Grundlagen der Modellierung und Simulation

3.6 Numerische Lösung nichtlinearer Zustands-DGLn
Übersicht Systemtypen

Foliensatz 9

Im Allg. keine explizite, formelmäßige, nur numerische Lösung möglich (Kap. 3.6).

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

System im Allgemeinen nichtlinear und zeitvariant (und explizit)

f, g hängen nicht explizit von t ab

f, g hängen linear von x, u ab

$$\dot{x} = f(x,u)$$
$$y = g(x,u)$$

System zeitinvariant (autonom) und i.Allg. nichtlinear

$$\dot{x} = A(t)\cdot x + B(t)\cdot u$$
$$y = C(t)\cdot x + D(t)\cdot u$$

System linear und i.Allg. zeitvariant

f, g hängen linear von x, u ab

A, B, C, D sind konstant

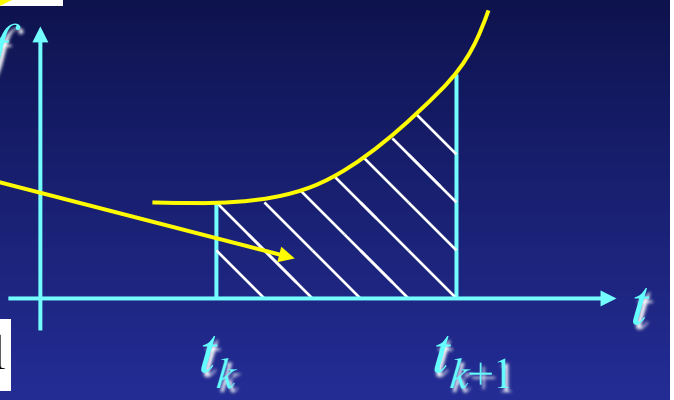
$$\dot{x} = A\cdot x + B\cdot u$$
$$y = C\cdot x + D\cdot u$$

System linear und zeitinvariant

3.6 Numerische Lösung nichtlinearer Zustands-DGLn
Grundidee numerische Integration

Ausgangsgleichung: $x(t_{k+1}) = x(t_k) + \int_{t_k}^{t_{k+1}} f(x(\tau))\,\mathrm{d}\tau$

Integralterm entspricht Fläche

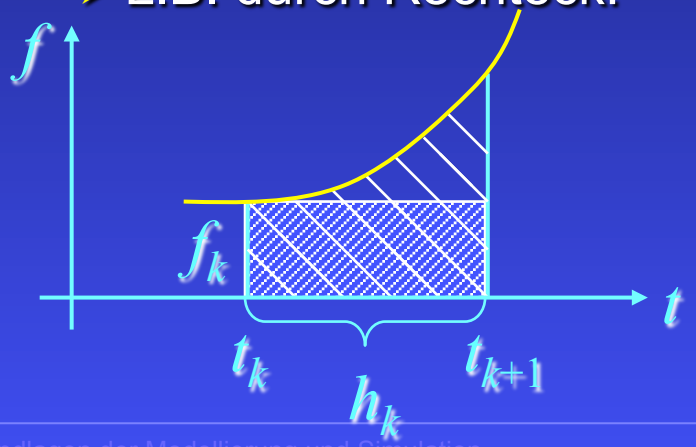


Ansatz für numerisches Integrationsverfahren:

Bezeichnung: $x_k \approx x(t_k), \quad k = 1,\ldots,n_t+1$

Approximation der Fläche

- z.B. durch Rechteck:



$$x(t_{k+1}) = x(t_k) + h_k \cdot f_k, \quad f_k := f(x(t_k))$$

- andererseits auch als Approximation der Ableitung interpretierbar:

$$\frac{x(t_{k+1}) - x(t_k)}{h_k} \approx \dot{x}(t_k) = f(x(t_k))$$

Diese Approximation nennt man (explizites) Euler-Verfahren!

3.6 Numerische Lösung nichtlinearer Zustands-DGLn
Grundlegende Verfahren

Zunächst Betrachtung für $n=1$ (skalare Zustandsdifferentialgleichung)

Gegeben: $x_k \approx x(t_k)$ Näherungslösung am Zeitpunkt t_k

Gesucht: $x_{k+1} \approx x(t_{k+1})$ Näherungslösung am Zeitpunkt t_{k+1}

3.6.2 Einschrittverfahren (one step methods)

Allgemeiner Ansatz:

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

Mit einer je nach Verfahren unterschiedlichen Funktion Φ

Explizites Euler-Verfahren: $\Phi = f(x_k)$

Implizites Euler-Verfahren: $\Phi = f(x_{k+1})$

3.6 Numerische Lösung nichtlinearer Zustands-DGLn
Übersicht Verfahrenstypen

Foliensatz 9

Numerische Integrationsverfahren unterscheiden sich in der Art der **Approximation** der **Fläche** $\int f$ und/oder des **Gradienten** $x'(t)$:

- **Einschrittverfahren**
- **Mehrschrittverfahren**
- **Extrapolationsverfahren**

In dieser Vorlesung werden nur Einschritt-Verfahren diskutiert.

