)$ linear ist, werden zwei Linearitätsbedingungen herangezogen, die beide erfüllt sein müssen:

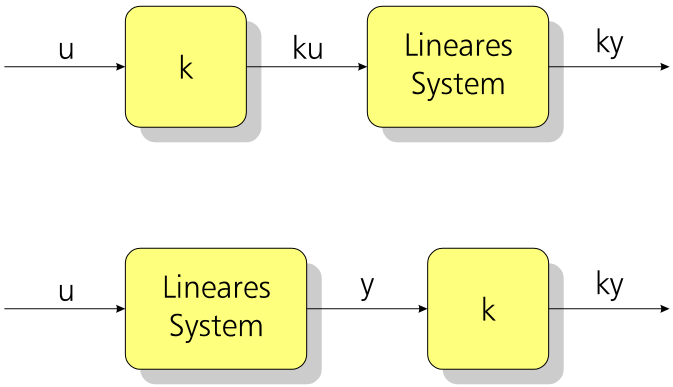

Bild 12 Verstärkungsprinzip: $g(k \cdot u) = k \cdot g(u) = k \cdot y$

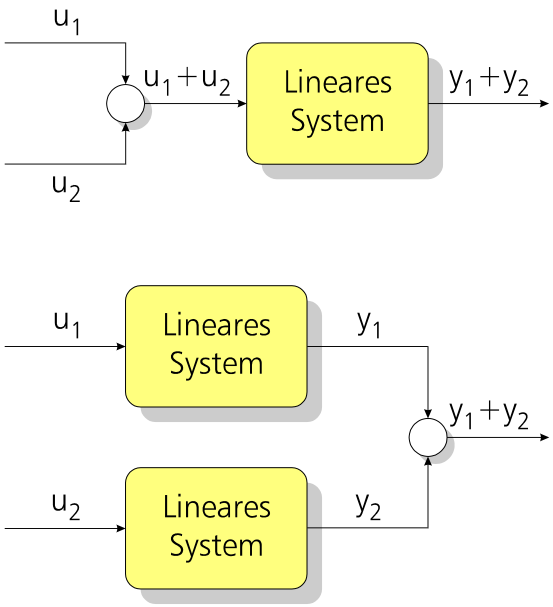

Bild 13 Überlagerungsprinzip: $g(u_1+u_2)=g(u_1)+g(u_2)=y_1+y_2$

☺ "Wenn's für eine Eingangsamplitude klappt, klappt's immer."

☺ Reihenfolge linearer Blöcke vertauschbar: $g(h(u))=h(g(u))$

6.1 Lineare Blöcke

- Integrator
- Differenzierer
- Verstärker (Konstante)
- Summe
- Totzeit
- zusammengesetzte Blöcke: P-T_1, P-T_2, PD-T_1-Filter, PID-Regler
- allgemeine Übertragungsfunktion
- Lineare Zustandsraumdarstellung

6.2 Mechanischer Schwinger zweiter Ordnung

Der einfachste mechanische Schwinger besteht aus einer Masse m, die über eine Feder mit der Federkonstante c und einen Dämpfer mit der Dämpfungskonstante r an das Inertialsystem gekoppelt ist. Dieses System zweiter Ordnung besitzt zwei Energiespeicher (Integratoren), nämlich die Masse zur Speicherung der kinetischen Energie und die Feder als Lageenergiespeicher. Folglich muß es sich durch eine Differentialgleichung zweiter Ordnung oder zwei Zustandsdifferentialgleichungen erster Ordnung beschreiben lassen.

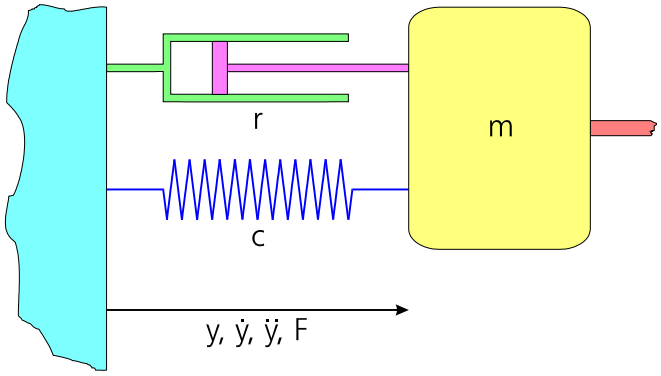

Bild 14 Mechanischer Schwinger zweiter Ordnung

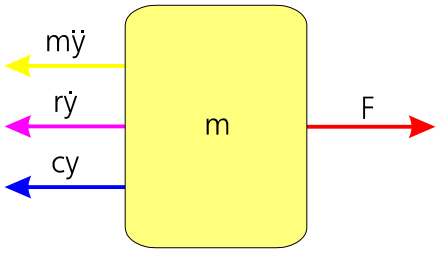

Bild 15 Freigeschnittener Schwinger zweiter Ordnung

Lineare inhomogene Differentialgleichung zweiter Ordnung: $m\ddot{y} + r\dot{y} + cy = F$

Normalform: $\ddot{y} + \frac{r}{m}\dot{y} + \frac{c}{m}y = \frac{F}{m}$

Allgemeines System zweiter Ordnung: $\ddot{y} + 2D\omega_0\dot{y} + \omega_0^2 y = k\omega_0^2 u$

Eigenfrequenz: $\omega_0^2 = \frac{c}{m} \quad \Rightarrow \quad \omega_0 = \sqrt{\frac{c}{m}}$

Dämpfung: $2D\omega_0 = \frac{r}{m} \quad \Rightarrow \quad D = \frac{r}{2m\omega_0} = \frac{r}{2m\sqrt{\frac{c}{m}}} = \frac{r}{2\sqrt{cm}}$

Verstärkungsfaktor: $k\omega_0^2 = \frac{1}{m} \quad \Rightarrow \quad k = \frac{1}{m\omega_0^2} = \frac{1}{m\frac{c}{m}} = \frac{1}{c}$

6.3 Zustandsraumdarstellung

Aufspalten der Differentialgleichung zweiter Ordnung:

$\ddot{y} + 2D\omega_0\dot{y} + \omega_0^2 y = k\omega_0^2 u$

durch Einführen von zwei Zustandsvariablen (Weg und Geschwindigkeit):

$x_1 = y \qquad x_2 = \dot{y}$

in zwei Differentialgleichungen erster Ordnung :

$$\dot{x}_1 = \dot{y} = x_2$$
$$\dot{x}_2 = \ddot{y} = k\omega_0^2 u - 2D\omega_0 \dot{y} - \omega_0^2 y$$

Eliminieren von y und $\dot{y}$ führt zur Zustandsraumdarstellung:

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -\omega_0^2 x_1 - 2D\omega_0 x_2 + k\omega_0^2 u$$

In Matrizenschreibweise:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & -2D\omega_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ k\omega_0^2 \end{bmatrix} u$$

Allgemeine Zustandsraumdarstellung: $\dot{x} = Ax + Bu$

Hier: $x = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} \qquad A = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & -2D\omega_0 \end{bmatrix} \qquad B = b = \begin{bmatrix} 0 \\ k\omega_0^2 \end{bmatrix}$

Allgemeine Ausgangsgleichung in Matrizenschreibweise:

$$y = Cx + Du$$

Hier: $y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]u$

mit: $C = c^T = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad D = d = [0] = 0$

6.4 Blockschaltbild

Zum Zeichnen des Blockschaltbildes eines dynamischen Systems müssen Sie pro Zustand einen Integrator spendieren und am Eingang der Integratoren die jeweiligen Zustandsdifferentialgleichungen modellieren.

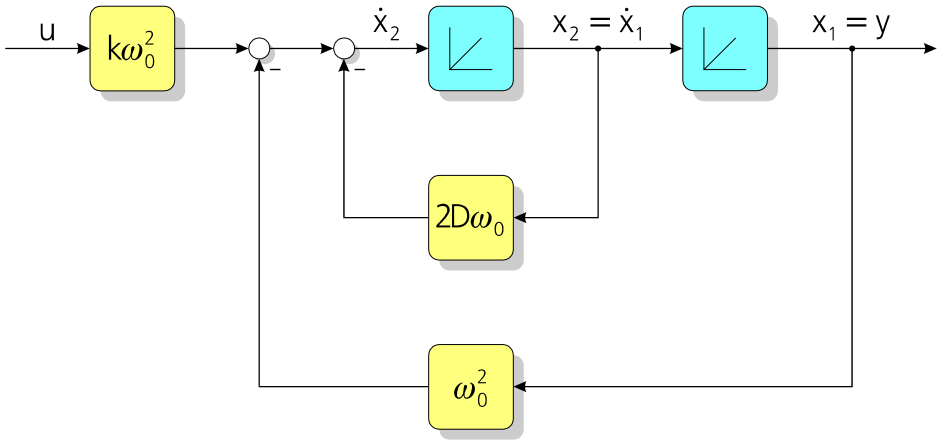

Bild 16 Blockschaltbild eines Schwingers zweiter Ordnung

6.5 Übertragungsfunktion

Bei der Transformation in den Laplace-Bereich spendieren Sie für jeden Ableitungspunkt ein "s" (Anfangsbedingungen gleich Null):

$$\mathcal{L}\{y(t)\} = Y(s)$$
$$\mathcal{L}\{\dot{y}(t)\} = sY(s)$$
$$\mathcal{L}\{\ddot{y}(t)\} = s^2Y(s)$$

Laplace-Transformation der linearen Differentialgleichung zweiter Ordnung:

$$\mathcal{L}\{\langle \ddot{y} + 2D\omega_0\dot{y} + \omega_0^2 y = k\omega_0^2 u\rangle\} = \langle s^2Y + 2D\omega_0 sY + \omega_0^2 Y = k\omega_0^2 U\rangle$$

Übertragungsfunktion ist definiert als "Ausgangsgröße zu Eingangsgröße":

$$F(s) = \frac{Y}{U} = \frac{k\omega_0^2}{s^2 + 2D\omega_0 s + \omega_0^2}$$

Charakteristische Gleichung durch Nullsetzen des Nenners: $s^2 + 2D\omega_0 s + \omega_0^2 = 0$

Pole, Eigenwerte, Nullstellen des Nenners:

$$s_{1,2} = -D\omega_0 \pm \sqrt{D^2\omega_0^2 - \omega_0^2}$$
$$= -D\omega_0 \pm \omega_0\sqrt{D^2 - 1} \qquad \text{für } (|D| > 1)$$
$$= \underbrace{-D\omega_0}_{\sigma} \pm j\underbrace{\omega_0\sqrt{1 - D^2}}_{\omega} \qquad \text{für } (|D| < 1)$$

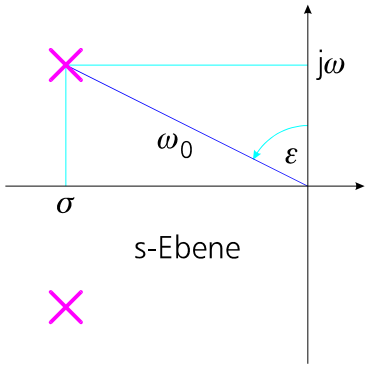

Bild 17 Pole eines Schwingers zweiter Ordnung in der s-Ebene

Eigenfrequenz (für $|D| < 1$) über Pythagoras: $\sigma^2 + \omega^2 = D^2\omega_0{}^2 + \omega_0^2(1-D^2) = \omega_0^2$

Dämpfung über Sinusfunktion: $\sin\varepsilon = \frac{D\omega_0}{\omega_0} = D$

7 Identifikation (Identifizierung)

Identifikation ist das Bestimmen der Parameter eines Modells durch Vergleich mit der Realität. Dazu werden die Parameter des Modells solange von einem Optimierer variiert, bis die Reaktionen von realer Strecke und Modell identisch sind.

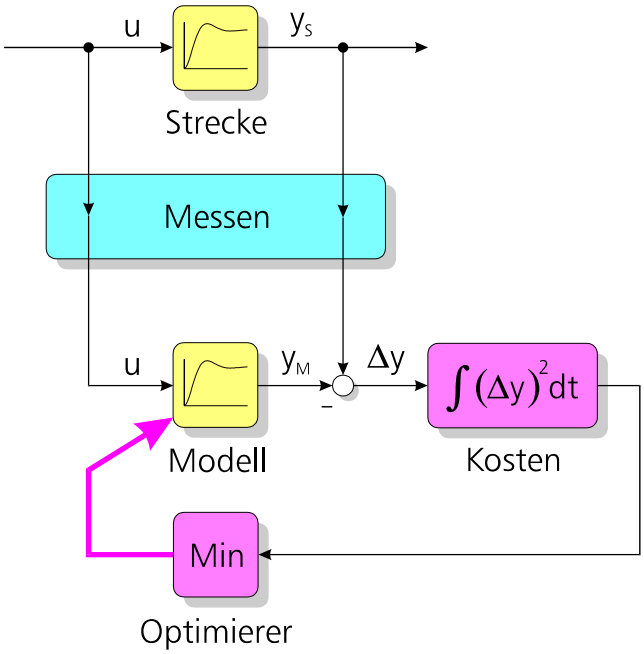

Bild 18 Prinzip der Identifikation

- ✓ Modellparameter sind anfangs falsch: $k_S \neq k_M$ $D_S \neq D_M$ $\omega_{0S} \neq \omega_{0M}$
- ✓ Daher sind die Antworten (Zeitantwort, Frequenzgang) unterschiedlich: $y_S \neq y_M$
- ✓ Ergo existiert ein Schätzfehler Δy, bzw. dessen Zeitintegral $\int(\Delta y)^2 dt$.
- ✓ Ein Optimierer hat die Kontrolle über die Modellparameter und versucht, das Fehlerintegral zu minimieren.

