

3. Modellierung und Simulation

3.3. Modellberechnung

3.3.1. Simulation zeitkontinuierlicher Systeme

- Kenngrößen eines Simulationslaufes (Zeitachse)
- Abschnitte eines Simulationslaufes
- Ableitungs- und Integralform von DGL
- Wechselwirkung von Zustandsgröße $Y(t)$ und Ableitung $Y'(t)$
- Zentraler "Integrationskern" (Solver)
- Prinzip der numerischen Integration
- Explizite und implizite Verfahren
- Einschritt- und Mehrschritt-Verfahren
- Explizite Einschrittverfahren (Beispiele)
- Implizite Verfahren (Prinzip)
- Implizite Verfahren (Prädiktor-Korrektor-Verfahren)
- Qualitäten von Integrationsverfahren
- Konfiguration des Solvers

3.3.2. Zeitdiskrete Ereignisse

- Ereignisse im Simulationslauf
- Numerische Definition von Ereignissen
- Klassifizierung numerischer Ereignisse
- Ereignisbehandlung (SimulationX)
- Beispiel: Extremwert-"Sensoren" (SimulationX)

3.3.3. Lineare und nichtlineare Systeme

- Modellcharakter von Systemen
- Dynamische Systeme
- Wann kann man ein System als Linear behandeln?
- Wann muss man ein System als Nichtlinear behandeln?

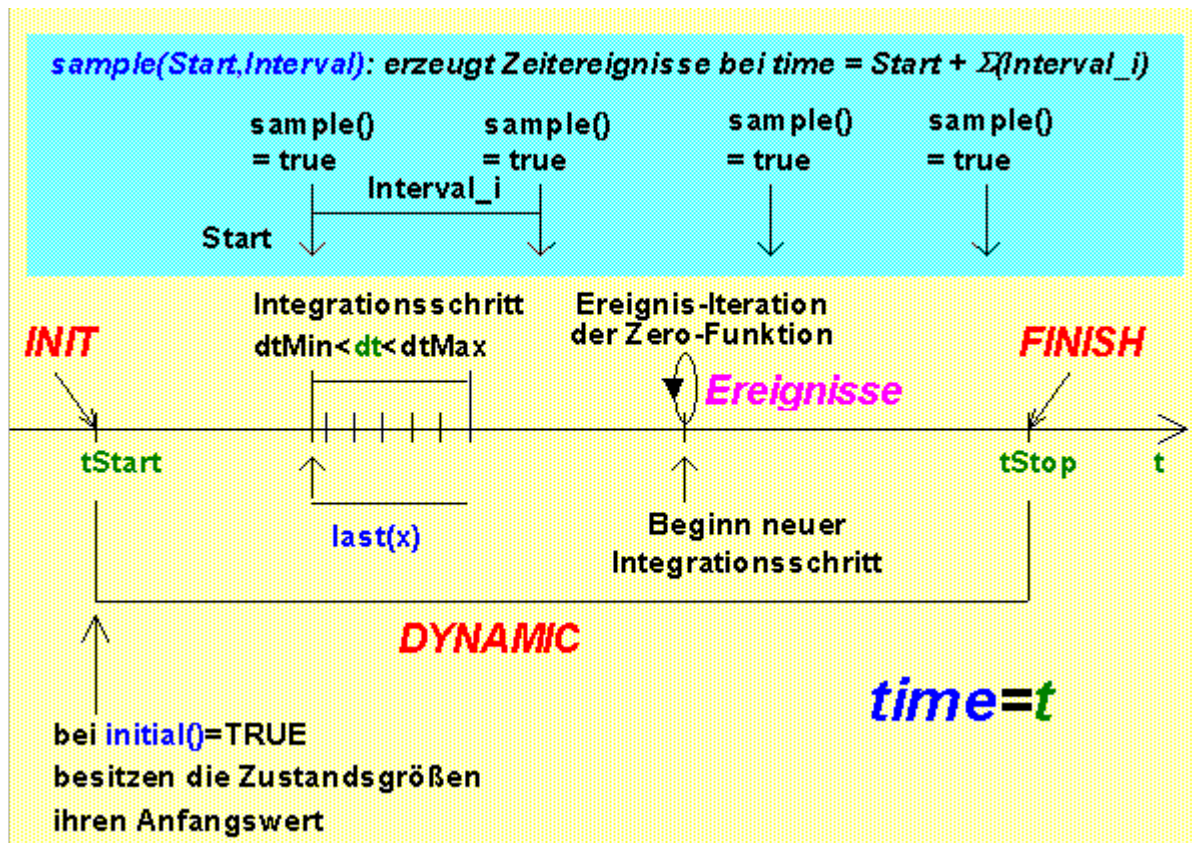


3.3. Modellierung und Simulation: Modellberechnung

3.3.1. Simulation zeitkontinuierlicher Systeme

----> *Kenngrößen eines Simulationslaufes (Zeitachse)*

- Aus Sicht des Modells läuft die Simulationszeit *time* bzw. *t* in diskreten Schritten von *tStart* bis *tStop* (meist vorwärts).
- Die meisten Modelldurchrechnungen sind Hilfsrechnungen (mit "unexakten" Werten). Man kann auf die letzten "richtigen" Modellwerte zugreifen (in SimulationX mit der *last*-Funktion).
- Ein Simulationslauf $tStart \leq time \leq tStop$ besteht aus den Abschnitten **INIT**, **DYNAMIC** und **FINISH**





3.3. Modellierung und Simulation: Modellberechnung

3.3.1. Simulation zeitkontinuierlicher Systeme

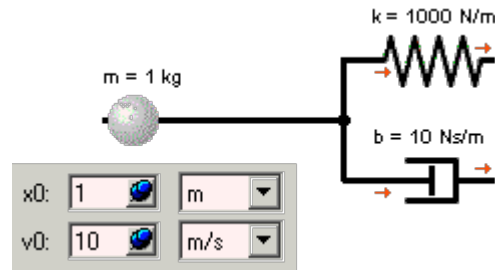
----> *Abschnitte eines Simulationslaufes*