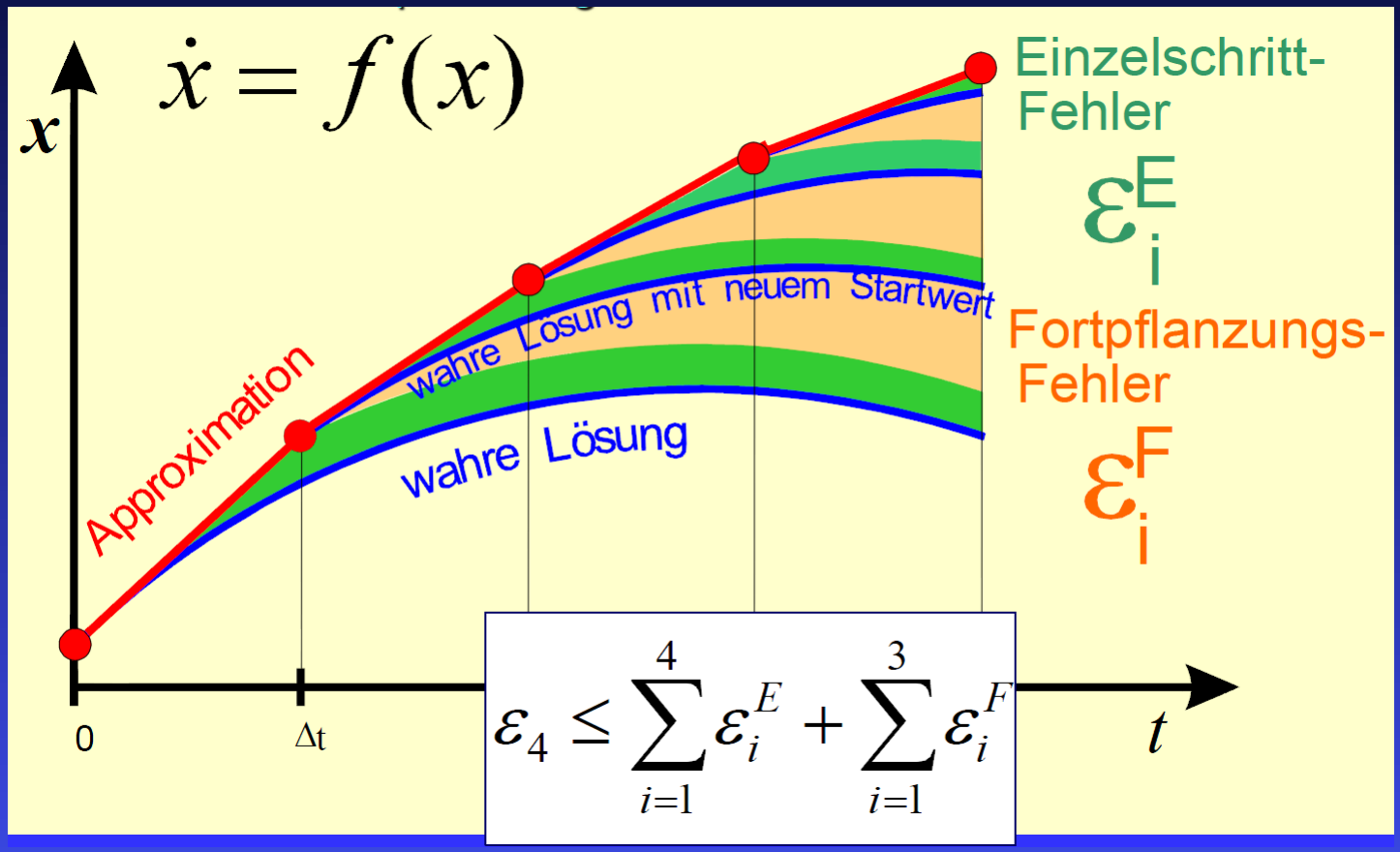
Diese Verfahrensklassen können jeweils in

- **explizite** und
- **implizite**

Verfahren unterteilt werden.

3.6 Numerische Lösung nichtlinearer Zustands-DGLn
Einzelschritt- und Fortpflanzungsfehler

Foliensatz 9



3.6 Numerische Lösung nichtlinearer Zustands-DGLn
Heun-Verfahren

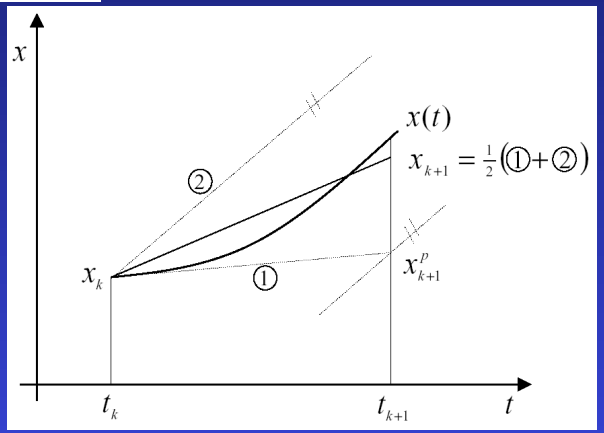
- Ist Beispiel für ein **Prädiktor-Korrektor Verfahren**
- **Prädiktor**-Schritt: $x_{k+1}^{\mathrm{p}} = x_k + h_k \cdot f(x_k)$ (explizites Euler-Verfahren)
- **Korrektor**-Schritt: Mittelung des Gradienten