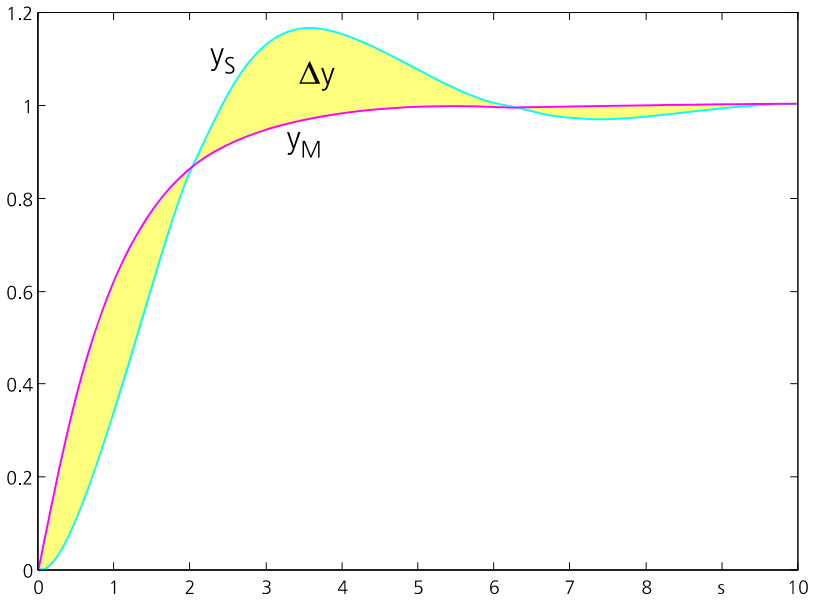

Bild 19 Minimieren des Schätzfehlers

- Auch zur Modellreduktion und Linearisierung verwendbar.
- Verfahren: Least Squares, Maximum Likelihood, Kalman-Filter

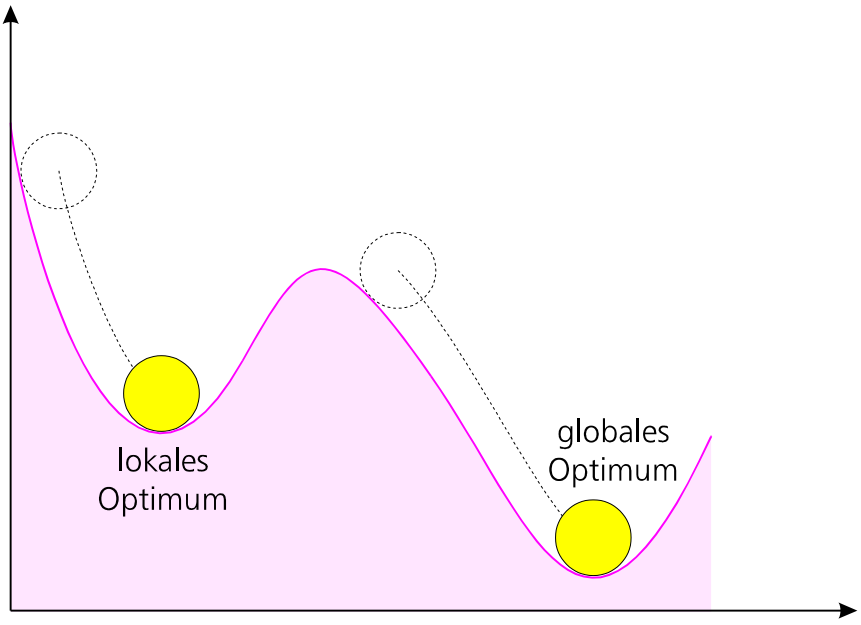

Bild 20 Lokales und globales Optimum

Multimodale Funktion: Klassische Gradientenverfahren (Steepest Descent, ...) finden häufig nur lokale Optima. Daher sind dort die Startwerte sehr wichtig. Andere Verfahren wie Genetische Algorithmen, Evolutionsstrategien oder Simulated Annealing verwenden gerichtet-stochastische Suchstrategien und sind daher wesentlich robuster und sicherer in unwegsamem Terrain. Allerdings sind diese Algorithmen im allgemeinen auch deutlich langsamer als die Gradientenverfahren.

7.1 Genetische Algorithmen

Genetische Algorithmen beispielsweise verteilen eine große Anzahl von “Individuen” über den gesamten zu durchsuchenden Parameterraum und verwenden dann die gleichen drei Prinzipien, die Mutter Natur seit Jahrmillionen zur Optimierung ihrer Individuen benutzt:

- ✓ Entsprechend ihrer Güte werden die besten Individuen der Population ausgewählt (Selektion).
- ✓ Diese besten Individuen tauschen dann in einem Rekombinationsschritt willkürlich Teile ihrer guten Eigenschaften aus und erzeugen in der nächsten Generation Nachkommen, die ähnlich gute Eigenschaften wie ihre Eltern haben, aber doch etwas von diesen verschieden sind.
- ✓ Eine stochastisch durchgeführte Mutation kann innerhalb der Population verlorengegangene Eigenschaften zurückbringen und ganz neue Bereiche des Parameterraumes durchsuchen.

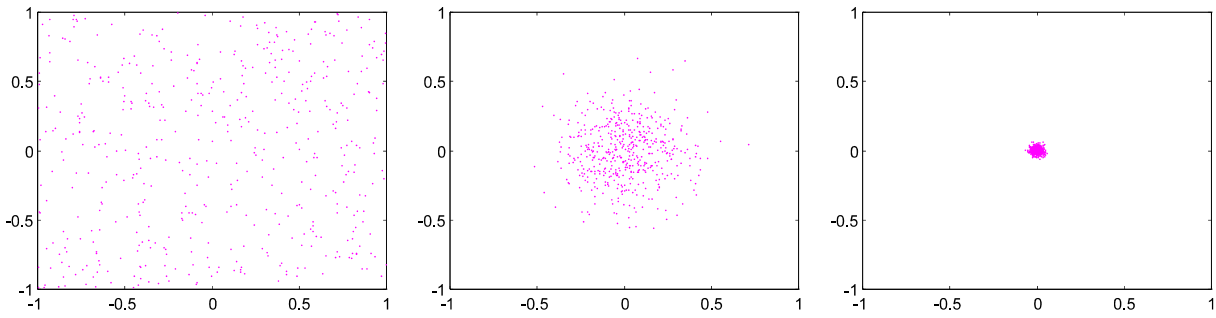

Bild 21 Konvergieren der Population zum Optimum

8 Nichtlineare Systeme

Definitionsgemäß ist “nichtlinear” der allgemeine Oberbegriff für lineare und nicht lineare Systeme. Ein nichtlineares System kann daher sehr wohl linear sein (z. B. bezüglich bestimmter Variablen oder innerhalb eines gewissen Arbeitsbereichs). Physikalisch gesehen ist jedes reale System nicht linear. Bei genügend großen Eingangsamplituden verläßt jedes reale System seinen Linearitätsbereich (Anschlag, Stellgrößenbegrenzung, ...).

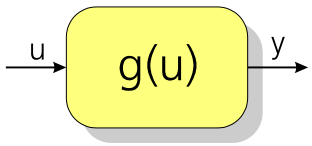

Bild 22 Nichtlineares System

8.1 Linearisierung

Wie schon erwähnt, lassen sich lineare Gleichungen wesentlich einfacher handhaben als die physikalisch realistischeren nichtlinearen. Für fast jede Art der Analyse (Stabilitätsuntersuchung, ...) oder Synthese (Reglerauslegung, ...) muß das nichtlineare System daher um einen geeignet zu definierenden Arbeitspunkt herum linearisiert werden. Meistens wird als Arbeitspunkt eine Ruhelage oder ein stationärer Betriebspunkt gewählt, an dem möglichst viele Größen (Zustände, Ableitungen, ...) Null sind.

Nichtlineare (Differential-)Gleichung in impliziter Form: $f(x_1,x_2) = 0$

Am Arbeitspunkt (Index 0): $x_1 = x_1|_0 = x_{10}$ $\qquad$ $x_2 = x_2|_0 = x_{20}$

(Kleine) Abweichungen vom Arbeitspunkt: $\Delta x_1 = x_1 - x_{10}$ $\qquad$ $\Delta x_2 = x_2 - x_{20}$

Taylorreihenentwicklung, nach dem linearen Glied abgebrochen:

$$f(x_1,x_2) = \underbrace{f(x_1,x_2)|_0}_{0} + \left.\frac{\partial f(x_1,x_2)}{\partial x_1}\right|_0 \Delta x_1 + \left.\frac{\partial f(x_1,x_2)}{\partial x_2}\right|_0 \Delta x_2$$

Lineare Gleichung: $\left.\frac{\partial f(x_1,x_2)}{\partial x_1}\right|_0 \Delta x_1 + \left.\frac{\partial f(x_1,x_2)}{\partial x_2}\right|_0 \Delta x_2 = 0$

Beispiel: Funktion einer Veränderlichen: $y = g(x)$

In impliziter Form: $f(x,y) = g(x) - y = 0$

Linearisierung: $\left.\frac{\partial f(x,y)}{\partial x}\right|_0 \Delta x + \left.\frac{\partial f(x,y)}{\partial y}\right|_0 \Delta y = 0$

Bilden der partiellen Ableitungen: $\left.\frac{\partial f(x,y)}{\partial x}\right|_0 = \left.\frac{dg(x)}{dx}\right|_0 = y'|_0 = y'_0 \qquad \left.\frac{\partial f(x,y)}{\partial y}\right|_0 = -1$

Lineare Gleichung: $y'_0 \Delta x - \Delta y = 0$

Geradengleichung mit Steigung der Funktion im Arbeitspunkt (Tangente):

$\Delta y = y'_0 \Delta x$

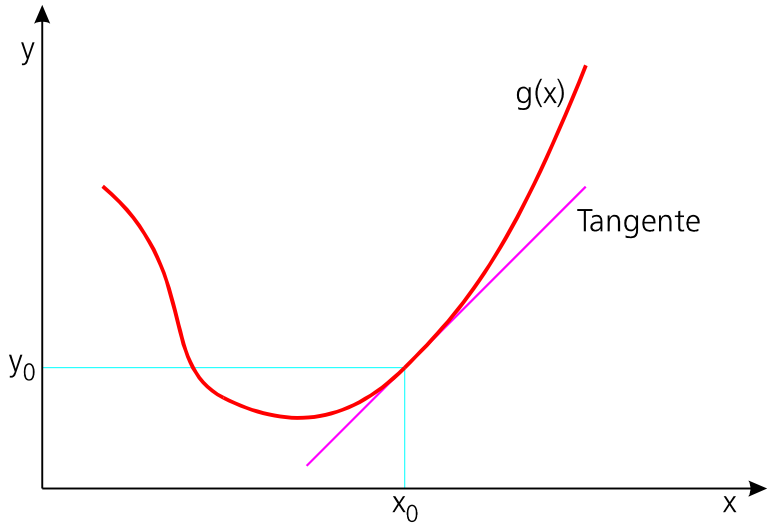

Bild 23 Tangente linearisiert die Funktion im Arbeitspunkt

Die linearisierte Gleichung gilt nur bei kleinen Abweichungen vom Arbeitspunkt (in der Nähe des Arbeitspunktes), solange die Tangente nicht zu sehr von der Funktion abweicht.