1. INIT (time=tStart)

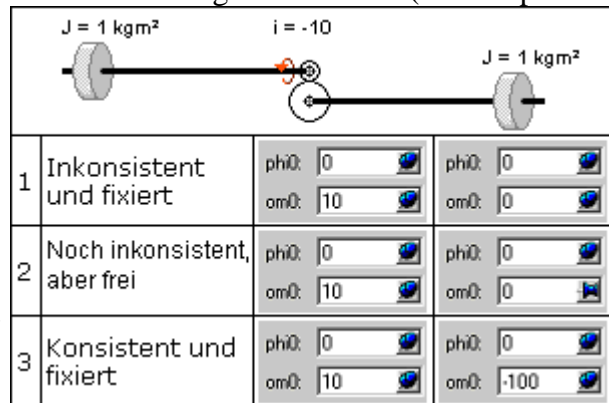
a) *Numerisch:*

Ermittlung konsistenter Anfangswerte für alle Zustandsgrößen (Bilder aus SimulationX-Hilfe)

- Die Anfangswerte charakterisieren den (energetischen) Zustand des Modells zu Beginn der Simulation.
- Anfangswerte gewöhnlicher DGL sind immer konsistent (ODE: **O**rdinary **D**ifferential **E**quation). Der Solver kann daraus eindeutig die Werte der höchsten Ableitungen berechnen (im Beispiel $a(tStart) = -1100 \text{ m/s}^2$).



- Modelle mit Zwangsbedingungen (z.B. für Getriebe) führen zu Algebra-Differenzialgleichungen (DAE: **D**ifferential **A**lgebraic **E**quation). Hier können einige Zustandsgrößen nicht unabhängig voneinander gewählt werden:



Hinweis: Möglichst viele gültige Anfangswerte vorgeben und fixieren!

Freie Anfangswerte nur dort, wo die analytische Berücksichtigung der Abhängigkeiten praktisch unmöglich ist!

b) *Konstruktiv:*

- Berechnung der zeitunabhängigen Parameter für die idealisierten Netzwerkelemente aus Geometrie und Stoffkenngrößen.

2. DYNAMIC (tStart £ time £ tStop)

- Es wirkt die Übertragungsfunktion des dynamischen Systems (=dynamisches Modell).

3. FINISH (time=tStop)

a) *Numerisch:*

- Berechnung integraler Werte (z.B. mittlere Verlustleistung, Statistik, Fourieranalyse)

b) *Konstruktiv:*

- Bewertungsgrößen berechnen für die Qualität des Modellverhaltens bzw. der Modelleigenschaften (z.B. Wirkungsgrad oder Kosten aus Material- und Energieverbrauch)



3.3. Modellierung und Simulation: Modellberechnung

3.3.1. Simulation zeitkontinuierlicher Systeme

----> *Ableitungs- und Integralform von DGL*

- Zweck der Verfahren zur numerischen Integration ist die Lösung von DGL-Systemen (meist im Zeitbereich).
- Dies gilt unabhängig von der Notationsform (z.B. Pendel):

Ableitungsform

$$\ddot{\varphi} + \frac{g}{l} \cdot \sin \varphi = 0$$

Integralform

$$\ddot{\varphi} = -\frac{g}{l} \cdot \sin \varphi$$

$$\dot{\varphi} = \int_{t=t_0}^t \ddot{\varphi} \cdot dt + \dot{\varphi}(t_0)$$

$$\varphi = \int_{t=t_0}^t \dot{\varphi} \cdot dt + \varphi(t_0)$$

- Durch das Simulationssystem erfolgt für den Anwender unsichtbar eine Transformation in die für den Solver geeignete Form eines Differentialgleichungssystems.



3.3. Modellierung und Simulation: Modellberechnung

3.3.1. Simulation zeitkontinuierlicher Systeme

----> Wechselwirkung von Zustandsgröße $Y(t)$ und Ableitung $YP(t)$

- $Y(t)$ als Resultat einer Integral-Bildung kann geometrisch als Fläche unter dem Funktionsverlauf der zugehörigen Ableitung $YP(t)$ interpretiert werden:

$$Y(t) = \int_{t=tStart}^t YP(t) \cdot dt + Y(tStart)$$

- $Y(t)$ repräsentiert in einem dynamischen Modell die Füllung eines Speicher-Elements mit Energie, Stoff bzw. Information (=Zustandsgröße).
 $YP(t)$ ist dabei die Änderungsgeschwindigkeit dY/dt für die Füllung des Speicherelements.
- Im Allgemeinen ist $YP(t)=f(Y(t))$, d.h. die Geschwindigkeit des Energie- /Stoff- /Info- Flusses zwischen den Speichern ist abhängig von den Potentialdifferenzen und den Übertragungselementen zwischen den Speichern.
- Da $YP(t)$ nicht explizit gegeben ist, kann $Y(t)$ nicht hinreichend genau durch numerische Verfahren der Flächenberechnung ermittelt werden (z.B.: [Trapez-/Tangentenformel](#) oder [Simpsonsche Regel](#)).
- Die konkreten zeitlichen Verläufe von $Y(t)$ und $YP(t)$ ergeben sich erst durch die zeitliche Wechselwirkung zwischen den Energiespeichern!

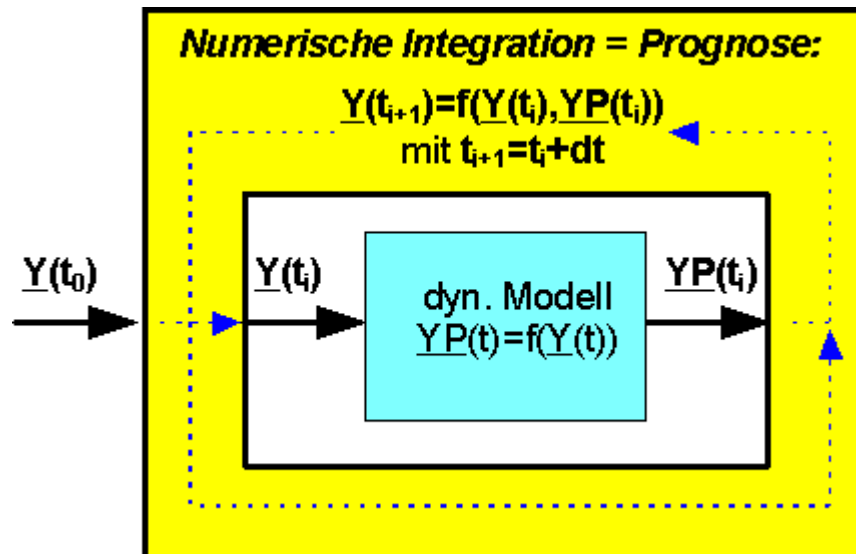


3.3. Modellierung und Simulation: Modellberechnung

3.3.1. Simulation zeitkontinuierlicher Systeme

----> Zentraler Integrationskern (Solver)