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

- **Insgesamt:**

Heun Verfahren ist ein 2-stufiges Einschrittverfahren

$$s_1 = f(x_k)$$
$$s_2 = f(x_k + h_k \cdot s_1)$$
$$x_{k+1} = x_k + \frac{h_k}{2}(s_1 + s_2)$$

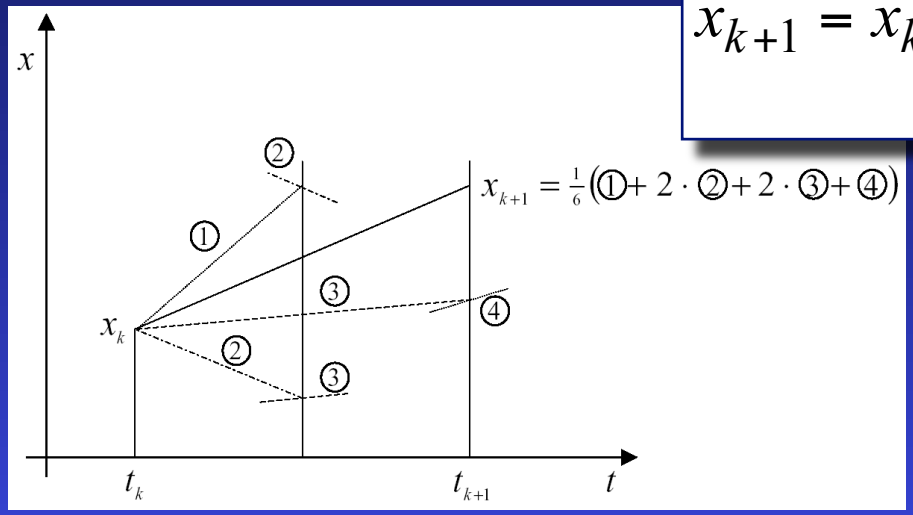


- **Rechenaufwand:** zwei Funktionsauswertungen von f
- **Genauigkeit:** Approximationsfehler $O(h^2)$
  Heun-Verfahren ist Verfahren 2. Ordnung.

3.6 Numerische Lösung nichtlinearer Zustands-DGLn
Runge-Kutta Verfahren 4. Ordnung

- Eine Möglichkeit: Das „klassische“ RK-Verfahren 4. Ordnung

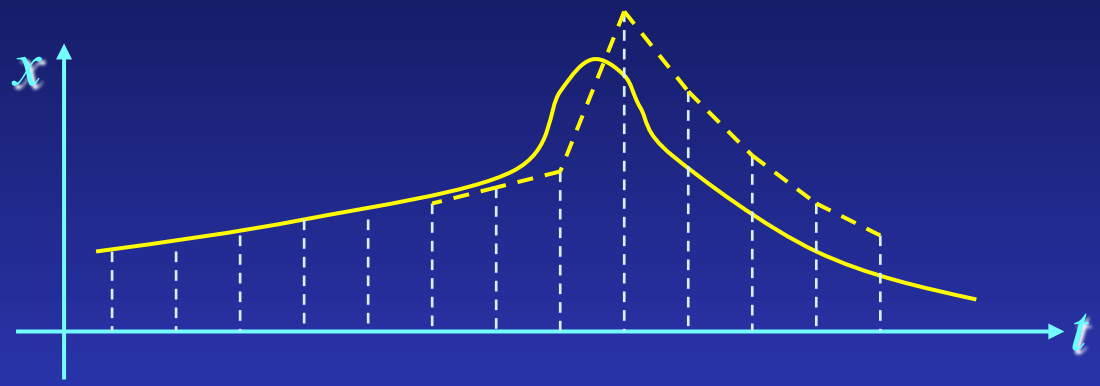
$$
\begin{aligned}
s_1 &= f(x_k) \\
s_2 &= f(x_k + h/2 \cdot s_1) \\
s_3 &= f(x_k + h/2 \cdot s_2) \\
s_4 &= f(x_k + h \cdot s_3) \\
x_{k+1} &= x_k + h/6 \cdot \underbrace{(s_1 + 2s_2 + 2s_3 + s_4)}_{\Phi}
\end{aligned}
$$



- **Rechenaufwand:** vier Funktionsauswertungen von f
- **Genauigkeit:** Approximationsfehler $O(h^4)$
  (falls rechte Seite f 4-mal stetig differenzierbar ist innerhalb von $]t_k, t_{k+1}[$)

3.6 Numerische Lösung nichtlinearer Zustands-DGLn
Schrittweitensteuerung

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



- Abhilfe mit Schrittweitensteuerung:
  - Wahl von h_k so „groß wie möglich“ und „so klein wie nötig“
  - Wahl von h_k, so dass lokaler Approximationsfehler nach einem Schritt unterhalb einer Fehlerschranke (die vom Benutzer vorgegeben wird)