8.2 Mathematisches Pendel

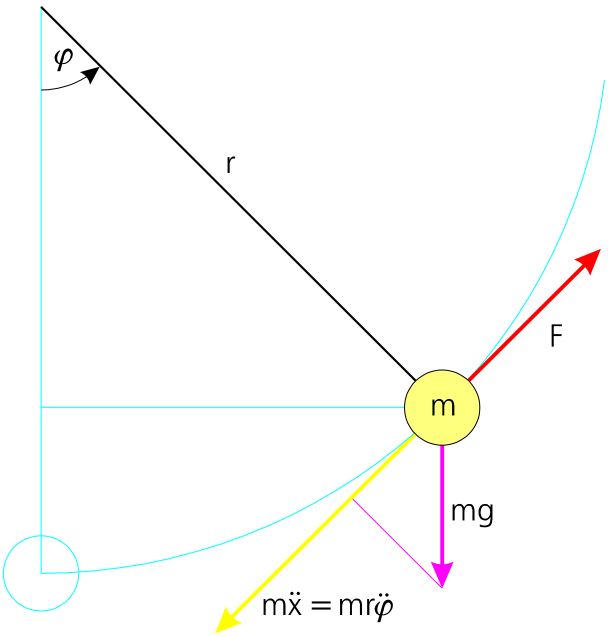

Bild 24 Mathematisches Pendel (Masselose Stange, keine Drehträgheit)

Kräftegleichgewicht in tangentialer Richtung: $mr\ddot{\varphi} + mg\sin\varphi - F = 0$

Implizite nichtlineare Differentialgleichung mit massebezogenen Kräften:

$$f(\ddot{\varphi}, \varphi, S) = r\ddot{\varphi} + g\sin\varphi - S = 0 \qquad \text{mit} \quad S = \frac{F}{m}$$

Linearisierung: $\left.\frac{\partial f}{\partial \ddot{\varphi}}\right|_0 \Delta\ddot{\varphi} + \left.\frac{\partial f}{\partial \varphi}\right|_0 \Delta\varphi + \left.\frac{\partial f}{\partial S}\right|_0 \Delta S = 0$

Partielle Ableitungen für die Linearisierung:

$$\left.\frac{\partial f}{\partial \ddot{\varphi}}\right|_0 = r \qquad \left.\frac{\partial f}{\partial \varphi}\right|_0 = \left(g\cos\varphi\right)\Big|_0 = g\cos\varphi_0 \qquad \left.\frac{\partial f}{\partial S}\right|_0 = -1$$

Linearisierte Differentialgleichung: $r\Delta\ddot{\varphi} + g\cos\varphi_0\Delta\varphi - \Delta S = 0$

Ein besonderer Arbeitspunkt (Ruhelage): $\varphi_0 = \dot{\varphi}_0 = \ddot{\varphi}_0 = 0 \quad \Rightarrow \quad \cos\varphi_0 = 1$

Linearisierte Differentialgleichung mit eingesetztem Arbeitspunkt: $r\Delta\ddot{\varphi} + g\Delta\varphi = \Delta S$

- Sie können jetzt die Deltas weglassen, aber Sie müssen sich immer im Klaren darüber sein, daß es sich bei den Variablen der linearen Gleichung nur um kleine Abweichungen vom Arbeitspunkt handelt.

Lineare Differentialgleichung: $r\ddot{\varphi} + g\varphi = S$

Andere Überlegung: Bei kleinen Winkeln kann der Sinus durch sein Argument ersetzt werden (und der Kosinus durch Eins): Ersetzen von $\sin\varphi$ durch φ direkt in der nichtlinearen Differentialgleichung führt sofort auf die lineare Differentialgleichung. Vorsicht: Funktioniert bei komplexeren Gleichungen und anderen Arbeitspunkten nur mit sehr viel Erfahrung.

Anderer Arbeitspunkt für die Linearisierung:

$$\dot{\varphi}_0 = \ddot{\varphi}_0 = 0, \quad \varphi_0 = \frac{\pi}{2} \quad \Rightarrow \quad \cos\varphi_0 = 0$$

Lineare Differentialgleichung am anderen Arbeitspunkt: $r\Delta\ddot{\varphi} - \Delta S = 0$ bzw.: $r\ddot{\varphi} = S$

9 Trimmrechnung

Die Trimmrechnung dient zur Berechnung fehlender Größen bei der Beschreibung des Zustands, in dem sich ein (nicht)lineares System befindet. Sie wird meist vor einer Simulation durchgeführt, damit sich das zu simulierende System zu Beginn der Simulation in einem definierten (meist stationären) Anfangszustand befindet. Dieser Anfangszustand ist eindeutig festgelegt, wenn sowohl der Zustandsvektor als auch der Eingangsvektor zahlenmäßig bekannt sind. Das Problem besteht nun darin, daß Sie von diesem Anfangszustand üblicherweise nur ein paar Zustands- und Eingangsgrößen kennen, während andere Zustands- und Eingangsgrößen (die Trimmgrößen) unbekannt sind. Zusätzlich haben Sie noch Forderungen für diverse Zustandsableitungen und Ausgangsgrößen (Trimmforderungen). Ein nichtlinearer Gleichungslöser ermittelt dann durch Einsetzen der Trimmforderungen in die Differential- und Ausgangsgleichungen die gesuchten Trimmgrößen.

Das in sich schlüssige Aufstellen von sinnvollen Trimmforderungen bei gleichzeitigem Freigeben korrespondierender Trimmgrößen ist bei halbwegs realistischen Systemen keineswegs trivial. Es erfordert vielmehr vom Ingenieur ein fundiertes Verständnis des zu trimmenden Systems und dessen innerer physikalischen Zusammenhänge.

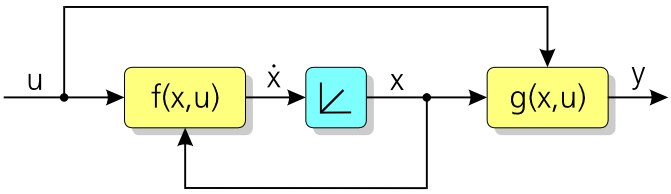

Bild 25 Allgemeines nichtlineares System

Vektordifferentialgleichung: $\dot{x} = f(x,u)$

Ausgangsvektorgleichung: $y = g(x,u)$

Trimmforderungen: Elemente von $\dot{x}$ und y

Trimmgrößen: Elemente von x und u

- ✓ Für jede Trimmforderung eine Trimmgröße freilassen, die die Forderung erfüllen (oder wenigstens beeinflussen) kann.
- ✓ Für jede Trimmgröße eine Trimmforderung aufstellen, durch die die Trimmgröße definiert (oder wenigstens eingeschränkt) wird.
- ✓ Die Elemente von x und u, die keine Trimmgrößen sind, auf feste Werte setzen.

- ✓ Die Elemente von $\dot{x}$, die keine Trimmforderungen sind, folgen automatisch.
- ● Vorsicht: Der Trimmpunkt ist bei nichtlinearen Systemen häufig nicht eindeutig. Es gibt also möglicherweise mehrere verschiedene Trimmgrößenkombinationen, die die Trimmforderungen erfüllen. Es ist daher immer sinnvoll, den Gleichungslöser mit vernünftigen Anfangswerten der Trimmgrößen zu versorgen, die in der Nähe der zu erwartenden Lösungen liegen. Außerdem verringert dies die Rechenzeit der Trimmrechnung.

9.1 Nichtlineares Pendel

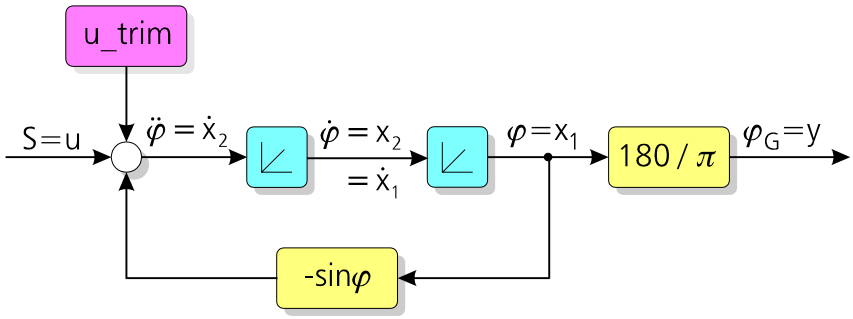

Bild 26 Blockschaltbild des nichtlinearen mathematischen Pendels

Ein doppelter Integrator mit Sinus-Rückführung ist ein ungedämpfter nichtlinearer Schwinger. Der Ausgangsgrößenblock rechnet Winkel von Bogenmaß in Grad um. Der u_trim-Block kann zum Vorgeben eines stationären Trimmwertes der Eingangsgröße genutzt werden.

Trimmzustand: Das Pendel soll bei einem Winkel von 45° stehen.

Trimmforderungen und selbsteinstellende Größen			Trimmgrößen und gesetzte Größen		
Zustandsableitungen $\dot{x}$		Ausgänge y	Zustände x		Eingänge u
$\dot{\varphi}$	$\ddot{\varphi}$	φ_G	φ	$\dot{\varphi}$	S
stellt sich ein	Null gefordert	45° gefordert	trimmen	Null gesetzt	trimmen
⑥	④	①	②	⑤	③

① Erste Trimmforderung: Winkel in Grad soll 45° betragen.

② Dann muß eine Trimmgröße freigelassen werden. Der Zustand φ kann so getrimmt werden, daß die erste Trimmforderung erfüllt wird. In diesem einfachen Fall ist das Trimmergebnis trivial: Die Trimmgröße φ wird natürlich auf $\frac{\pi}{4}$ getrimmt, was genau den 45° der Trimmforderung entspricht.

③ Eingang S (Trimmgröße) soll getrimmt werden.

④ Dann muß eine Trimmforderung gestellt werden, die durch die Trimmgröße S beeinflußt werden kann. Es wird daher $\ddot{\varphi}$ zu Null gefordert.

⑤ Der Zustand $\dot{\varphi}$ (keine Trimmgröße) wird zu Null gesetzt, muß also nicht getrimmt werden.

⑥ Da $\dot{\varphi}$ nicht nur ein Zustand, sondern gleichzeitig auch eine Zustandsableitung ist, folgt die Zustandsableitung $\dot{\varphi}$ trivialerweise automatisch aus dem Setzen des Zustands $\dot{\varphi}$. Analog könnte man auch die Zustandsableitung $\dot{\varphi}$ als Trimmforderung zu Null fordern und den Zustand $\dot{\varphi}$ trimmen lassen, was aber die Anzahl der zu trimmenden Größen unnötigerweise erhöhen würde.

10 Numerische Integration

Nur die allerwenigsten Differentialgleichungen lassen sich mit vertretbarem Aufwand analytisch lösen. Alle anderen müssen numerisch integriert (gelöst) werden.

- Anfangswertproblem: Finde Zeitantwort $x(t)$ für gegebenes $u(t)$ und gegebenen Anfangszustand $x(t=0)$:

Lineares (hier zeitinvariantes) System: $\dot{x}(t) = Ax(t) + Bu(t)$

Nichtlineares (hier zeitvariantes) System: $\dot{x}(t) = f(x(t), u(t), t)$

- ✓ Lineare Systeme am besten diskretisieren und mit Hilfe der Transitionsmatrix lösen. Oder wie nichtlineare Systeme integrieren.
- ✓ Nichtlineare Systeme in Differentialgleichungssysteme erster Ordnung zerlegen und jede einzeln mittels numerischem Integrationsverfahren über der Zeit integrieren.

10.1 Diskretisierung eines linearen Systems

Zur Berechnung der Gesamtlösung einer linearen Differentialgleichung berechnen Sie zuerst die homogene Lösung (rechte Seite gleich Null) und machen dann für die Partikulärlösung einen Ansatz vom Typ der rechten Seite.

Die Lösung des homogenen linearen Vektordifferentialgleichungssystems:

$$\dot{x} - Ax = 0$$

lautet, wovon Sie sich durch Einsetzen leicht überzeugen können:

$$x_h = e^{At}k \qquad \dot{x}_h = Ae^{At}k$$

Für die Partikulärlösung gehen Sie im einfachsten Fall davon aus, daß sich der Eingangsvektor **u** im betrachteten Zeitraum nicht ändert. Wenn also die rechte Seite konstant ist, ($u = const.$) können Sie auch die partikuläre Lösung als Konstante ansetzen:

$$x_p = const. \qquad \dot{x}_p = 0$$

Ins Differentialgleichungssystem eingesetzt ergibt sich:

$$0 = Ax_p + Bu$$

Unter der Voraussetzung, daß die Systemmatrix **A** nicht singulär ist, können Sie direkt nach der partikulären Lösung auflösen:

$$x_p = -A^{-1}Bu$$

Die Gesamtlösung ist dann die Summe aus homogener und partikulärer Lösung:

$$x = x_h + x_p$$
$$x(t) = e^{At}k - A^{-1}Bu$$

Der Zustand einen Abtastschritt Δt später ergibt sich zu:

$$x(t+\Delta t) = e^{A(t+\Delta t)}k - A^{-1}Bu$$

Nach dem Ausmultiplizieren des Exponenten und dem Aufspalten in zwei e-Funktionen:

$$x(t+\Delta t) = e^{A\Delta t}e^{At}k - A^{-1}Bu$$

kann **k** durch Einsetzen des Zustands zum Zeitpunkt t eliminiert werden:

$$x(t+\Delta t) = e^{A\Delta t}\left(x(t) + A^{-1}Bu\right) - A^{-1}Bu$$

Nach Zusammenfassen der nicht von **x** abhängigen Glieder ergibt sich die rekursive Matrixdifferenzengleichung:

$$\begin{aligned} x(t+\Delta t) &= e^{A\Delta t}x(t) + \left(e^{A\Delta t} - E\right)A^{-1}Bu \\ &= Fx(t) + Gu \end{aligned}$$

Diese diskrete Differenzengleichung gibt Ihnen jetzt die Möglichkeit, die Matrizen **F** und **G** für die Schrittweite Δt einmalig zu berechnen und dann rekursiv den jeweils neuen Zustand $x(t+\Delta t)$ durch zwei Matrix-Vektor-Multiplikationen und eine Vektoraddition aus dem alten Zustand $x(t)$ und dem im Intervall konstant angenommenen Eingangsvektor **u** zu berechnen.

Die Genauigkeit, mit der Sie beispielsweise auch über sehr große Intervalle "integrieren" können, hängt dabei weitestgehend von der exakten Berechnung der Matrixexponentialfunktion $e^{A\Delta t}$ ab. Die korrekte Ermittlung dieser Transitionsmatrix wiederum ist ein numerisch keineswegs triviales Problem. Gute numerische Bibliotheken stellen Ihnen daher mehrere Verfahren zur Verfügung (Pade-Approximation, Taylorreihenentwicklung, Eigensystem, ...), deren Ergebnisse Sie im Einzelfall kritisch miteinander vergleichen können. Auch die Matrix **G** wird in der Praxis natürlich nicht über die direkte Auswertung von $\left(e^{A\Delta t} - E\right)A^{-1}Bu$ berechnet, sondern mit numerisch

stabileren Algorithmen aus Bibliotheken, die dann auch mit singulären Systemmatrizen klarkommen.