- Es existieren verschiedene Verfahren zur Lösung (=Simulation) von DGL mittels numerischer Integration. Diese Verfahren sind im "zentralen Integrationskern" = "Solver" implementiert.
- Die **Integral**- bzw. **Differential**-Funktion ist die Grundlage für die Verbindung zwischen Modell und Solver:
:
 $\mathbf{a} := \mathbf{F}(\mathbf{m});$
 $\mathbf{v} := \mathit{integral}(\mathbf{a}, \mathbf{v0});$
 $\mathbf{x} := \mathit{integral}(\mathbf{v}, \mathbf{x0});$
- Es sieht auf den ersten Blick so aus, als würde über diese Funktionsaufrufe der zentrale Integrationskern vom Modell aus aufgerufen, jedoch verhält es sich umgekehrt!
- Das Modell ist aus Sicht der numerischen Integration eine Black-Box:



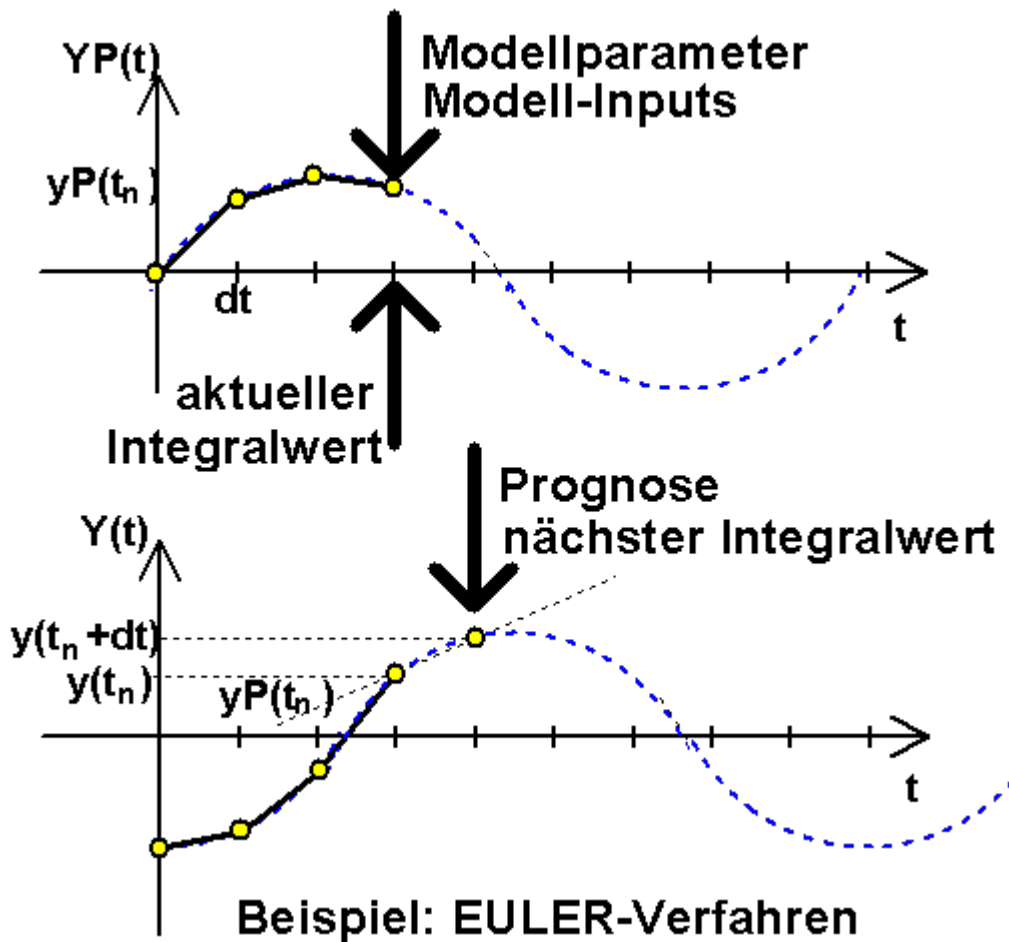


3.3. Modellierung und Simulation: Modellberechnung

3.3.1. Simulation zeitkontinuierlicher Systeme

----> Prinzip der numerischen Integration

- Das DGL-System wird durch die numerische Integration zum "Leben" erweckt. Es entwickelt eine Eigendynamik auf der Basis seiner Elemente (Speicher, Übertragungsglieder und Wandler) und ihrer Wechselwirkung.
- Der Solver ist in der Lage, über einen begrenzten zeitlichen Prognosehorizont den Zustand des Systems hinreichend genau vorauszusagen:

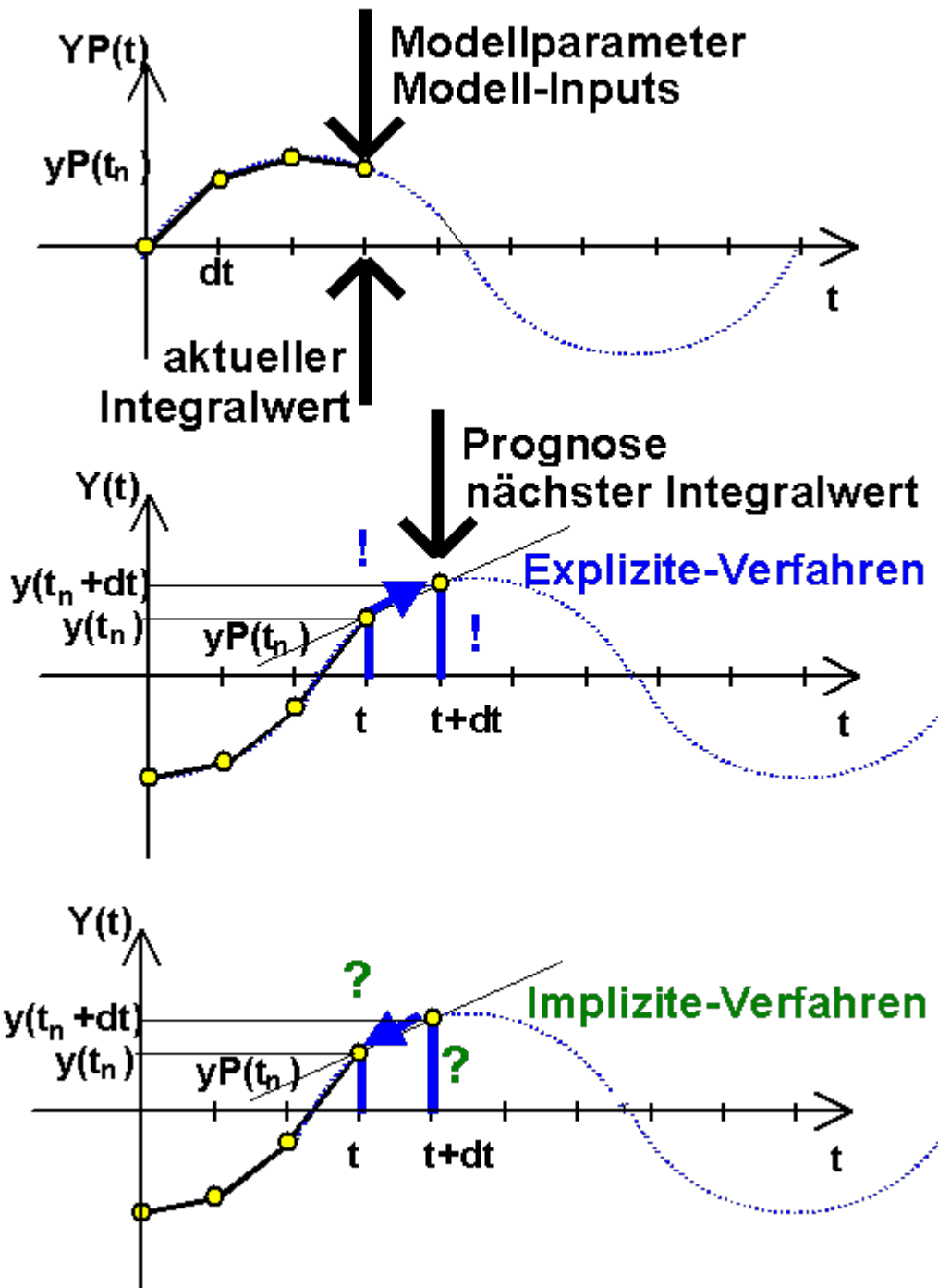




3.3. Modellierung und Simulation: Modellberechnung

3.3.1. Simulation zeitkontinuierlicher Systeme

----> Explizite und implizite Verfahren





3.3. Modellierung und Simulation: Modellberechnung

3.3.1. Simulation zeitkontinuierlicher Systeme

----> *Ein- und Mehrschrittverfahren*

Einschritt-Verfahren:

- "Betrachten" für die Prognose des nächsten Systemzustandes (Zeitpunkt= $t+dt$) ausgehend von $Y(t)$ nur den aktuell auszuführenden Zeitschritt dt .
- Je nach Genauigkeitsordnung des konkreten Verfahrens wird eine Anzahl von Systemabtastungen innerhalb dieses Zeitschrittes durchgeführt (=Modellrechnungen).
- Nach der erfolgreichen Prognose für $Y(t+dt)$ rückt das vom Verfahren betrachtete Zeitfenster um dt weiter.
- Da nur Informationen aus dem aktuellen Simulationsschritt verwendet werden, sind starke Änderungen der dynamischen Beziehungen relativ unproblematisch (Schrittweitenregelung).

Mehrschritt-Verfahren:

- Durch Bestimmung eines Interpolationspolynoms, das durch zurückliegende Zeit-Stützpunkte der Lösungsfunktion gelegt wird, kann deren weiterer Verlauf prognostiziert werden.
- Es wird zwischen reinen Mehrschrittverfahren (Adams-Bashforth-Verfahren) und den Prädiktor-Korrektor-Verfahren unterschieden.
- Bei Prädiktor-Korrektor-Verfahren wird der extrapolierte Wert der Lösungsfunktion in mehreren Iterationen korrigiert.
- Bei stetigen Modellen benötigen Mehrschritt-Verfahren weniger Abtastungen als Einschritt-Verfahren und sind somit schneller.
- Bei Unstetigkeiten geht dieser Vorteil durch die ständigen Wiederanlaufphasen schnell verloren (erste Schritte mit Einschritt-Verfahren!).



3.3. Modellierung und Simulation: Modellberechnung

3.3.1. Simulation zeitkontinuierlicher Systeme

----> Explizite Einschrittverfahren (Beispiele)

- Das **EULER-Verfahren** ist das einfachste Verfahren:
Man nimmt an, dass die Änderungsgeschwindigkeit der Zustandsgrößen über die Zeitspanne dt hinweg konstant bleibt:

$$Y(t + dt) := Y(t) + dt \cdot YP(Y(t))$$

Infolge der Aufsummierung der Prognosefehler wird das EULER-Verfahren sehr schnell instabil!

- Beim **HEUN-Verfahren** wird ein Korrekturschritt angefügt:
Mit der nach EULER prognostizierten Zustandsgröße $Y_1(t+dt)$ berechnet man die Ableitung $YP(Y_1(t+dt))$.
Dann nimmt man den Mittelwert beider Ableitungen als genauere Annahme für den Anstieg $YP(t)$:

$$Y(t + dt) := Y(t) + dt \cdot \frac{YP(Y(t)) + YP(Y_1(t + dt))}{2}$$

- Allgemein (**Runge-Kutta-Verfahren**):
Je nach Genauigkeitsordnung des konkreten Verfahrens wird der Funktionswert $YP(t)$ mit mehreren Stützstellen (Modellrechnungen) im betrachteten Zeitschritt "geschätzt"

$$Y(t + dt) := Y(t) + dt \cdot YP(t)$$

Durch Ermittlung von $YP(t)$ mittels zweier unterschiedlicher Genauigkeitsordnungen kann eine Fehlerabschätzung und Schrittweitenregelung erfolgen.