Grundlagen der Modellierung und Simulation

3.7 Integration von Zustands-DGLn mit Unstetigkeiten
Einleitung

Ausgangsproblem: $\dot{x} = f(x,u)$ bzw. $\dot{x} = f(x,u,t)$ mit $x(0) = x_0$

- **Numerische Integrationsverfahren** aus Kap. 3.6 erfordern
  - dass f mindestens so oft **stetig differenzierbar** ist wie Ordnung des Verfahrens (z.B. Euler-Verfahren Ord. 1, „klassisches“ Runge-Kutta-Verfahren Ord. 4);
  - sonst: „**Ordnungsreduktion**“ in der berechneten Lösung, also (möglicherweise massiver) **Genauigkeitsverlust**
- **Aber** (vgl. Kap. 3.3.10):
  - Die physikalische und technische Welt sind nicht nur stetig differenzierbar.
  - Es gibt **kontinuierliche und diskrete**, zeitveränderliche Phänomene!
- **Annahme:**
  - $f(x,t)$ abschnittsweise mehrfach stetig differenzierbar.
  - An Übergängen der Abschnitte ist f möglicherweise unstetig / stetig aber nicht differenzierbar / etc.
  - Übergänge werden durch formulierbare **Ereignisse (events)** ausgelöst, die **zustandsabhängig** oder **zeitgesteuert** sein können.

3.7 Integration von Zustands-DGLn mit Unstetigkeiten
Schaltfunktionen

Die (Um-)Schaltzeitpunkte (events) $t_{s,i}, \quad i = 1,\ldots,n_s$ werden i.Allg. als (einfache) Nullstellen n_q reellwertiger Schaltfunktionen

$$q_l(\boldsymbol{x}(t_{s,i}), t_{s,i}) = 0, \quad l \in \{1,\ldots,n_q\}$$

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

- wenn zwischen der letzten berechneten Näherung $\boldsymbol{x}_k \approx \boldsymbol{x}(t_k)$ und der neuen Näherung $\boldsymbol{x}_{k+1} \approx \boldsymbol{x}(t_{k+1})$ ein Vorzeichenwechsel in (mind.) einer der Schaltfunktionen stattfindet,
- muss der dazwischen liegende erste Schaltzeitpunkt bestimmt werden (und zwar in der durch die Genauigkeitsanforderung des Benutzers und der Ordnung des Verfahrens implizit gegebenen Genauigkeit):

  → Kombination von numerischer Integration und (eindim.) Nullstellensuche
- Voraussetzung: Schrittweite h_k klein genug, so dass kein doppelter Vorzeichenwechsel in derselben Schaltfunktion stattfindet.

Grundlagen der Modellierung und Simulation

3.8 Integration steifer Zustands-DGLn
Beispiel

Annahme: $|\lambda_2| >> |\lambda_1|$ zum Beispiel: $\lambda_1 = -1, \quad \lambda_2 = -1000$

Dann ist $e^{\lambda_2 t} << e^{\lambda_1 t}$ und liefert keinen nennenswerten Beitrag zum Ergebnis.

Wegen der Stabilitätsbedingung $|1 + h\lambda_2| < 1 \quad \Leftrightarrow \quad h < \frac{2}{|\lambda_2|} = 0.002$ muss h sehr klein gewählt werden, obwohl $e^{\lambda_2 t} = e^{-1000\, t}$ zur Lösung praktisch nichts beiträgt.

- Ein solches, **steifes** Verhalten ist allgemein zu erwarten, wenn für $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ die Jacobi-Matrix $\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{x}}$ Eigenwerte λ_i mit Realteil << 0 besitzt.
- (zustandsabhängiges) **Steifigkeitsmaß / Steifigkeitskoeffizient**: $\frac{\max\limits_i |\,\mathrm{Re}(\lambda_i)\,|}{\min\limits_k |\,\mathrm{Re}(\lambda_k)\,|}$
- Zur Berechnung sind **implizite Verfahren** (mit geeignetem Stabilitätsgebiet) geeignet.

3.8 Integration steifer Zustands-DGLn
Kriterien zur Auswahl des Integrators

Zur **Auswahl** und zur **Bewertung** von numerischen Integrationsverfahren betrachtet man i.Allg.