10.2 Näherungslösungen durch direkte Transformation

Wenn der numerische Aufwand zum exakten Berechnen der Transitionsmatrix zu groß ist, oder kein numerisch stabiler Algorithmus dafür verfügbar ist, lassen sich Näherungslösungen finden, die durch direkte Substitution des Laplace-Operators zu vereinfachten Differenzengleichungen führen. Zur Herleitung der verschiedenen Transformationsverfahren drückt man das Übertragungsverhalten eines reinen Integrators im (Zeit-,) Laplace- und z-Bereich aus und gelangt durch einfache Analogiebetrachtungen zu praktischen Ersetzungsregeln:

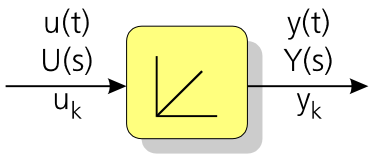

Bild 27 Der Integrator

Integrator im Zeitbereich: $y(t) = \int_0^t u(\tau)\,d\tau$

Integrator im Laplace-Bereich: $\mathcal{L}\{y(t)\} = Y(s) = \frac{1}{s}U(s)$

Übertragungsfunktion: $F(s) = \frac{Y(s)}{U(s)} = \frac{1}{s}$

Integral über einen Abtastschritt: $y(t+\Delta t) = y(t) + \int_t^{t+\Delta t} u(\tau)\,d\tau$

Diskretisierung des Abtastschrittes (Differenzengleichung):

$$y_{k+1} = y_k + \int_{t_k}^{t_{k+1}} u(\tau)\,d\tau = y_k + \Im$$

Unterschiedliche Berechnungsverfahren für das Integral $\Im$:

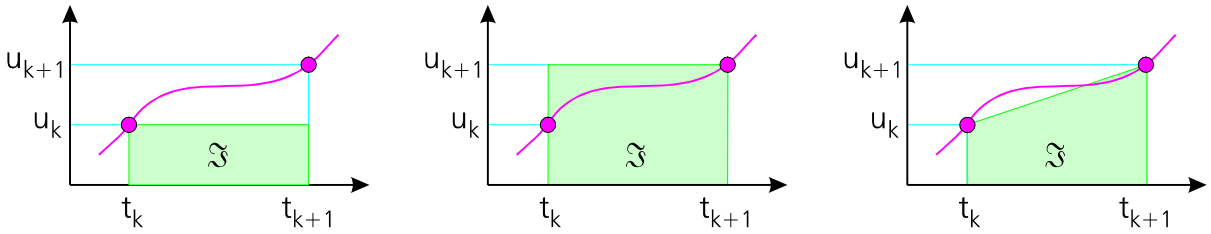

Bild 28 Untersumme, Obersumme und Trapezregel

Analogie zwischen Laplace-Operator “s” und diskretem Operator “z”:

Laplace-Trafo von Ableitung und Integral:

$$\mathcal{L}^{-1}\{s\cdot X(s)\} = \frac{dx(t)}{dt} \qquad \mathcal{L}^{-1}\left\{\frac{X(s)}{s}\right\} = \int x(t)\,dt$$

Z-Trafo von nächstem und vorherigem Wert: $z\cdot x_k = x_{k+1}$ $\qquad \frac{x_k}{z} = x_{k-1}$

- Genauso, wie die Multiplikation mit “s” im Laplace-Bereich eine zeitliche Ableitung einer Größe beschreibt und die Division durch “s” eine Integration, so bedingt die Multiplikation bzw. Division mit “z” im Diskreten eine Verschiebung einer Größe in positive bzw. negative Zeitrichtung.

Integralnäherung	$\Im = u_k\Delta t$	$\Im = u_{k+1}\Delta t$	$\Im = \frac{u_k + u_{k+1}}{2}\Delta t$
Differenzengleichung	$y_{k+1} = y_k + u_k\Delta t$	$y_{k+1} = y_k + u_{k+1}\Delta t$	$y_{k+1} = y_k + \frac{u_k + u_{k+1}}{2}\Delta t$
Differenzengleichung in z	$zy_k = y_k + u_k\Delta t$	$zy_k = y_k + zu_k\Delta t$	$zy_k = y_k + \frac{u_k + zu_k}{2}\Delta t$
z-Übertragungsfunktion	$F(z) = \frac{y_k}{u_k} = \frac{1}{z-1}\Delta t$	$F(z) = \frac{z}{z-1}\Delta t$	$F(z) = \frac{z+1}{z-1}\cdot\frac{\Delta t}{2}$
Ersetze s in F(s) durch	$s \Leftrightarrow \frac{z-1}{1}\cdot\frac{1}{\Delta t}$	$s \Leftrightarrow \frac{z-1}{z}\cdot\frac{1}{\Delta t}$	$s \Leftrightarrow \frac{z-1}{z+1}\cdot\frac{2}{\Delta t}$
Bezeichnung der Transformation	Zero order hold, Untersumme	Zero order hold, Obersumme	First order hold, Tustin-Formel, Trapezregel

10.2.1 Beispiel am Tiefpaß erster Ordnung

Übertragungsfunktion P-T_1: $F(s) = \frac{1}{Ts+1}$

Untersumme: Ersetze “s” in $F(s)$ nach Tabelle durch $\frac{z-1}{1}\cdot\frac{1}{\Delta t}$

um $F(z)$ zu erhalten: $F(z) = \frac{y_k}{u_k} = \frac{1}{T\left(\frac{z-1}{1} \cdot \frac{1}{\Delta t}\right) + 1} = \frac{1}{\frac{T}{\Delta t}(z-1) + 1} = \frac{\Delta t}{T(z-1) + \Delta t}$

Differenzengleichung in “z”: $y_k\left(T(z-1) + \Delta t\right) = u_k \Delta t$

Multiplikation mit “z” bedeutet “nächster Wert” (k+1):

Differenzengleichung : $y_{k+1} T + y_k(\Delta t - T) = u_k \Delta t$

Auflösen nach “neuem Wert”:

$$y_{k+1} = y_k \frac{(T - \Delta t)}{T} + u_k \frac{\Delta t}{T}$$
$$= y_k\left(1 - \frac{\Delta t}{T}\right) + u_k \frac{\Delta t}{T}$$
$$= y_k(1 - q_T) + u_k q_T$$

Diese rekursive Gleichung für einen diskreten Tiefpaß erster Ordnung kann direkt und sehr einfach implementiert werden:

- Berechne einmalig $q_T = \frac{\Delta t}{T}$.
- Speichere jeweils den letzten Ausgangswert y_k.
- Berechne zu jedem Zeitpunkt das neue y_{k+1} aus dem aktuellen Eingangswert u_k und dem gespeicherten Ausgangswert y_k nach der rekursiven Gleichung.

10.3 Integrationsverfahren

Zur Lösung nichtlinearer Differentialgleichungen ist die numerische Integration über jeweils einen Abtastschritt erforderlich:

$$\int_t^{t+\Delta t} \dot{x}(t)\, dt = \int_t^{t+\Delta t} f(x(t), u(t), t)\, dt$$

$$x(t + \Delta t) - x(t) = \int_t^{t+\Delta t} f(x(t), u(t), t)\, dt$$

$$x(t + \Delta t) = x(t) + \int_t^{t+\Delta t} f(x(t), u(t), t)\, dt$$

$$x(t + \Delta t) = x(t) + \Delta x$$

Der neue Wert zum Zeitpunkt $t + \Delta t$ ergibt sich aus dem alten Wert zum Zeitpunkt t plus einem Integral Δx, das von den verschiedenen Algorithmen unterschiedlich berechnet wird.

- Runge-Kutta: “Das Arbeitspferd”, in verschiedenen Ordnungen:

- 1. Ordnung: Euler
- 2. Ordnung: Mittelpunktsregel
- 4. Ordnung: RK4, “Der Klassische”

- Adams-Bashforth-Moulton: Prädiktor-Korrektor
- Bulirsch-Stoer: Richardson’s Extrapolation, “... wenn’s etwas genauer sein darf...”

10.4 Euler (Runge-Kutta 1. Ordnung)

Euler eignet sich hervorragend, um das Prinzip der numerischen Integration von Differentialgleichungen zu erklären. Er sollte aber in der Praxis nicht eingesetzt werden, da er im Vergleich zu anderen Verfahren ungenau, langsam und instabil ist. Trotzdem findet man ihn immer wieder in den üblichen “quick and dirty”-Anwendungen, in denen man ja nur mal schnell etwas ausprobieren wollte. Außerdem wird Euler manchmal in Echtzeitsimulationen eingesetzt, in denen die wirklich korrekte Implementation der Zwischenschritte anderer Verfahren ziemlich umständlich werden kann.

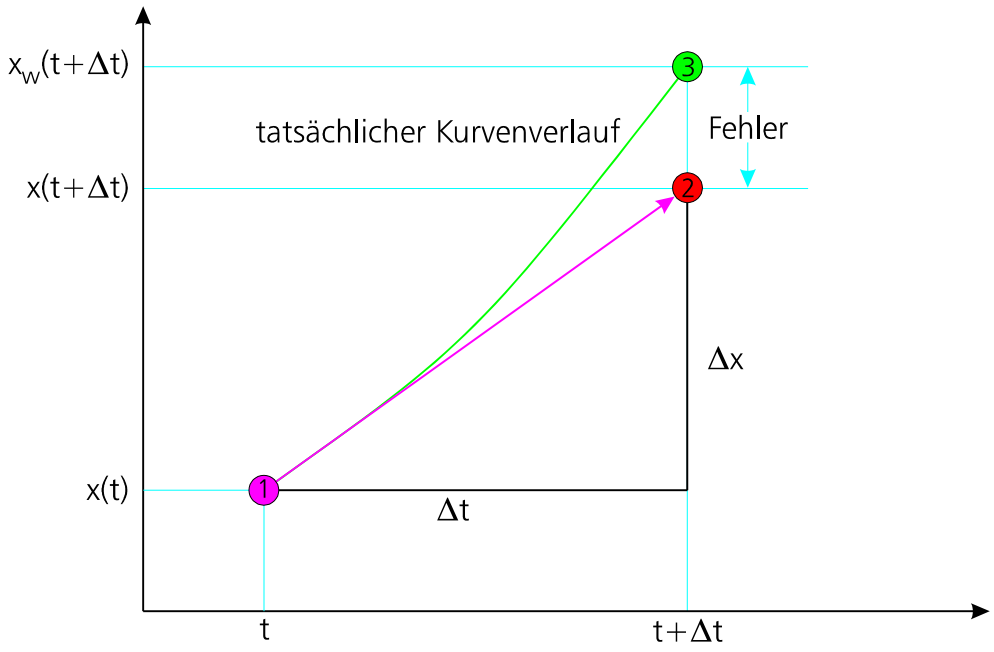

Bild 29 Integration nach Euler (Runge-Kutta 1. Ordnung)

Das Integral Δx ergibt sich direkt aus der Steigung der Kurve zum Zeitpunkt t:

Steigungsdreieck (Tangente): $\Delta x = \dot{x}(t)\Delta t = f(x(t),u(t),t)\Delta t$

Neuer Zustand: $x(t+\Delta t) = x(t) + \Delta x = x(t) + f(x(t),u(t),t)\Delta t$

Fehler zwischen wahrem Wert $x_w(t+\Delta t)$ und berechnetem Wert $x(t+\Delta t)$

10.4.1 Geometrische Interpretation

- Lege zum Zeitpunkt t die Tangente an die Kurve.
- Berechne den neuen Wert $x(t+\Delta t)$ als Wert der Tangente zum Zeitpunkt $t+\Delta t$.

10.5 Mittelpunktsregel (Runge-Kutta 2. Ordnung)

Auch die Mittelpunktsregel findet ihre hauptsächliche Bedeutung in der Erklärung der Verallgemeinerung von Euler in Richtung auf Verfahren höherer Ordnung. Außerdem kann sie in modifizierter Form von Bulirsch-Stoer als Integrationsschritt verwendet werden (s. u.).