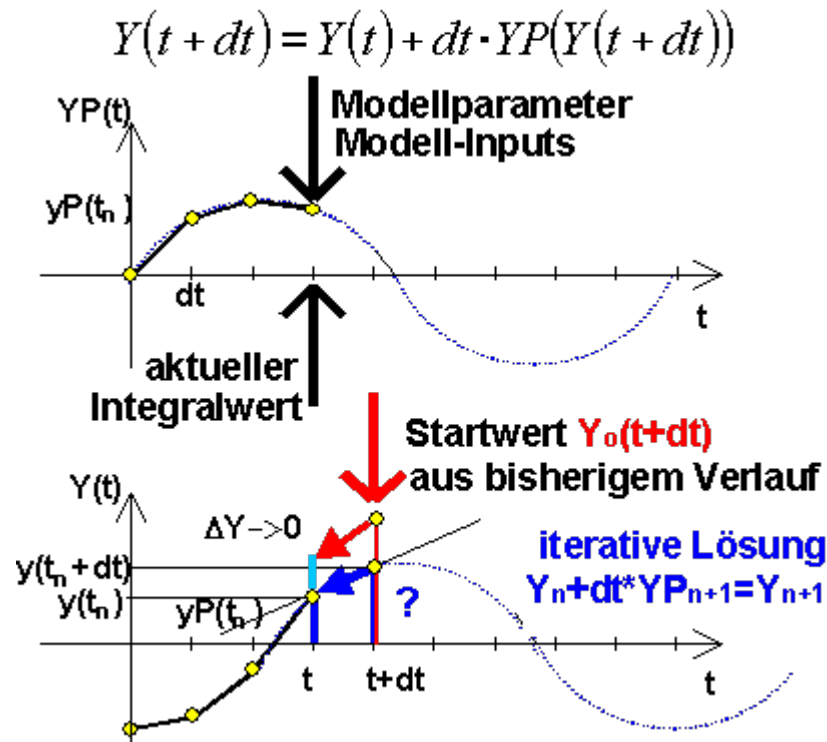


3.3. Modellierung und Simulation: Modellberechnung

3.3.1. Simulation zeitkontinuierlicher Systeme

----> *Implizite Verfahren (Prinzip)*

- In SimulationX sind das **BDF-Verfahren** (Backward-Differential Formulas) und ein modifiziertes und erweitertes BDF-Verfahren (MEBDF) implementiert.
- Dabei handelt es sich um implizite Verfahren mit einstellbarer Ordnung und automatischer Schrittweitensteuerung:



- aus bisherigem Y -Verlauf bis zum Zeitpunkt t Startwert-Ermittlung $Y_0(t+dt)$ als Ausgangspunkt für die "exakte" Lösung $Y(t+dt)$ je nach Ordnung des Verfahrens:

$$Y_0(t + dt) = f(Y(t), Y(t - dt), \dots, Y(t - n \cdot dt))$$

- iterative Lösung des nichtlinearen Gleichungssystems mit Abbruch bei der geforderten Genauigkeit.

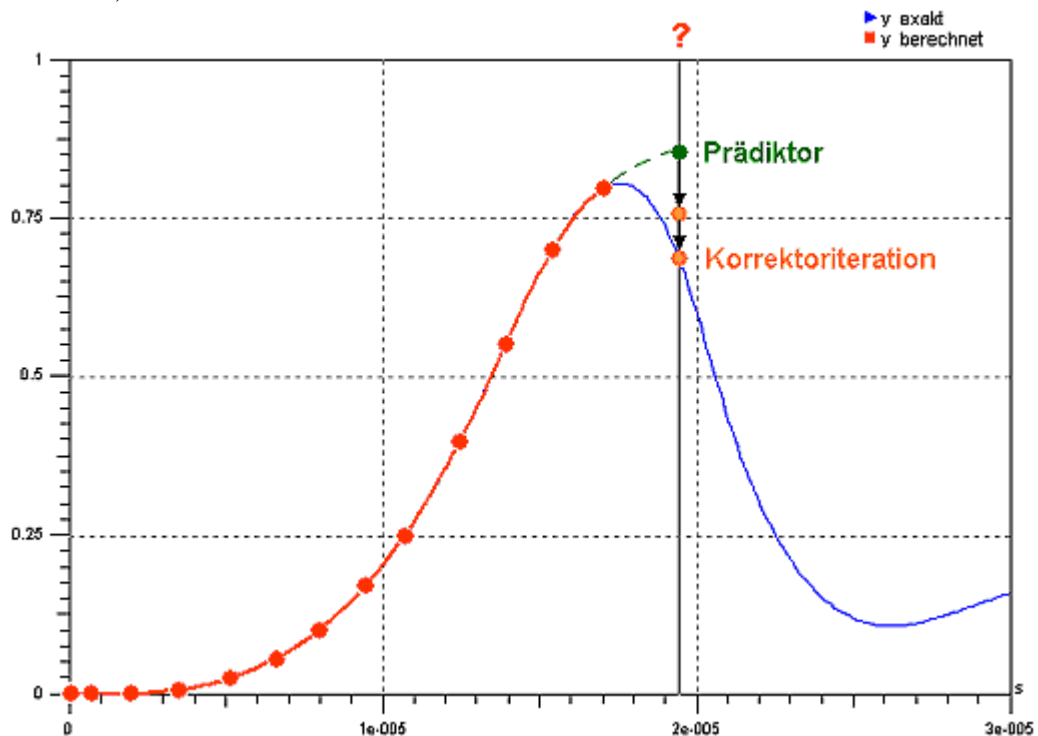


3.3. Modellierung und Simulation: Modellberechnung

3.3.1. Simulation zeitkontinuierlicher Systeme

----> Implizite Verfahren (Prädiktor-Korrektor-Verfahren)

- BDF- und MEBDF-Verfahren sind Prädiktor-Korrektor-Verfahren mit automatischer Steuerung der Schrittweite (Δt) und Ordnung (k). Die maximale Verfahrensordnung (k_{\max}) ist wählbar.
- Für jede Zustandsgröße wird aus $(k+1)$ bereits berechneten Werten der nächste aktuelle Wert extrapoliert (Prädiktor=Prognose durch explizites Mehrschrittverfahren).
- Dieser wird iterativ durch ein implizites Mehrschrittverfahren gleicher Ordnung so lange verbessert, bis bestimmte Konvergenzkriterien erfüllt sind (*corrector* = Verbesserer):



- In jeder Korrektor-Iteration muss ein lineares Gleichungssystem gelöst werden:
 $\mathbf{res} = \mathbf{J} \cdot (\mathbf{Y}_{i-1} - \mathbf{Y}_i)$ mit \mathbf{res} = Residuen / \mathbf{J} = Jacobimatrix / \mathbf{Y} = Zustandsgrößen
 - Das Residuum widerspiegelt den momentanen Fehler in den Differentialgleichungen.
 - Nach jeder Iteration wird die zu \mathbf{Y}_i gehörende Ableitung mittels Modellberechnung ermittelt und "rückwärts" überprüft, ob damit \mathbf{Y}_{i-1} erreicht wird (implizites Verfahren!).
 - Die Jacobi-Matrix enthält die Werte der partiellen Ableitungen aller Zustandsgrößen nach allen Zustandsgrößen (=Wechselwirkung zwischen Zustandsgrößen).
 - Die Jacobi-Matrix wird erst neu berechnet, wenn dies auf Grund von Genauigkeitskriterien als erforderlich erkannt wird (z.B. mangelnde Konvergenz). Die Berechnung erfolgt in modernen Systemen symbolisch, in älteren noch über Abtastung des Modells.