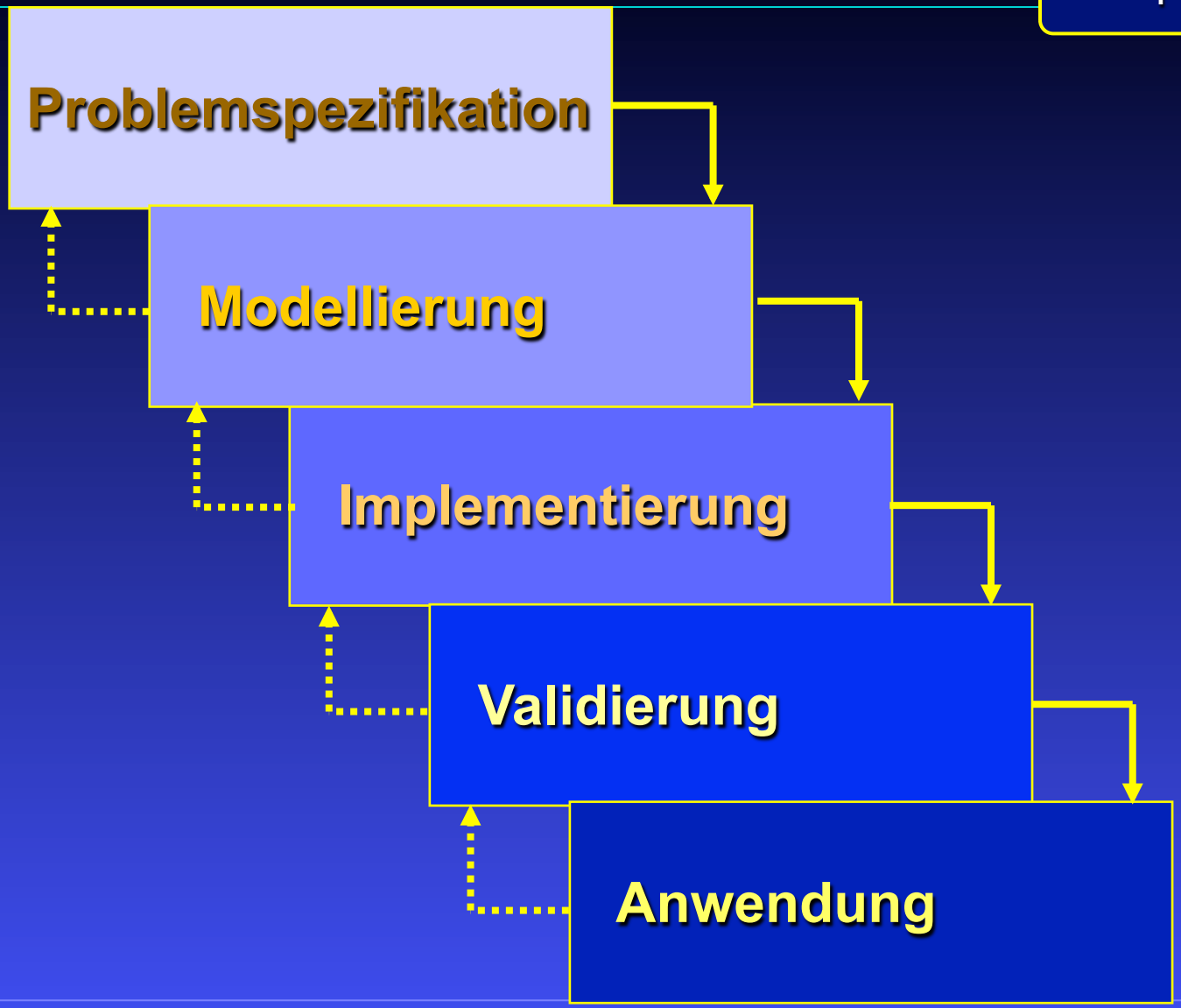
- **Rechenaufwand** (Berechnungseffizienz)
- **Genauigkeit** (Approximationsfehler)
- **Eignung für steife Systeme** (Stabilität)
- **Implementierungsaufwand** (Eigenprogrammierung oder Verwendung einer Bibliothek)

Foliensatz 11

4. Teilschritte einer Simulationsstudie

4. Teilschritte einer Simulationsstudie
Übersicht

Foliensatz 11



Klassifikation zeitkontinuierlicher Simulationswerkzeuge

Foliensatz 12

Level 3

Multidisziplinäre Modellgenerierung (Modelica, VHDL-A)

Level 2

a) Graphische Modellierung (SIMULINK, WorkingModel, Aspen, STELLA,...)
b) Spezialsimulatoren (ADAMS, SIMPACK, KSIM,...)

Level 1

a) Simulationssprachen (ACSL, VHDL, Dare-P, Desire,...)
b) Simulationsframeworks (SIMULINK, C++-Klassenbibliothek)

Level 0

Direkte Programmierung (FORTRAN, C, Pascal, MATLAB,...)

Level 0: Direkte Programmierung mit MATLAB

Beispiel: Euler-Verfahren für Schiffschaukel

(nach W. Wiechert, Uni Siegen)

Modellparameter

```
d = 100.0;      % Reibungskonstante
m = 100.0;      % Schaukelmasse incl. Mensch
l =   2.5;      % Abstand Schwerpunkt zu Drehachse
g =  10.0;      % Erdbeschleunigung
```

Simulationsparameter

```
deltat =  0.1;  % Schrittweite
tEnd   = 20.0;  % Endzeit
```

Startwerte

```
t     = 0.0;    % Startzeit
phi   = 1.0;    % Startwinkel
omega = 0.0;    % Startwinkelgeschwindigkeit
```

Zeitschleife

```
while t<=tEnd
```

Ausgabe

```
    % Ausgabe der Werte
    disp(sprintf('%8.4f   %8.4f   %8.4f',t,phi,omega));
```

rechte Seite

```
    % Berechnung der rechten ODE-Seite
    dphi_dt  =           omega              ;
    domega_dt=-d/(m*l^2)*omega-g/l*sin(phi);
```

Euler-Schritt

```
    % Neue Zeit und Werte
    t    =t    +deltat           ;
    phi  =phi  +deltat*dphi_dt   ;
    omega=omega+deltat*domega_dt;
end;
```

Level 1: Differentialgleichungslöser in MATLAB

Modellgleichungen als Funktion

```
function dxdt=Schaukel(t,x);

    d = 100.0;       % Reibungskonstante
    m = 100.0;       % Schaukelmasse incl. Mensch
    l =   2.5;       % Abstand Schwerpunkt zu Drehachse
    g =  10.0;       % Erdbeschleunigung
```