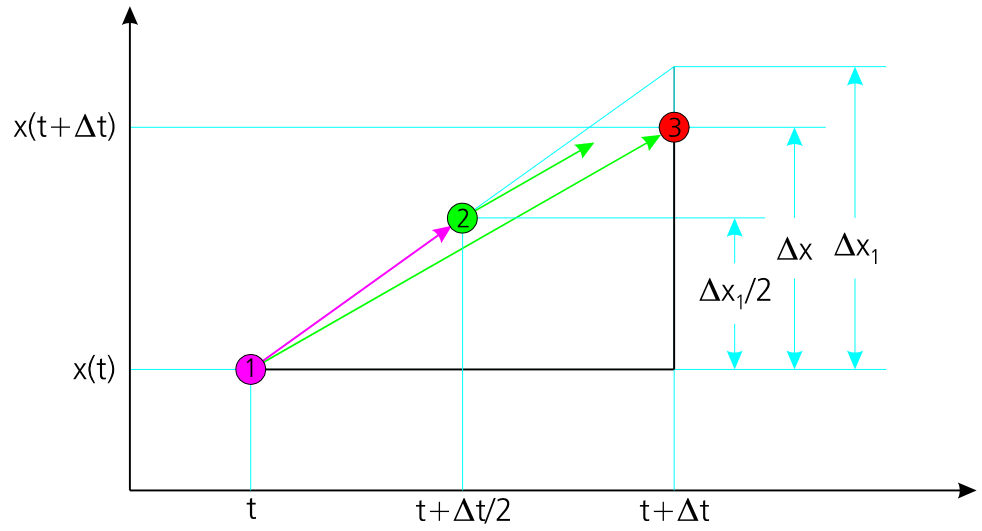

Bild 30 Mittelpunktsregel (Runge-Kutta 2. Ordnung)

Berechnung des Integrals in zwei Schritten:

$$\Delta x_1 = f(x(t),u(t),t)\Delta t$$

$$\Delta x = f\left(x(t)+\tfrac{\Delta x_1}{2},u(t+\tfrac{\Delta t}{2}),t+\tfrac{\Delta t}{2}\right)\Delta t$$

10.5.1 Geometrische Interpretation

- Lege zum Zeitpunkt t die Tangente an die Kurve.
- Berechne die Steigung der Kurve für den Punkt der Tangente zum Zeitpunkt $t+\frac{\Delta t}{2}$.

- Verwende diese Steigung für eine Sekante durch die Kurve zum Zeitpunkt t.
- Berechne den neuen Wert $x(t+\Delta t)$ als Wert der Sekante zum Zeitpunkt $t+\Delta t$.

10.6 “Der Klassische” (Runge-Kutta 4. Ordnung)

Praktisch jede Simulationsumgebung stellt neben anderen Integrationsroutinen auch einen Runge-Kutta 4. Ordnung zur Verfügung. RK4 ist das “Arbeitspferd” der Simulation. Gutmütig und behäbig zieht RK4 mit hinreichend kleinen Schritten auch durch diskontinuierliches, stark nichtlineares Terrain. Steife Systeme, die Anteile mit stark unterschiedlicher Dynamik besitzen, liegen RK4 allerdings nicht besonders gut.

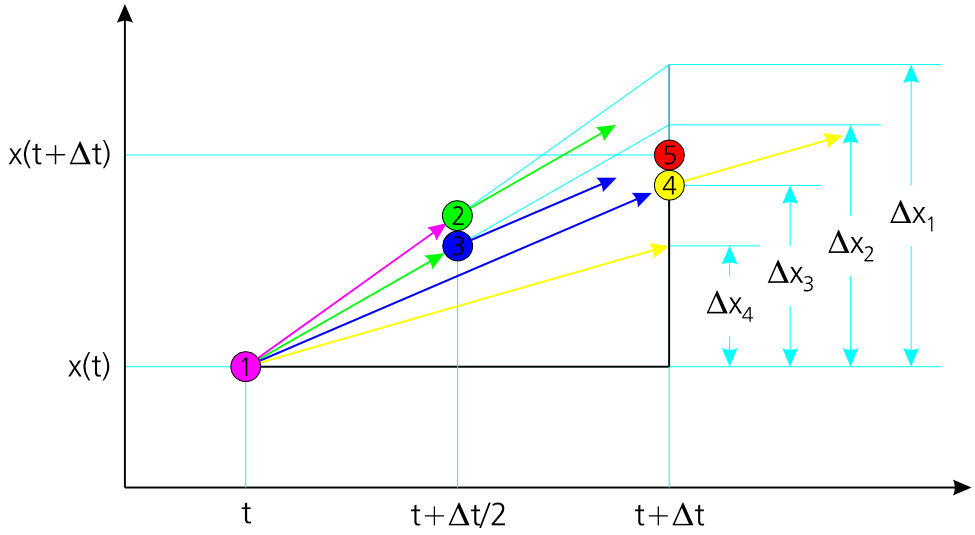

Bild 31 “Der Klassische” (Runge-Kutta 4. Ordnung)

Berechnung des Integrals in 5 Schritten:

$$\Delta x_1 = f\left(x(t), u(t), t\right)\Delta t$$

$$\Delta x_2 = f\left(x(t) + \tfrac{\Delta x_1}{2}, u\left(t + \tfrac{\Delta t}{2}\right), t + \tfrac{\Delta t}{2}\right)\Delta t$$

$$\Delta x_3 = f\left(x(t) + \tfrac{\Delta x_2}{2}, u\left(t + \tfrac{\Delta t}{2}\right), t + \tfrac{\Delta t}{2}\right)\Delta t$$

$$\Delta x_4 = f\left(x(t) + \Delta x_3, u(t + \Delta t), t + \Delta t\right)\Delta t$$

$$\Delta x = \tfrac{\Delta x_1}{6} + \tfrac{\Delta x_2}{3} + \tfrac{\Delta x_3}{3} + \tfrac{\Delta x_4}{6}$$

10.6.1 Geometrische Interpretation

- Lege zum Zeitpunkt t die Tangente an die Kurve.
- Berechne die Steigung der Kurve für den Punkt der Tangente zum Zeitpunkt $t + \frac{\Delta t}{2}$.
- Verwende diese Steigung für eine erste Sekante durch die Kurve zum Zeitpunkt t.
- Berechne die Steigung der Kurve für den Punkt der ersten Sekante bei $t + \frac{\Delta t}{2}$.
- Verwende diese Steigung für eine zweite Sekante durch die Kurve zum Zeitpunkt t.
- Berechne die Steigung der Kurve für den Punkt der zweiten Sekante bei $t + \Delta t$.
- Verwende diese Steigung für eine dritte Sekante durch die Kurve zum Zeitpunkt t.
- Berechne das Integral Δx als gewichteten Mittelwert der Integrale aller vier Schätzgeraden (Tangente und drei Sekanten).

10.7 Prädiktor-Korrektor (Adams-Bashforth-Moulton, ...)

Diese Verfahren sind genauer und effektiver als Runge-Kutta bei glatten Zeitverläufen. Nachteil: Sie sind nicht selbststartend (s. u.). Auch sie kommen nicht gut mit steifen Systemen klar.

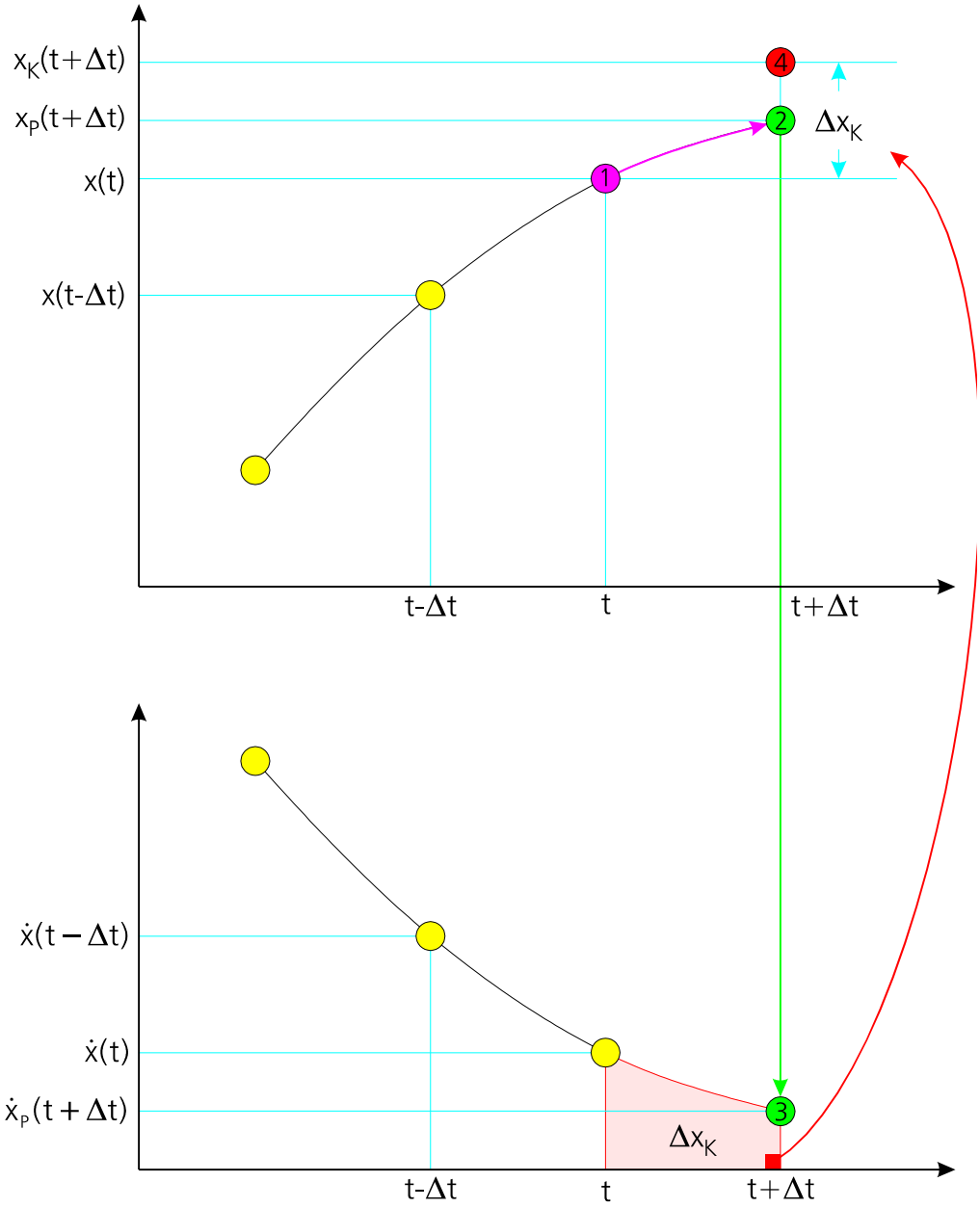

Bild 32 Adams-Bashforth-Moulton (Prädiktor-Korrektor)

10.7.1 Prädiktor

- Schätze einen Prädiktionswert $x_P(t+\Delta t)$, z. B. durch Extrapolation (Polynom, Spline, ...) aus alten Zuständen und Ableitungen $x(t), \dot{x}(t), x(t-\Delta t), \dot{x}(t-\Delta t), \ldots$. Dabei muß das Differentialgleichungssystem nicht ausgewertet werden.

10.7.2 Korrektor

- Berechne Ableitung des Prädiktionswertes $\dot{x}_P(t+\Delta t) = f(x_P(t+\Delta t), u(t+\Delta t), t+\Delta t)$.

- Lege ein Polynom durch $\ldots, \dot{x}(t-\Delta t), \dot{x}(t), \dot{x}(t+\Delta t)$.
- Berechne das Integral Δx_K als Fläche unter der Kurve zwischen $[t, t+\Delta t]$:
  $$\Delta x_K = \int_t^{t+\Delta t} \dot{x}(\tau)\, d\tau$$
- Verwende $\Delta x = \Delta x_K$, wie bei allen anderen Verfahren, zur Berechnung des neuen, korrigierten Wertes: $x(t+\Delta t) = x_K(t+\Delta t) = x(t) + \Delta x_K$.

Da Zustände und Ableitungen aus der Vergangenheit benötigt wird, können Prädiktor-Korrektor-Verfahren nicht selbst starten. Abhilfe: Zu Beginn ein paar Runge-Kutta-Schritte benutzen.

10.7.3 Euler implizit als Prädiktor-Korrektor

- Berechne einen Prädiktionswert durch einen gewöhnlichen expliziten Euler-Schritt: $x_P(t+\Delta t) = x(t) + \Delta x_P = x(t) + \dot{x}(t)\Delta t = x(t) + f(x(t), u(t), t)\Delta t$
- Berechne Ableitung des Prädiktionswertes $\dot{x}_P(t+\Delta t) = f(x_P(t+\Delta t), u(t+\Delta t), t+\Delta t)$.
- Für den Korrektorschritt gibt es jetzt mindestens drei Möglichkeiten, in der Ableitungskurve das Integral zu berechnen:
  1. Explizit als Obersumme: $\Delta x_K = \int_t^{t+\Delta t} \dot{x}(\tau)\, d\tau = \dot{x}(t)\Delta t$. Bringt nichts, da dann der Korrektor gleich dem Prädiktor ist: $\Delta x_K = \Delta x_P \quad \Rightarrow \quad x_K = x_P$
  2. Implizit als Untersumme: $\Delta x_K = \dot{x}_P(t+\Delta t)\Delta t$.
  3. Implizit mit der Trapezregel: $\Delta x_K = \dfrac{\dot{x}(t) + \dot{x}_P(t+\Delta t)}{2}\Delta t$

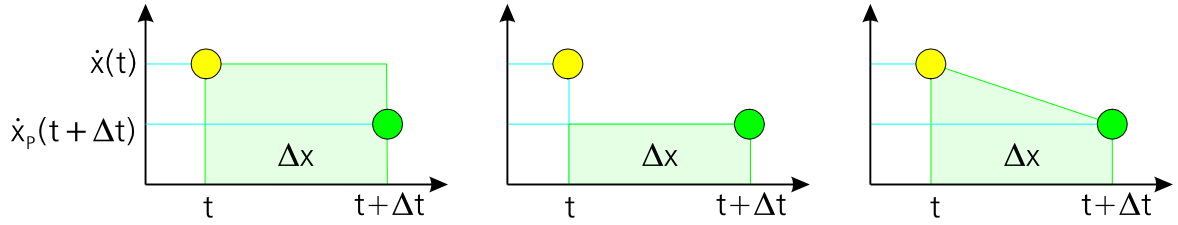

Bild 33 Obersumme, Untersumme und Trapezregel

10.8 Bulirsch-Stoer (Richardson's Extrapolation)

Wenn RK4 der gutmütige, träge "Ackergaul" ist, dann kann Bulirsch-Stoer wohl als sensibles "Rennpferd" bezeichnet werden. Falls die zu simulierenden Differentialgleichungen "glatt" sind, also keine Unstetigkeiten oder zu viele scharfe Nichtlinearitäten beinhalten, und wenn ein großes Intervall mit

möglichst hoher Präzision und minimalen Rechenaufwand integriert werden soll, dann sollte Bulirsch-Stoer das Verfahren Ihrer Wahl sein. Vorausgesetzt, Ihre Simulationsumgebung stellt Ihnen ein BS-Verfahren zur Verfügung oder Sie kommen irgendwie an entsprechende, implementierbare Quellen. Vom Selberbasteln solcher High-Tech-Integrationsverfahren sollte der numerisch durchschnittlich vorbelastete Ingenieur am besten Abstand nehmen.