Quelle: SimulationX-Hilfe



3.3. Modellierung und Simulation: Modellberechnung

3.3.1. Simulation zeitkontinuierlicher Systeme

----> Qualitäten von Integrationsverfahren

Explizite Verfahren:

- max. zulässiges Δt wird dadurch bestimmt, wie schnell sich im dynamischen Modell Zustandsgrößen ändern können.
- der Wert T repräsentiert die kleinste Zeitkonstante bzw. die Periodendauer der höchsten Eigenfrequenz (z.B. C/R , L/R , $\sqrt{L \cdot C}$, $\sqrt{m/c}$)

| <u>Beispielverfahren</u> | <u>max. Schrittweite</u> | <u>Verhalten bei Δt zu groß</u> |
|--------------------------|--------------------------|--|
| EULER | ca. $T/1000$ | stark aufschaukelnd |
| HEUN | ca. $T/10$ | schwach aufschaukelnd |
| Runge-Kutta 4.Ordn. | ca. $T/5$ | dämpfend, dann chaotisch |
| Runge-Kutta-Merson* | ca. $T/3$ | dämpfend, dann chaotisch |

Implizite Verfahren:

- Nur ab einer deutlichen Erregung von schwingungsfähigen Teilsystemen des dynamischen Modells hat deren Zeitkonstante Auswirkung auf die Integrationsschrittweite. Dann gilt:

| <u>Beispielverfahren</u> | <u>max. Schrittweite</u> | <u>Verhalten bei Δt zu groß</u> |
|--------------------------|--------------------------|--|
| Weighted Average | ca. $T/200$ (Einschritt) | dämpfend, dann chaotisch |
| Rosenbrock-Wanner* | ca. $T/3$ (Einschritt) | stochastisch |
| DBF / MEBDF* | ca. $T/3$ (Mehrschritt) | Fehlermeldung |

- Vernachlässigen "Mikroschwingungen" in steifen DGL -> großes Δt !
- Insbesondere die DBF-Verfahren, bekannt auch als **DASSL** (Differential Algebraic System Solver) sind hervorragend für steife DGL mit algebraischen Schleifen $Y=f(Y)$ geeignet.
- Da algebraische Schleifen auf Grund des Modellkonzepts in SimulationX die Regel sind, hat man (leider) auf die Implementierung "normaler" Einschrittverfahren verzichtet!



3.3. Modellierung und Simulation: Modellberechnung

3.3.1. Simulation zeitkontinuierlicher Systeme

----> *Konfiguration des Solvers*

Geschwindigkeit, Genauigkeit und Durchführbarkeit eines Simulationslaufes hängen von der Konfiguration des Solvers ab:

Integrationsverfahren: Entscheidung für ein Verfahren auf Basis der Modelleigenschaften (Erfahrungswissen nach Versuch und Irrtum!), z.B. SimulationX:
BDF/MEBDF/ext.Solver+Ordnung+lin.Gleichungslöser
(1..5 1..7 ← *Ordnung*) ("*schwachbesetzte Jacobi-Matrix*")

Simulationszeitbereich: *tStart ... tStop*

Integrationschrittweite: *dt* oder Intervall *dtMin ... dtMax*

Protokollschrittweite: *dtProtMin* für Aufzeichnung von Signalverläufen (min. möglicher Abstand zwischen protokollierten Punkten).
Alle Unstetigkeiten bei Sprüngen im Signalverlauf!
z.B. $(tStop - tStart) / 1000$ benutzen!

Genauigkeit: *absTol* für Iterationsgenauigkeit des Gleichungslösers in Hinblick Residuum (0);
→ *absTol* vergrößern, wenn z.B. "Genauigkeit nicht erreicht"
relTol & *absTol* für dt-Regelung in Hinblick auf max. zulässige Änderung der num. Zustandsgrößen pro Schritt.
→ *relTol* verringern, wenn zu ungenau
dtDetect = $dtMin * 1e-4$ für Ereignisbehandlung

max.zul.Abweichung pro Simulationsschritt:
= $relTol * Zustandsgröße + absTol$



3.3. Modellierung und Simulation: Modellberechnung

3.3.2. Zeitdiskrete Ereignisse

----> *Ereignisse im Simulationslauf*

- Was ist ein Ereignis?
 1. Das Zustandekommen einer bestimmten Konstellation kann als Ereignis definiert werden (meist vor dem Ereignis).
 2. Es muss ein Beobachter existieren, damit Ereignisse bemerkt werden.

- Eigenschaften von Ereignissen
 1. Als Ereignis wird meist der Augenblick des Zustandekommens der besonderen Umstände gewertet. Es tritt dann zu einem diskreten Zeitpunkt schlagartig ein (Ja/Nein).
 2. *Prognostizierbare Ereignisse:*
Zeitpunkt ist durch Beobachtung der sich ändernden Umstände voraussagbar.
 3. *Zufällige Ereignisse:*
Wenn sich die Umstände aus Sicht des Beobachters stochastisch ändern, ist der Zeitpunkt erst nach Eintritt des Ereignisses bekannt.