Modellparameter

```
    % Sprechende Variablen einführen
    phi   = x(1);
    omega = x(2);
```

Variablen

```
    % Berechnung der rechten ODE-Seite
    dphi_dt  =           omega            ;
    domega_dt=-d/(m*l^2)*omega-g/l*sin(phi);
```

rechte Seite

```
    % Sprechende Variablen ausführen
    dxdt = [ dphi_dt ; domega_dt ];
```

Ergebnis

Aufruf eines Differentialgleichungslösers (Integrators):

```
phi0   = 1.0; % Startwert des Winkels
omega0 = 0.0; % Startwert der Winkelgeschwindigkeit

tEnd  = 20.0; % Simulationsdauer
```

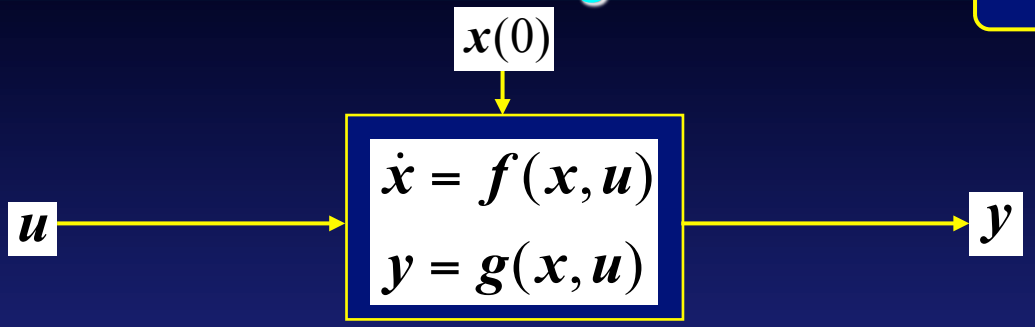
Simulationsparameter

```
% Integrator-Aufruf
[T,X] = ode45 (@Schaukel,[0,tEnd],[phi0;omega0]);
```

Integratoraufruf

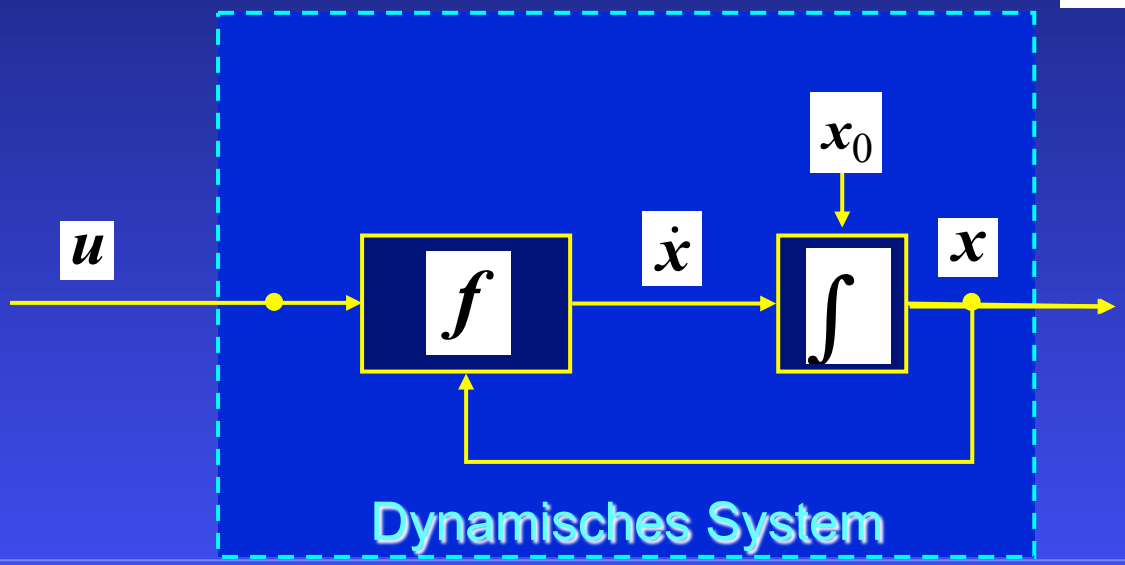
Level 3: Simulink und
Allgemeine blockorientierte Darstellung