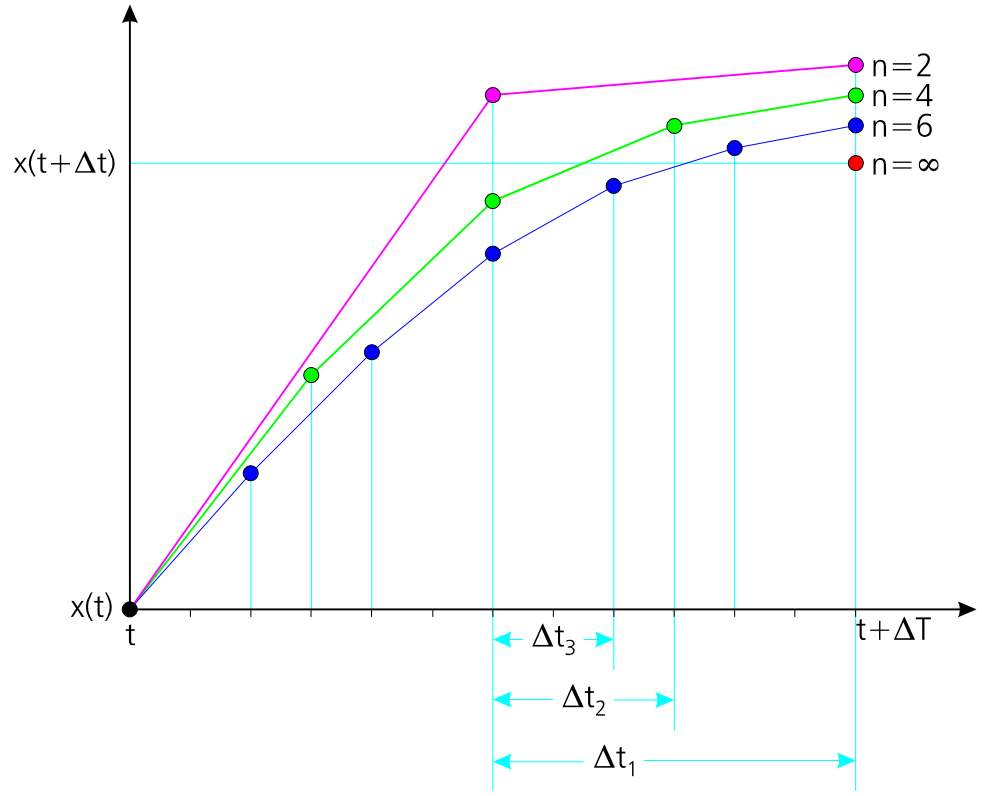

Bild 34 Bulirsch-Stoer (Richardson's Extrapolation)

- Aufgabe: Großes Intervall ΔT besonders genau und schnell zu integrieren.
- In immer feineren Schritten $\Delta t = \frac{\Delta T}{n}$ mit n=2, 4, 6, 8, 12, 16, 24, 32, ... integrieren (z. B. viele RK2), so daß zu jeder Schrittweite Δt_i ein Integrationsergebnis x_i gehört.
- Durch alle Punkte $\{\Delta t_i, x_i\}$ eine gebrochen rationale Interpolationsfunktion legen: $R(\Delta t) = \frac{b_0 + b_1\Delta t + b_2\Delta t^2 + \cdots}{a_0 + a_1\Delta t + a_2\Delta t^2 + \cdots}$
- Extrapolieren der Interpolationsfunktion auf den Wert für beliebig kleine Schrittweiten liefert optimalen Schätzwert für Zustand am Ende des großen Intervalls: $x(t + \Delta T) = \lim_{\Delta t \to 0} R(\Delta t)$

Die Extrapolation wird nach jeder neuen Schrittweite gemacht und liefert Schätzwerte sowohl für den Zustand als auch für den Fehler zurück. Das Verfahren wird abgebrochen, wenn der Fehler kleiner als die vorgegebene Schranke geworden ist.

10.9 Variable Schrittweite

Jedes Verfahren (Runge-Kutta, Adams-Bashforth, ...) kann und sollte mit Schrittweitenkontrolle und -anpassung ausgestattet werden. Wenn die Zeitantwort und die Eingangsfunktion sehr glatt und kontinuierlich (z. B. langsame Eigenwerte bei linearem System) sind, kann mit großer Schrittweite integriert werden. Bei schnellen Eigenwerten, Unstetigkeiten, Diskontinuitäten, ... muß die Schrittweite verkleinert werden.

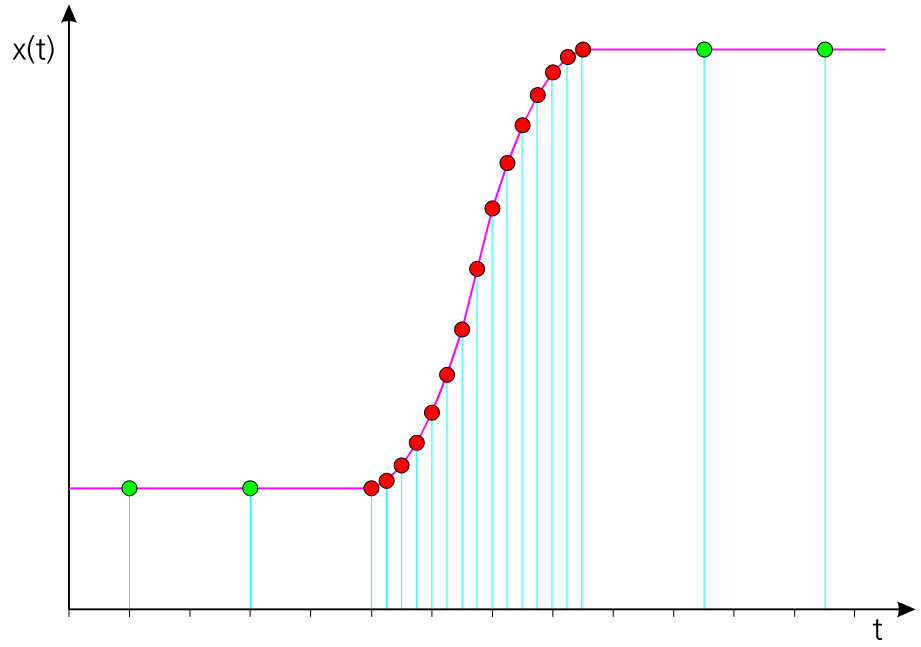

Bild 35 Schrittweitenanpassung

Prinzip: Geben Sie nicht die Schrittweite, sondern direkt den maximalen Fehler vor. Das Integrationsverfahren muß dann die Schrittweite (nur) solange verkleinern, bis die Fehlerschranke unterschritten ist. Dazu integriert es einmal mit nominaler Schrittweite und ein zweites Mal mit halber Nominalschrittweite. Die Differenz der beiden Integrationsergebnisse liefert einen Anhaltspunkt, ob und um wieviel im nächsten Schritt die Schrittweite vergrößert werden darf (wenn der Fehler kleiner war als die geforderte Schranke), oder um welchen Faktor die Schrittweite verkleinert werden muß (weil die Differenz beider Integrationsergebnisse zu groß war).

Die Alternative zum nochmaligen Integrieren mit halber Nominalschrittweite ist das Verwenden zweier Integrationsverfahren mit unterschiedlicher Ordnung und der Vergleich beider Ergebnisse. So lassen sich beispielsweise ein Runge-Kutta vierter Ordnung und ein Runge-Kutta fünfter Ordnung so zu einem Runge-Kutta-Merson-Algorithmus kombinieren, daß insgesamt nur fünf Auswertungen des Differentialgleichungssystems notwendig sind und dabei gleichzeitig eine Schätzung des Fehlers möglich ist.

11 Stabilität

Ein System ist entweder stabil (beschränkte Systemantwort bei beschränkter Anregung) oder instabil (unbeschränkte Systemantwort bei beschränkter Anregung). Leider aber wird ein System nicht von jedem Integrationsverfahren entsprechend seiner wahren Stabilitätseigenschaften simuliert. Abhängig vom Integrationsverfahren und der gewählten Integrationsschrittweite kann es sowohl passieren, daß ein stabiles System instabil simuliert wird, als auch, daß ein instabiles System stabil simuliert wird. Im Regelfall treten diese Stabilitätsprobleme immer dann auf, wenn die Schrittweite zu groß im Verhältnis zur Eigendynamik des Systems gewählt wird.

Jedes Integrationsverfahren hat einen Stabilitätsbereich, in dem es lineare Systeme stabil simuliert.

Beispiel: Lineares System erster Ordnung (P-T_1) und Euler:

Übertragungsfunktion (stabil für $T > 0$): $F(s) = \frac{X}{U} = \frac{1}{Ts+1}$

Pol: $s_1 = -\frac{1}{T}$

Differentialgleichung: $T\dot{x} + x = u$

$$\dot{x} = -\frac{1}{T}x + \frac{1}{T}u$$

Simuliertes System nach Euler:

$$\begin{aligned} x(t+\Delta t) &= x(t) + f\big(x(t),u(t),t\big)\Delta t \\ &= x(t) + \dot{x}(t)\Delta t \\ &= x(t) + \left(-\frac{1}{T}x(t) + \frac{1}{T}u(t)\right)\Delta t \\ &= \left(1-\frac{\Delta t}{T}\right)x(t) + \frac{\Delta t}{T}u(t) \end{aligned}$$

Für die Stabilität eines linearen Systems ist die Eingangsgröße irrelevant:

Homogene Differenzengleichung: $x(t+\Delta t) = \left(1-\frac{\Delta t}{T}\right)x(t)$ ist eine geometrische Folge:

Geometrische Folge in rekursiver Darstellung: $a_{k+1} = q \cdot a_k$

Geometrische Folge in unabhängiger Darstellung: $a_k = a_0 \cdot q^{k-1}$ mit $k \in N$

Stabilitätsbedingung einer geometrischen Folge: $|q| < 1 \Rightarrow -1 < q < 1$

Differenzengleichung in unabhängiger Darstellung:

$$x(t) = x(0)\left(1 - \frac{\Delta t}{T}\right)^{\frac{t}{\Delta t}} \quad \text{mit} \quad t \in \{0, \Delta t, 2\Delta t, \cdots\}$$

Stabil, wenn: $\left|\left(1 - \frac{\Delta t}{T}\right)\right| < 1$

Verallgemeinern ins Komplexe durch Einsetzen des Pols: $\left|(1 + s\Delta t)\right| < 1$

Real- und Imaginärteil des Pols: $s = \sigma + j\omega \quad \Rightarrow \quad \left|(1 + (\sigma + j\omega)\Delta t)\right| < 1$

Betrag bilden und ausquadrieren: $(1 + \sigma\Delta t)^2 + (\omega\Delta t)^2 < 1$

Euler: Kreisgleichung mit Mittelpunkt -1 und Radius 1 in der $s\Delta t$ -Ebene.