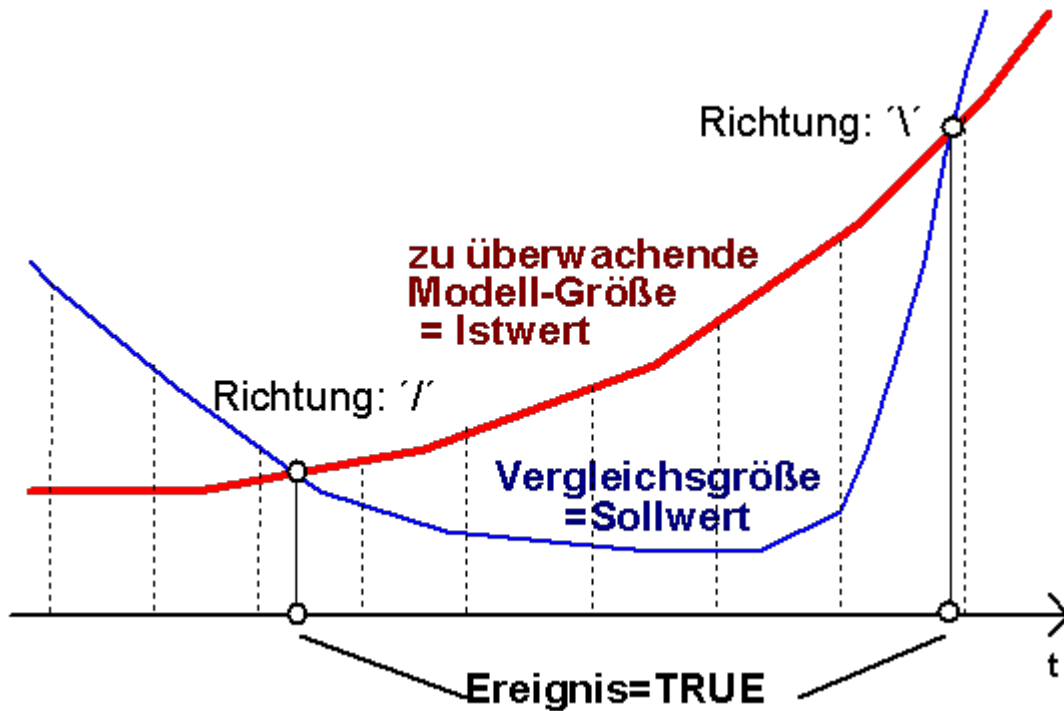


3.3. Modellierung und Simulation: Modellberechnung

3.3.2. Zeitdiskrete Ereignisse

----> *Numerische Definition von Ereignissen*

- Erreichen eines Zustands während des Modellaufes, in dem eine sich zeitlich ändernde Größe den Wert einer Vergleichsgröße annimmt:



- In SimulationX wird als Ereignisbeobachter intern die so genannte Zero-Function genutzt. Diese registriert, bewertet und behandelt die Nulldurchgänge der Differenz von Ist- und Sollgröße.
- Zero-Functions werden automatisch für alle unstetigen Funktionen (sign, abs, ...) und für die Testbedingung in Alternativ- und Schleifenanweisungen generiert.



3.3. Modellierung und Simulation: Modellberechnung

3.3.2. Zeiddiskrete Ereignisse

----> Klassifizierung numerischer Ereignissen

- Numerische Ereignisse bewirken:
 - Unstetigkeiten von Zustandsgrößen $Y(t)$ oder ihren Ableitungen $\dot{Y}(t)$ oder
 - Umschaltungen der Modellstruktur (d.h. des Gleichungssystems)

| Zustandsabhängige Unstetigkeiten | Zeitabhängige Unstetigkeiten |
|---|--|
| <ul style="list-style-type: none">• werden durch Zerofunctions erkannt (ändern zum Ereigniszeitpunkt ihr Vorzeichen)• Ausdrücke: $\text{abs}(x)$, $\text{sign}(x)$ bedingte Anweisungen: <code>if..then..else</code>• Solver versucht, durch Prognose den Zeitpunkt möglichst genau zu treffen• Die kontinuierliche Integration wird kurz <u>hinter</u> dem tatsächlichen Ereigniszeitpunkt gestoppt (maximal dt_{Detect} danach).• Zu diesem prognostizierten Zeitpunkt wird die Ereignisbehandlung gestartet | <ul style="list-style-type: none">• Zeiddiskrete Signalglieder:<ul style="list-style-type: none">◦ AD-/DA-Wandler◦ Abtast- & Halteglied◦ diskrete Integratoren / Differentierer◦ Z-Übertragungsfunktion• Ausdrücke: <code>sample()</code> bzw. <code>delay()</code> zum Setzen von Zeit-Ereignissen bzw. Signal-Verzögerung• Eintrag in Ereigniswarteschlange von SimX: Der Solver startet exakt zu diesen Zeitpunkten die Ereignisbehandlung |



3.3. Modellierung und Simulation: Modellberechnung

3.3.2. Zeitdiskrete Ereignisse

----> Ereignisbehandlung (SimulationX)

Die Ereignisbehandlung soll einen neuen konsistenten Zustand aller Zustandsgrößen nach der Unstetigkeit herstellen:

- Während der Ereignisbehandlung bleibt die Modellzeit konstant (*time* bzw. *t*).
- Zu diesen Zeitpunkt wird eine so genannte *Ereignis-Iteration* durchgeführt.
- Die Werte der elementaren Relationen (**true** bzw. **false**) bleiben während der kontinuierlichen Integration konstant, d.h. kurz vor einer Ereignis-Iteration wird noch mit den alten Werten gerechnet, obwohl sich die Relation schon geändert haben kann (Stopp nach dem wirklichen Ereigniszeitpunkt!)
- Während der Ereignis-Iteration ändern die Relationen ihre Werte und danach werden für den gleichen Zeitpunkt neue konsistente Anfangswerte bestimmt. So erhält man zwei Werte eines Werteverlaufs an dem Zeitpunkt der Unstetigkeit: einen vor der Iteration und einen danach.
- Mit dem noEvent-Operator kann man verhindern, dass ausgewählte Relationen Ereignisiterationen auslösen. Das kann man z.B. dazu benutzen, um den Definitionsbereich einer Funktion abzusichern:

```
y:= if noEvent(x>=0) then sqrt(x) else 0;
```

Ohne noEvent würde ein Fehler auftreten, weil kurz vor der Iteration *x* schon negativ wird, obwohl $x \geq 0$ noch "true" ist.