Foliensatz 12



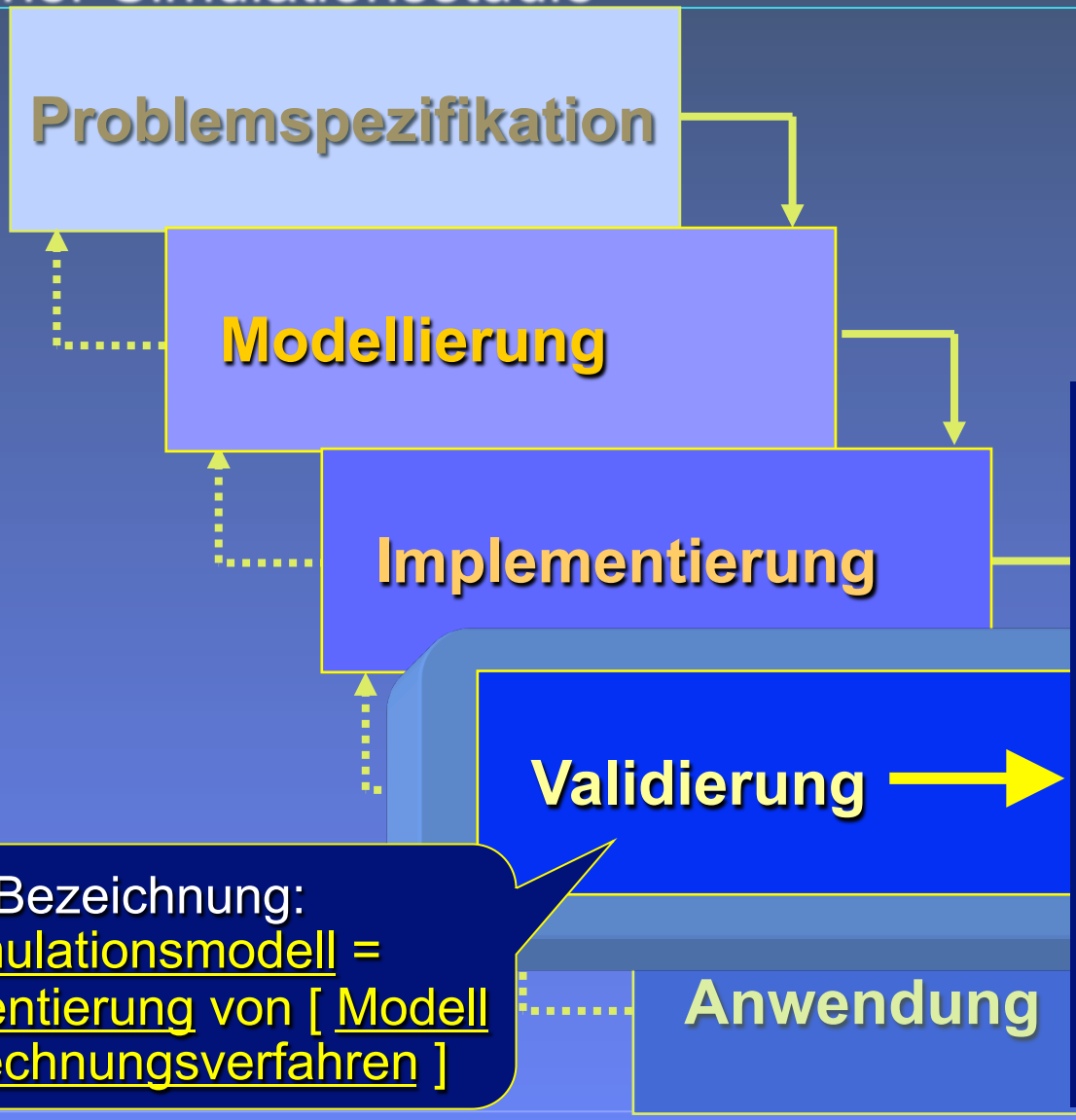
Das DGL-System mit Anfangswert: $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \in \Re^n$

kann äquivalent geschrieben werden als: $\boldsymbol{x}(t) = \boldsymbol{x}_0 + \int_0^t \boldsymbol{f}(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau))\,\mathrm{d}\tau$



5. Interpretation und Validierung

5. Interpretation und Validierung
Schritte einer Simulationsstudie



Foliensatz 12

Systematische Plausibilitäts-überprüfung („Stimmen Modell und Simulation?“):

- Fehlersuche
- Konsistenzprüfung
- Daten-, Parameterabgleich

Bezeichnung:
Simulationsmodell = Implementierung von [Modell + Berechnungsverfahren]