Für die anderen Integrationsverfahren ist die Herleitung des Stabilitätsgebietes etwas umständlicher. Es ergeben sich Gebiete, die nur noch numerisch iterativ ermittelt werden können:

Mittelpunktsregel: $\left|\left(1 + s\Delta t + \frac{(s\Delta t)^2}{2}\right)\right| < 1$

Runge-Kutta 4. Ordnung: $\left|\left(1 + s\Delta t + \frac{(s\Delta t)^2}{2} + \frac{(s\Delta t)^3}{6} + \frac{(s\Delta t)^4}{24}\right)\right| < 1$

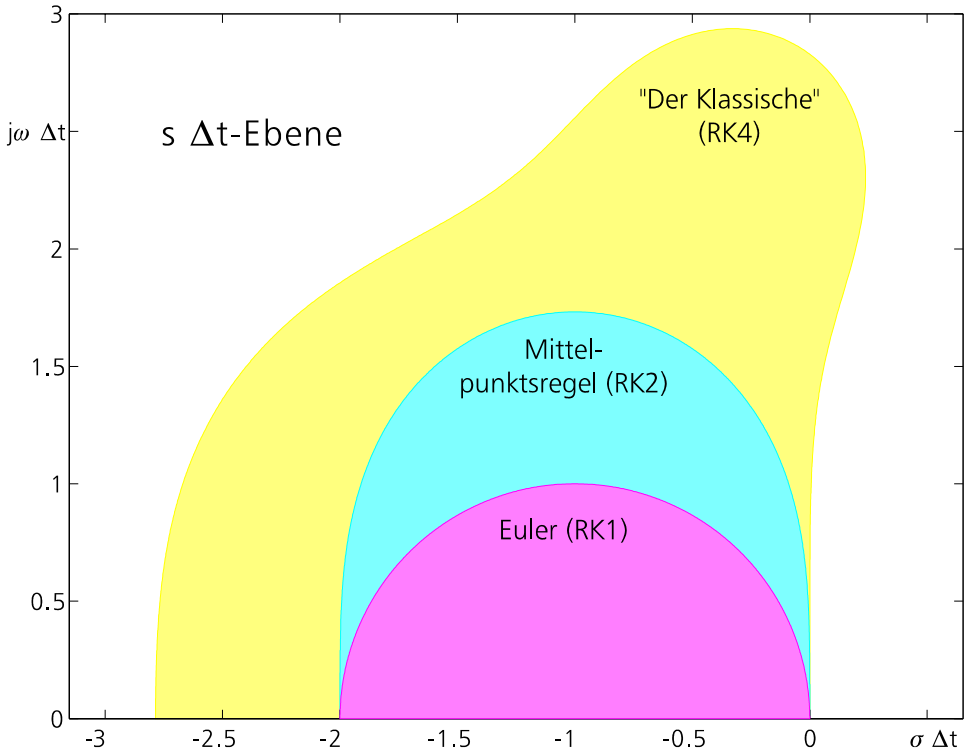

Bild 36 Stabilitätsgebiete von RK1, RK2 und RK4 in der sΔt-Ebene

An dem Bild läßt sich ablesen:

- Je höher die Ordnung des Verfahrens, desto größer das Stabilitätsgebiet.
- Für reelle Pole ($P-T_1$) werden RK1 und RK2 bei der gleichen Schrittweite instabil. RK4 kann mit der gleichen Schrittweite noch etwas schnellere Systeme stabil simulieren.
- RK1 und RK2 können ungedämpfte Schwinger (Pole auf der imaginären Achse) überhaupt nicht korrekt simulieren. Die gesamte imaginäre Achse liegt im instabilen Bereich. Es entsteht eine aufklingende Schwingung.
- RK4 simuliert ungedämpfte Schwinger korrekt grenzstabil, wenn sie langsam sind. Die Pole liegen dann sehr dicht am Ursprung und daher praktisch auf der Grenzkurve des Verfahrens. Wenn die Schwingung so schnell wird, bzw. die Schrittweite so groß, daß die Pole in der “Beule” liegen, wird ein ungedämpfter Schwinger stabil, also gedämpft und damit falsch simuliert. Wenn die Pole des ungedämpften Schwingers das Stabilitätsgebiet nach oben verlassen, weil er für die Schrittweite zu schnell geworden ist, wird das Simulationsergebnis ebenso falsch eine aufklingende Schwingung zeigen.
- Es ist wichtig, daß Sie sich klarmachen, daß innerhalb des Stabilitätsgebietes nur stabil, also abklingend simuliert wird. Dies bedeutet definitiv nicht, daß das Simulationsergebnis im Stabilitätsgebiet korrekt ist. Ganz im Gegenteil wird ein schwach instabiles System, das genau in Runge-Kuttas “Beule” rechts der imaginären Achse liegt, tatsächlich auch stabil simuliert, was eindeutig falsch ist.

Wenn die Integrationsschrittweite größer als die doppelte (kleinste) Systemzeitkonstante wird, simuliert Euler ein stabiles System instabil!

Deshalb: Immer überprüfen, ob nicht die Integrationsschrittweite zu groß und das Integrationsergebnis deshalb falsch ist! Beispielsweise durch nachträgliches Halbieren der Schrittweite und Vergleich beider Simulationsergebnisse. Oder besser gleich Verfahren mit Schrittweitenkontrolle verwenden.

12 Steife Systeme

Ein steifes System ist für alle Verfahren, die nicht speziell dafür entwickelt wurden, ein ernstzunehmender Gegner. Das klassische Paradebeispiel ist ein Gesamtsystem, das aus zwei Teilsystemen mit sehr unterschiedlicher Dynamik besteht:

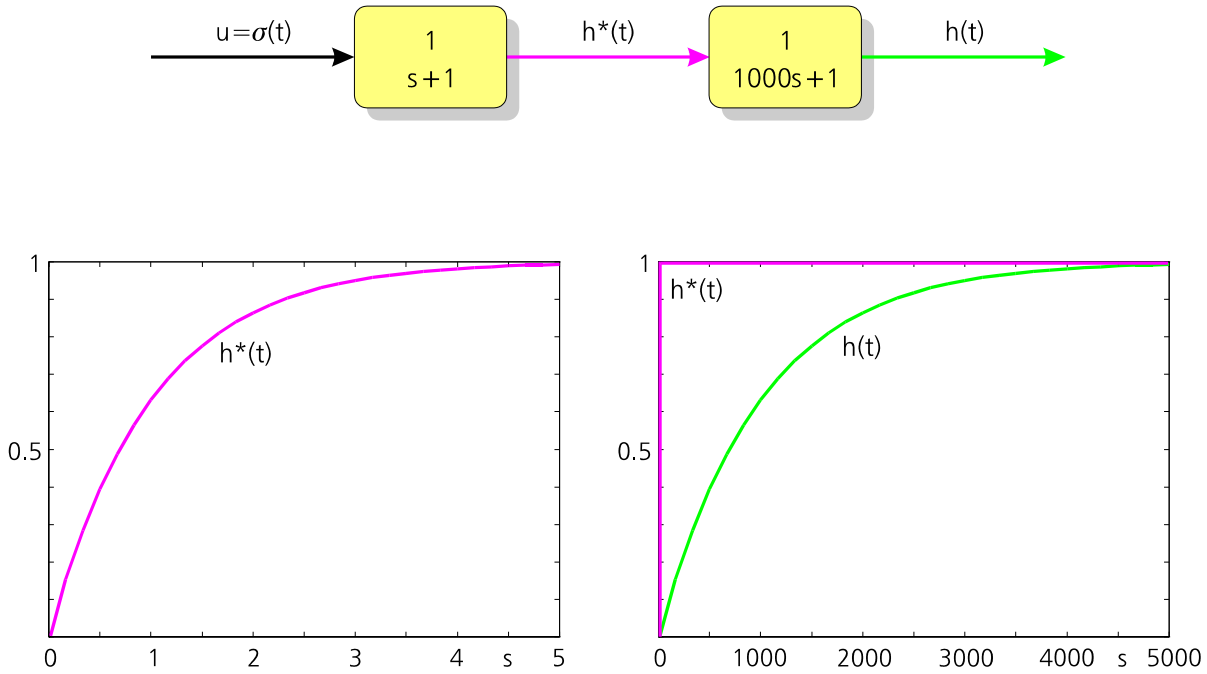

Bild 37 Steifes System aus einem schnellen und einem langsamen P-T_1

Das erste, sehr schnelle System trägt zur Gesamtdynamik praktisch nichts bei, da sein Ausgang dem Eingang unmittelbar (innerhalb weniger Sekunden) folgt. Die Gesamtantwort wird also fast ausschließlich vom zweiten, langsameren System bestimmt (einige tausend Sekunden). Trotzdem muß die Integrationsschrittweite der kleineren Zeitkonstanten des schnelleren Systems Rechnung tragen, damit dieses und folglich die ganze Simulation nicht instabil wird.

Praktisches Beispiel: Langsame, aber komplexe Flugzeugbewegung mit sehr vielen Eigenwerten nahe des Ursprungs. Einige wenige, gut gedämpfte, aber schnelle Stellerpole weit weg vom Ursprung:

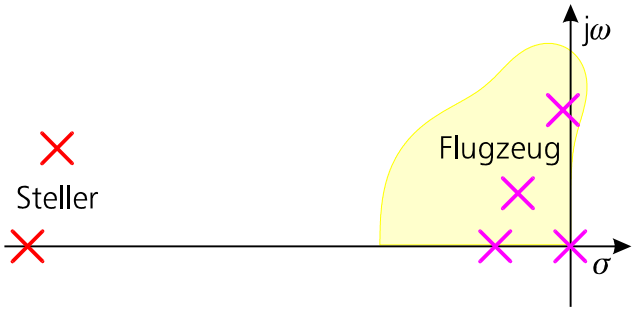

Bild 38 Langsame Flugzeugbewegung, aber schnelle Steller⇒ steifes System

Eine Schrittweite, die nur auf die langsamere Flugzeugbewegung abgestimmt ist, führt zu einer instabilen Simulation der dann außerhalb des Stabilitätsgebietes liegenden schnellen Stellerpole. Da die dann divergierenden Stellerausgänge aber die Eingangsgrößen der Flugzeugbewegung darstellen, wird die ganze Simulation instabil und unbrauchbar.

Lösungsmöglichkeiten:

1. Verkleinern der Schrittweite, bis auch die schnellen Stellerpole noch im Stabilitätsgebiet liegen. Nachteile: Stark erhöhter Rechenaufwand, da die rechenzeitintensive Flugzeugbewegung (viele nichtlineare Differentialgleichungen, Tabellen, ...) aufgrund der sehr kleinen Abtastzeit zu oft unnötigerweise durchgerechnet wird. Das Integral, das beim langsamen System in jedem Schritt zum aktuellen Wert addiert wird, kann so gering sein, daß sich dieser Wert aufgrund der endlichen Mantissenlänge der gewählten Zahlendarstellung nicht oder nur stark fehlerbehaftet ändert. Beispiel: Flugzeug in Höhe 10000 m, Mantissenlänge bei vier Byte ungefähr sechs Stellen. Wenn die Schrittweite nun wegen der schnellen Steller so gering ist, daß sich die Höhe in einem Schritt nur noch um 1 cm ändert, kann 10000.01m mit sechs signifikanten Stellen nicht mehr aufgelöst werden.

2. Schnelle und langsame Systemteile mit unterschiedlichen Schrittweiten integrieren. Hier: Flugzeugbewegung mit großer und Steller mit kleiner Schrittweite rechnen. Nachteile: Nur wenige Simulationsumgebungen lassen unterschiedliche Schrittweiten für unterschiedlich schnelle Subsysteme zu. Wenn das schnelle Subsystem Zustände oder Ausgangsgrößen des langsamen Teilsystems benötigt, müssen diese fehlerträchtig extrapoliert werden.

3. Verwendung spezieller impliziter Verfahren (Gear, ...), die einen Stabilitätsbereich besitzen, der für lineare System die gesamte linke $s\Delta t$-Ebene umfaßt. Nachteile: Stabilität kann bei nichtlinearen Systemen nicht garantiert werden. Ziemlich komplizierte Algorithmen, wenn von höherer Ordnung und mit vernünftiger Schrittweitensteuerung. Empfehlenswert, wenn als “black-box-Routine” vorhanden.

13 Begrenzungen

Sie wollen einen Steller (Aktor) mit Positionsbegrenzung, Geschwindigkeitsbegrenzung und Beschleunigungsbegrenzung simulieren. Ein realer Steller ist nun mal mechanisch in seinem Ausschlag limitiert, seine Rate wird durch die zugeführte Energie (Strom, Hydrauliköldurchfluß, ...) begrenzt und Massen wollen auch erst mal beschleunigt werden.

Ihr erster naiver Modellierungs- und Simulationsansatz eines positionsbegrenzten Stellers ist ein System erster Ordnung, an dessen Ausgang Sie einen Begrenzer anschließen, der nur Signale durchläßt, die kleiner als seine Obergrenze und größer als seine Untergrenze sind.

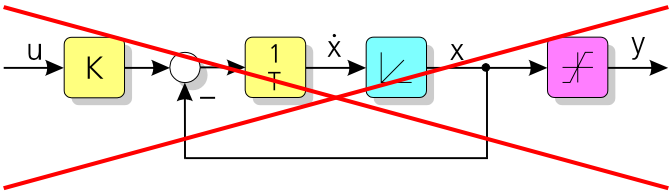

Bild 39 FALSCHE Implementation eines positionsbegrenzten P-T_1

Das Problem mit dieser Implementation besteht nun darin, daß, wenn der Zustand x größer als die Grenze geworden ist und der Begrenzer das Ausgangssignal y begrenzt, der Integrator nicht anhält, sondern so weiter integriert ("volläuft"), als wäre der Begrenzer nicht vorhanden.