3.3. Modellierung und Simulation: Modellberechnung

3.3.2. Zeiddiskrete Ereignisse

----> *Beispiel: Extremwert-Sensoren (SimulationX)*

Eine typische Modellierungsaufgabe ist die Erfassung des Extremwertes einer physikalischen Größe während des Simulationslaufes:

- Vor der Dynamik-Berechnung im Zeitbereich muss die Extremwert-Variable einen sinnvollen Startwert erhalten (*vMaxInit als Anfangswert von vMax*).
- Man muss gewährleisten, dass die "irreversible" Speicherung der aktuellen Extremwerte nur auf Basis "gültiger" Modellwerte erfolgt (*last*).
- Die Verwendung unstetiger Funktionen sollte nicht zu unnötigen Ereignisbehandlungen führen (*noEvent*).
- Entnimmt man die zu überwachende Größe einem energetischen Konnektor, so muss man zur Vermeidung undefinierter Rückwirkungen die Flussgröße auf Null setzen (*ctrl.F:=0*):

The screenshot shows the SimulationX environment. On the left, a block diagram for 'ctr1 (Mechanischer Anschluss)' is visible, containing variables: x (Weg), v (Geschwindigkeit), a (Beschleunigung), F (Externe Kraft), vMaxInit (Startwert), a (Beschleunigung;), v (Geschwindigkeit aktuell), and vMax (Geschwindigkeit maximal). On the right, the 'Algorithmus' editor contains the following code:

```
1 v := ctrl.v;  
2 vMax:= if noEvent (abs((last(v)))>vMax)  
3   then noEvent(abs(last(v))) else vMax;  
4  
5 // v-Extremwerte bei Nulldurchgang a:  
6 a := if(ctrl.a==0) then 0 else ctrl.a;  
7  
8 ctrl.F:=0; // Flussgröße zuweisen!
```

Hinweise:

- Tritt der Extremwert nicht an einer bereits behandelten Unstetigkeitsstelle auf, so muss gezielt eine solche Testbedingung installiert werden! Nur so wird auch der Extremwert-Zeitpunkt berechnet (z.B. z.B. Nulldurchgang von a).
- Sensoren der Modellbibliothek liefern nicht nur "gültige" Signale. Das muss man ebenfalls durch Anwendung der last-Funktion berücksichtigen.



3.3. Modellierung und Simulation: Modellberechnung

3.3.3. Lineare und nichtlineare Systeme

----> *Modellcharakter von Systemen*

Der Begriff "System" wird häufig mit verschiedenen Zusätzen benutzt:

- lineares bzw. nichtlineares System
- chaotisches System
- offenes, geschlossenes, abgeschlossenes System
- sequentielles und paralleles System
- emergentes System oder Vivisystem;

Was ist ein System?

- besitzt eine Abgrenzung zu seiner Umgebung;
- enthält Komponenten, die Wirkung aufeinander auswirken;
- es entsteht ein Systemverhalten, welches eine neue Qualität im Vergleich zu den Elementen aufweist (z.B. Spule und Kondensator);
- die resultierenden Systemgesetze engen den Freiheitsgrad der Komponenten ein! (Versklavung der Komponenten);
- ist immer ein *idealisiertes, vereinfachtes Abbild* des betrachteten Gebildes;
- ist deshalb Modell (ein Ersatzobjekt) zum Verstehen komplexer Gebilde;



3.3. Modellierung und Simulation: Modellberechnung

3.3.3. Lineare und nichtlineare Systeme

----> *Dynamische Systeme*

- Es existiert ein Vektor von *kontinuierlichen* bzw. *diskreten* Zustandsgrößen $Y(t)$, d.h., das System *enthält Speicherelemente*.
- es gibt eine Überföhrungsfunktion T (= *dynamisches Modell des Systems!*):

$$Y(t + \Delta t) = T(Y(t); t; u(\tau); \Delta t)$$

mit

$$u(\tau) = \text{Eingangsvektor des Systems für } t \leq \tau \leq t + \Delta t$$

deterministisches System: Überföhrungsfunktion T ist eindeutig

stochastisches System: Überföhrungsfunktion T ist mehrdeutig

autonomes System: wenn T zeitunabhängig:

$$Y(t + \Delta t) = T(Y(t); u(\tau); \Delta t)$$

- **Beispiele:**

- Ein Feder-Masse-Schwinger lässt sich als kontinuierliches, deterministisches und autonomes System beschreiben (DGL).
- **Biotope** lassen sich als autonome, stochastische Systeme beschreiben (Übergangswahrscheinlichkeiten).



3.3. Modellierung und Simulation: Modellberechnung

3.3.3. Lineare und nichtlineare Systeme

----> *Wann kann man ein System als Linear behandeln?*

- Für die Behandlung linearer Systeme existiert ein umfangreiches mathematisches Methoden-Inventar.
- Das Verhalten linearer Systeme kann analytisch durch Anwendung der entsprechenden mathematischen Verfahren auf der Basis linearer Gleichungssysteme berechnet werden.
- **Beispiel "Theorie der Wechselstromschaltung":**
Wenn die Schaltelemente **konstante** Parameter R , L , C besitzen und die Schaltung mit Sinusspannung gespeist wird, kann man durch Transformation in den Bildbereich der komplexen Zahlen lineare GLS aufstellen und lösen.
- Sind Speicher- und Übertragungselemente konstant, so kann man Transformationen in einen Bildbereich durchführen, wenn sich der Eingangsvektor $u(t)$ des Systems auf ein Spektrum harmonischer Schwingungen zurückführen läßt:
 - Sinus (Bildbereich der komplexen Zahlen),
 - Sprungfunktion und DIRAC-Impuls (Laplace-Operatoren),
 - beliebige periodische Vorgänge (Z-Transformation).



3.3. Modellierung und Simulation: Modellberechnung

3.3.3. Lineare und nichtlineare Systeme

----> *Wann muss man ein System als Nichtlinear behandeln?*

1. es enthält speichernde Komponenten, deren Fassungsvermögen nicht konstant ist (z.B. Pirouette)

oder

2. es enthält Übertragungselemente zwischen den Speichern, deren Übertragungsfunktion abhängig ist vom aktuellen Wert der übertragenen Größe (durch sich ändernde Werkstoffeigenschaften)

oder

3. der Eingangsvektor $u(t)$ ist aus Sicht des Systems zufällig.
-

- DGL beschreiben den Fluss von Energie, Stoff oder Information zwischen den Speicherkomponenten zeitkontinuierlicher dynamischer Systeme.
- Auch für nichtlineare Systeme ist das Aufstellen der DGL meist einfach, jedoch ist ihre analytische Lösung äußerst schwierig bis unmöglich.
- "Zum Glück" können diese nichtlinearen dynamischen Modelle numerisch mit geeigneten Simulationssystemen berechnet werden.