5. Interpretation und Validierung
Begriffe

Foliensatz 12

Verifikation

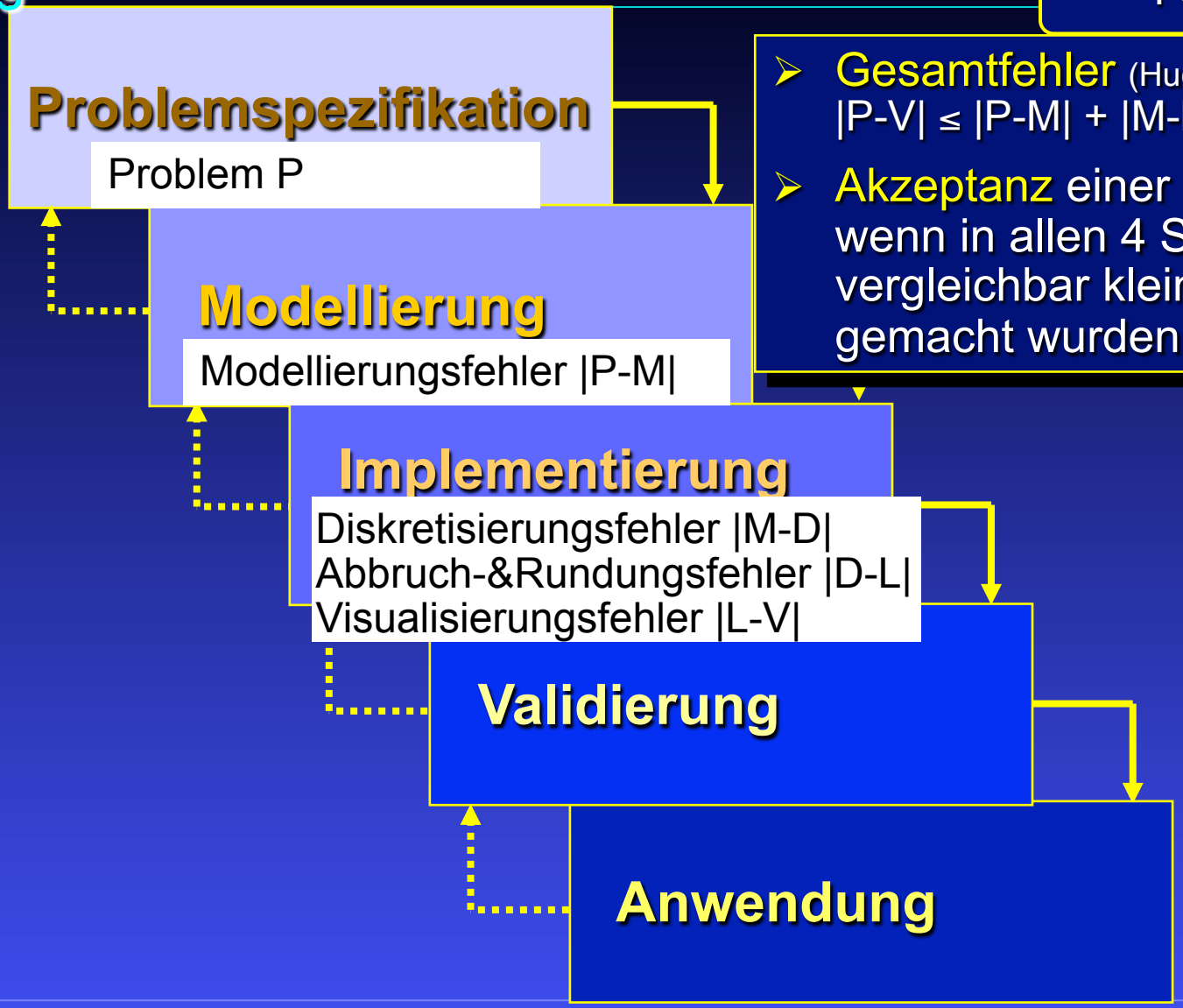
- **Formaler (meist mathematischer) Nachweis** der Korrektheit, dass ein Programm (z.B. Simulationsmodell) einer vorgegebenen Spezifikation entspricht
- Aufgrund der unendlich großen Anzahl von Zustandsverläufen und Störungseinflüssen nichtlinearer dynamischer Systeme ist es i.d.R. **unmöglich**, die **vollständige Korrektheit** eines kontinuierlich dynamischen Simulationsmodells zu beweisen.

Validierung

- **Plausibilitätsüberprüfung**, dass ein Programm (z.B. Simulationsmodell) einer vorgegebenen Spezifikation entspricht
- Ziel der Validierung ist der Nachweis der ausreichenden **Glaubwürdigkeit** des Simulationsmodells im Hinblick auf dessen Einsatzbereich

5. Interpretation und Validierung
Validierung

Foliensatz 12



- Gesamtfehler (Huckle/Schneider, 2002): $|P-V| \leq |P-M| + |M-D| + |D-L| + |L-V|$
- Akzeptanz einer Lösung L, wenn in allen 4 Schritten vergleichbar kleine Fehler gemacht wurden!

Fehlerquellen bei Modellierung und Simulation

Modellierungsfehler

- vereinfachende Modellannahmen (z.B. starrer statt elastischer Körper)
- Ungenauigkeiten in Modellparametern

Approximationsfehler des iterativen Berechnungsverfahrens

(z.B. beim Euler-Verfahren proportional zur Schrittweite)

Rundungsfehler

(Ausführung des Berechnungsverfahrens auf Computer mit endlicher Zahldarstellung)

Programmier-, Implementierungsfehler

5. Interpretation und Validierung
Parameteridentifikation und -kalibrierung

„Tuning“ der Modellparameter anhand von **Messwerten:**

$$\hat{\mathbf{x}}_j = \mathbf{x}(t_j) + \boldsymbol{\varepsilon}_j$$

experimentelle Messwerte für Zustandstrajektorie (mit Messfehler $\boldsymbol{\varepsilon}_j$) $t_j, j = 1,\ldots,n_t$

Optimierungsproblem zur Kalibrierung der Modellparameter

$$\min_{\mathbf{p}\in\Re^{n_p}} \varphi(\mathbf{x}), \quad \varphi(\mathbf{p}) = \frac{1}{2}\sum_{j=1}^{n_t} \omega_j \left\| \hat{\mathbf{x}}_j - \mathbf{x}(t_j;\mathbf{p}) \right\|^2 \in \Re \qquad \omega_j = \text{const.} > 0$$

unter der **Nebenbedingung**, dass $\mathbf{x}(t;\mathbf{p})$

(numerische) Lösung des nichtlinearen Systemmodells

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x},t;\mathbf{p}), \quad \mathbf{x}(0) = \mathbf{x}_0$$