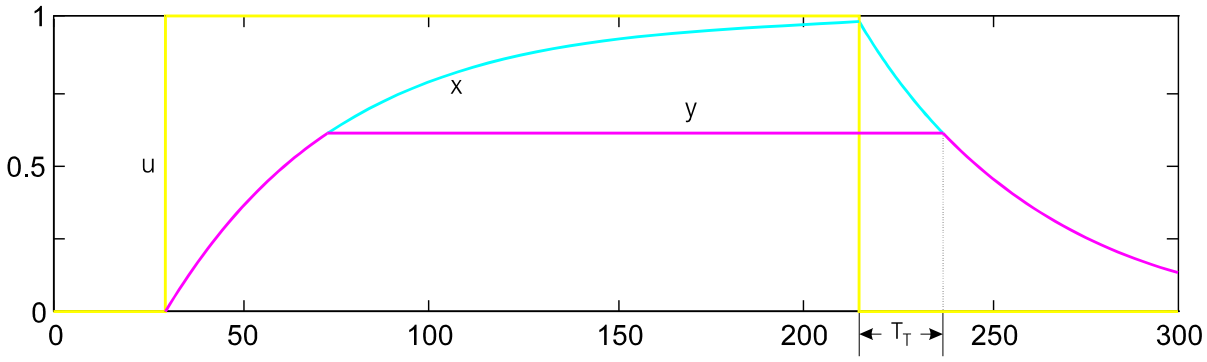

Bild 40 Totzeit zwischen Rücknahme des Kommandos und Reaktion des Systems

Wenn dann das Kommando zurückgenommen wird, so daß auch die Ausgangsgröße sofort wieder anfangen sollte, zurückzugehen, vergeht erst eine Totzeit T_T, während der der Integrator als Tiefpaß "leerläuft". Erst, wenn der Zustand x kleiner als der Grenzwert geworden ist, folgt auch der Ausgang dem Zustand. Diese Totzeit existiert bei einem realen Steller natürlich nicht. Wenn das Kommando an einen gesättigten, realen Steller auch nur etwas geringer als dessen Begrenzung wird, beginnt dieser sofort, dem Kommando zu folgen.

Da Sie das “Vollaufen” des Integrators als das Problem erkannt haben, können Sie dieses lösen, indem Sie den Integrator anhalten, sowie der Begrenzer anspricht. Simulationstechnisch müssen Sie dazu den Eingang des Integrators auf Null setzen und halten, damit er nicht mehr weiterintegriert und seinen Zustand (Ausgang) einmalig auf den Begrenzungswert zurücksetzen.

Jetzt brauchen Sie aber noch eine Bedingung dafür, daß der Integrator aus seinem Haltezustand erweckt wird und wieder normal integrieren darf, wenn das Kommando unter die Grenze sinkt. Sie überwachen dazu den Eingang des Integrators (die Zustandsableitung, die rechte Seite der Differentialgleichungen) und überprüfen, ob diese Ableitung den Integrator noch weiter in die Begrenzung “hineintreiben” würde, oder ob die Ableitung im nächsten Integrationsschritt dazu führen würde, daß der Integrator die Sättigung verließe. Wenn der Integrator also in seiner positiven Begrenzung (Maximalwert) angehalten wurde und die aus dem Kommando resultierende Ableitung positiv ist, muß er weiter in der Sättigung verharren und Sie müssen die Ableitung erneut explizit auf Null setzen. Wenn hingegen die Ableitung bei einer positiven Begrenzung negativ ist, also aus der Begrenzung “herauszeigt”, erlösen Sie den Integrator aus seinem Haltezustand und lassen die negative Ableitung zu, die den Integrator schon im nächsten Integrationsschritt aus der Begrenzung hinausführt.

Programmtechnisch müssen Sie, je nach Ihrer Lieblingssprache, etwas in der folgenden Form codieren:

```
! Wenn Zustand Begrenzung überschritten hat,
! Zustand begrenzen und Ableitung berechnen.
! Wenn zusätzlich Ableitung in Begrenzung hineinzeigt,
! Ableitung auf Null setzen.

IF x >= x_max
   x  = x_max
   x_punkt = (k*u - x)/T
   IF x_punkt > 0
      x_punkt = 0
   END
ELSE IF x <= x_min
   x = x_min
   x_punkt = (k*u - x)/T
   IF x_punkt < 0
      x_punkt = 0
   END
ELSE
   x_punkt = (k*u - x)/T
END

! Ausgang berechnen

y = x
```

Viele Simulationsumgebungen stellen Ihnen freundlicherweise sowohl einen Integrator zur Verfügung, dessen Zustand Sie während der Simulation setzen können, als auch möglicherweise einen fertigen Integrator mit Begrenzungen, der den oben dargestellten Algorithmus intern umsetzt.

Die Begrenzung der Geschwindigkeit eines Tiefpasses erster Ordnung ist im Vergleich zur Positionsbegrenzung recht einfach: Ein Begrenzerblock direkt vor dem Eingang des Integrators begrenzt ohne weitere Rückwirkungen die Ableitung der Position, also die Geschwindigkeit und führt im “rate limit”-Fall dazu, daß die Position nur noch linear in Form einer Gerade anwächst.

13.1 P-T_2

Wenn nun schon die simulationstechnische Umsetzung der Positionsbegrenzung eines P-T_1 nicht ganz trivial ist, so sei an dieser Stelle die Vermutung erlaubt, daß Sie im rauhen Simulationsalltag wohl nur recht selten wirklich korrekte Simulationsmodelle eines positions-, geschwindigkeits- und beschleunigungsbegrenzten Systems zweiter Ordnung finden werden.

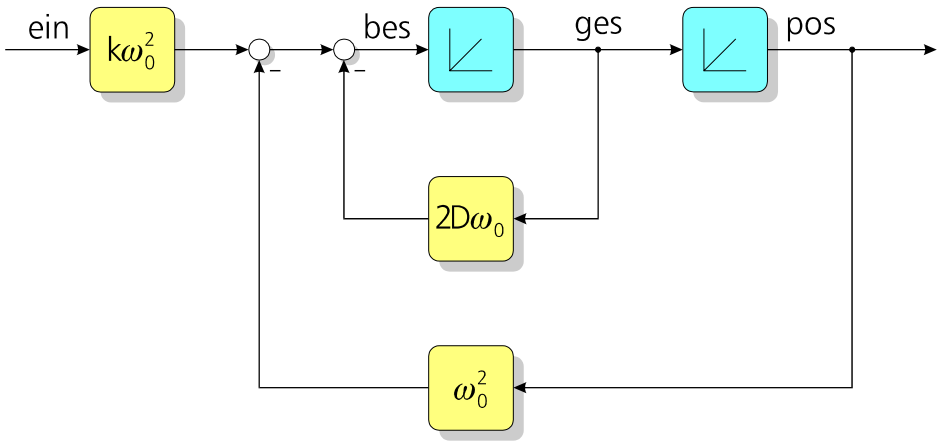

Bild 41 Position, Geschwindigkeit und Beschleunigung eines Stellers (P-T_2)

Bedenken Sie, daß Sie jetzt, wenn der Steller an seinen Anschlag rauscht, sowohl den Positionsintegrator auf seinen Grenzwert, den Geschwindigkeitsintegrator auf Null, als auch die Beschleunigung auf Null setzen müssen. Wenn der Steller mechanisch am Anschlag steht, besitzt er eben weder Geschwindigkeit noch Beschleunigung. Zur Entscheidung der Frage, ob der Steller weiter in der Sättigung bleibt, oder ob er aus seiner Begrenzung herausintegrieren soll, muß jetzt aber sowohl die Geschwindigkeit am Eingang des Positionsintegrators als auch der Beschleunigungswert am Eingang des Geschwindigkeitsintegrators überprüft werden.

Eine korrekte programmtechnische Umsetzung aller Begrenzungen im System zweiter Ordnung sieht folglich beispielsweise so aus:

```
! Beschleunigung berechnen

bes = -2.0*d*om*ges + om*om*(-pos + k*ein)


! Wenn Position Begrenzung überschritten hat,
! Position begrenzen.
! Wenn zusätzlich Geschwindigkeit in Begrenzung hineinzeigt,
! Geschwindigkeit auf Null setzen.
! Wenn zusätzlich Beschleunigung in Begrenzung hineinzeigt,
! Beschleunigung auf Null setzen.

IF pos >= pos_max
   pos  = pos_max
   IF ges >= 0
      ges  = 0
      bes  = om*om*(-pos + k*ein)
      IF bes > 0
         bes = 0
      END
   END
ELSE IF pos <= pos_min
   pos  = pos_min
   IF ges <= 0
      ges  = 0
      bes  = om*om*(-pos + k*ein)
      IF bes < 0
         bes = 0
      END
   END
END

! Wenn Geschwindigkeit Begrenzung überschritten hat,
! Geschwindigkeit begrenzen.
! Wenn zusätzlich Beschleunigung in Begrenzung hineinzeigt,
! Beschleunigung auf Null setzen.

IF ges >= ges_max
   ges  = ges_max
   bes  = -2.0*d*om*ges + om*om*(-pos + k*ein)
   IF bes > 0
      bes = 0
   END
ELSE IF ges <= ges_min
   ges = ges_min
   bes = -2.0*d*om*ges + om*om*(-pos + k*ein)
   IF bes < 0
      bes = 0
   END
END
```

```
! Wenn Beschleunigung Begrenzung überschritten hat,
! Beschleunigung begrenzen.

IF bes >= bes_max
   bes  = bes_max
ELSE IF bes <= bes_min
   bes = bes_min
END
```

Wenn Sie alles richtig programmiert haben, müßte Ihr begrenztes System zweiter Ordnung etwa so reagieren:

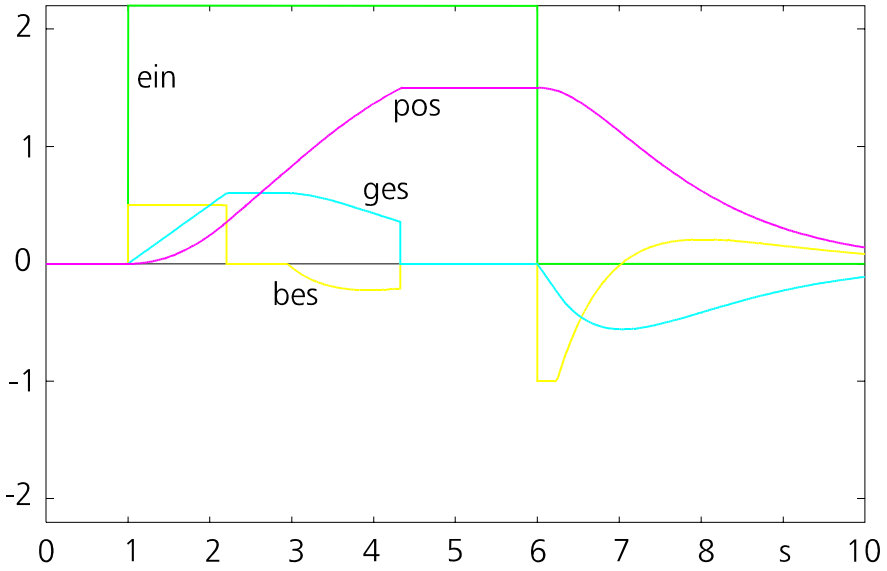

Bild 42 Begrenzungen in Beschleunigung, Geschwindigkeit und Position

Dargestellt ist die Reaktion des begrenzten Schwingers auf einen Blockeingang der Höhe 2 von 1.0 s bis 6.0 s. Zum Zeitpunkt 1.0 s springt die Beschleunigung auf ihre Begrenzung von 0.5. Die Geschwindigkeit steigt als Folge linear an, während die Position quadratisch, also im Gegensatz zum Tiefpaß erster Ordnung mit waagerechter Anfangstangente, wächst. Bei 2.2 s erreicht die Geschwindigkeit ihr Maximum von 0.6 und wird auf diesen Wert begrenzt. Entsprechend fällt die Beschleunigung auf Null und die Position steigt nur noch linear an. Zum Zeitpunkt 3.0 s ist die Position schon so nahe an ihren Sollwert herangekommen, daß über die interne Rückführung begonnen wird abzubremsen. Die Beschleunigung wird also negativ und die Geschwindigkeit sinkt, bis bei 4.3 s die Position ihren Grenzwert von 1.5 erreicht. In der nun folgenden Phase konstanter Position müssen sowohl die Geschwindigkeit als auch die Beschleunigung vom Algorithmus explizit auf Null gesetzt werden. Erst, wenn zum Zeitpunkt 6.0 das Kommando auf Null zurückgenommen wird, reagiert das System mit einer auf ein Minimum von -1.0 begrenzten Beschleunigung, die dann ziemlich schnell verlassen wird. Ab dann entspricht die Reaktion der eines unbegrenzten Schwingers zweiter Ordnung.

14 Literatur

Press, W. H. et al.	*Numerical Recipes*, Cambridge University Press, Cambridge, 1986 “Das Buch!” (Anmerkung des Autors)
Schmidt, G.	*Simulationstechnik*, R. Oldenbourg Verlag, München, 1980
Unbehauen, H.	*Regelungstechnik I und II*, Vieweg, Braunschweig, 1982
The Mathworks	MATLAB®- und SIMULINK®-Handbuch, 1994
Breitenecker, F. et al.	*Simulation kontinuierlicher Systeme*, CCG-Kurs IT 4.31, Carl-Cranz-Gesellschaft, Oberpfaffenhofen, 1994
Hartley, T. T. et al.	*Digital Simulation of Dynamic Systems*, Prentice Hall, New York, 1994
Matko, D. et al.	*Simulation and Modelling of Continuous Systems*, Prentice Hall, New York, 1992