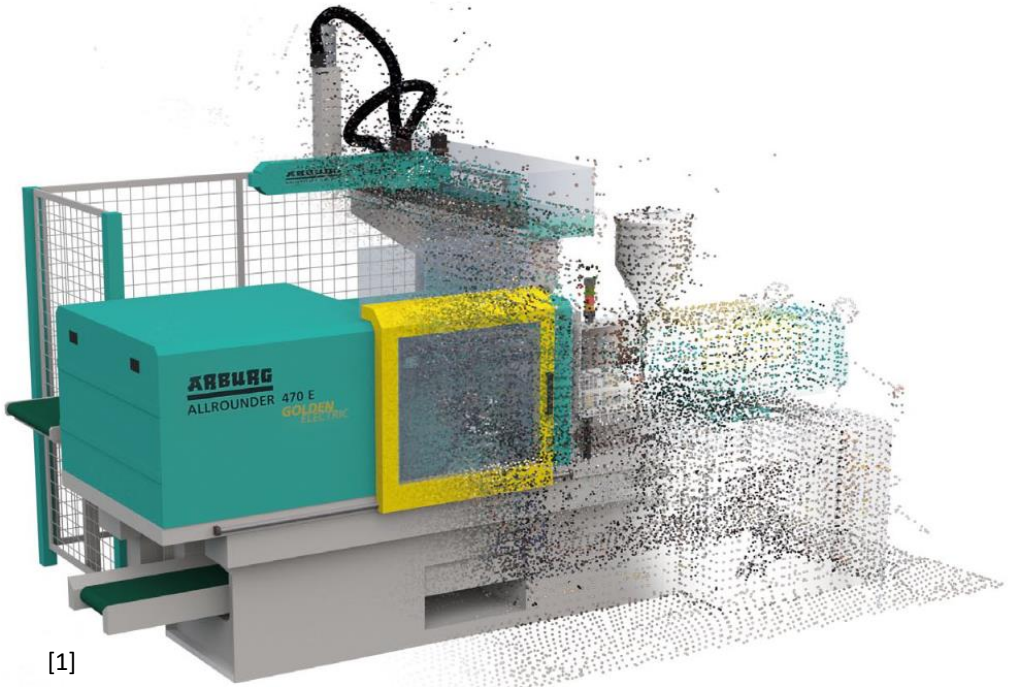
EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

Modellbildung Teil 1 Seminar & Workshop
Digital Twin of Injection Molding (DIM)

15.03.2022



[1]

Digital Twin of Injection Molding

Forschungskonsortium

- Institut für Werkstofftechnik / FG Kunststofftechnik, Prof. Dr.-Ing. H.-P. Heim
- FG Mess- und Regelungstechnik, Prof. Dr.-Ing. A. Kroll

Alexander Rehmer

+

Marco Klute

+

Stefan Rosenbach

+

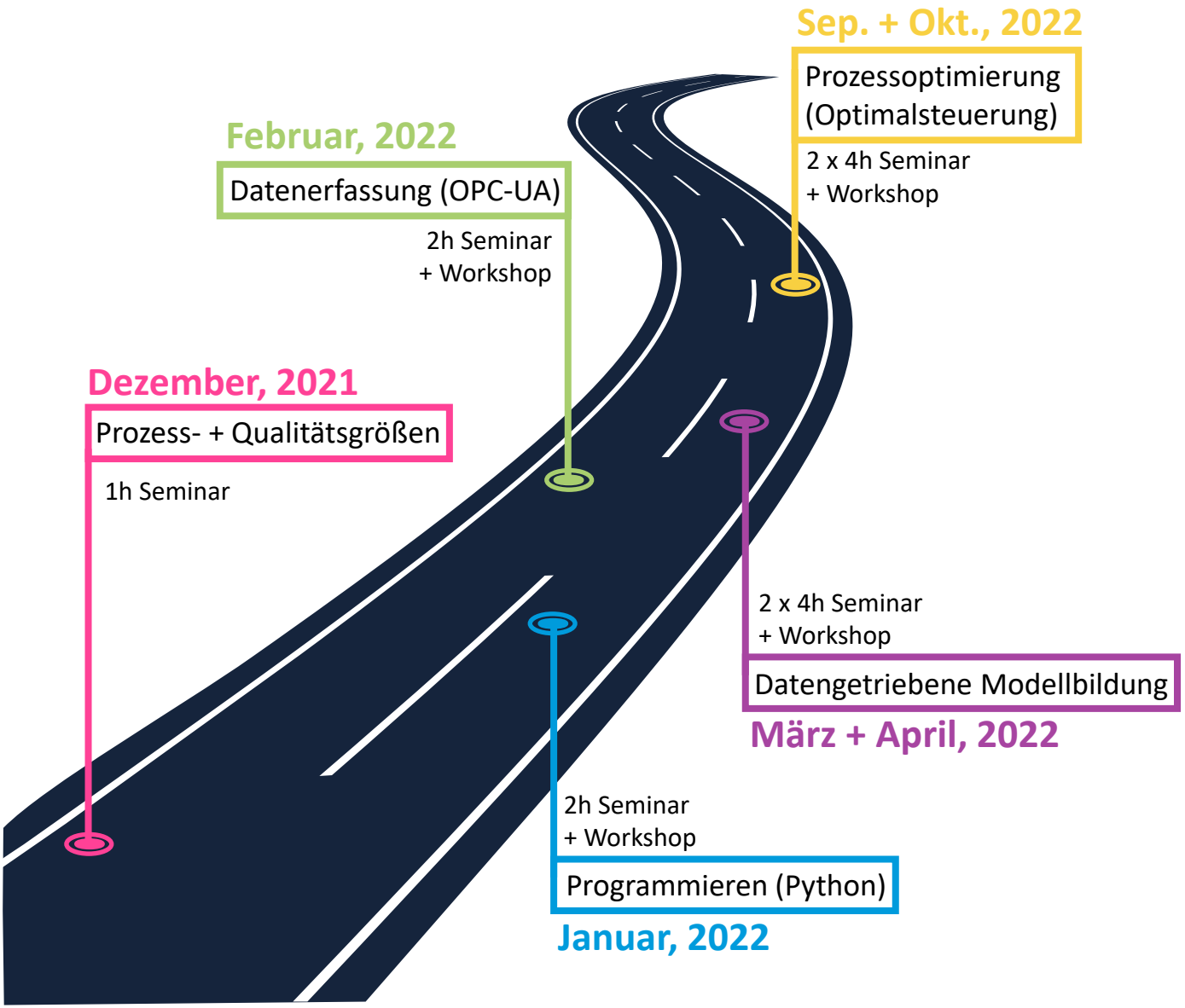
Studentische Hilfskräfte

Kontakt:

- dim@uni-kassel.de
- www.uni-kassel.de/go/DIM

Überblick über die Workshopreihe

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung



- Erfassung von Prozess- und Qualitätsgrößen
- Programmieren mit Python
- Datenerfassung mit OPC-UA
- Datengetriebene Modellbildung
  - Seminar: Grundlagen der datengetriebenen Modellbildung und linearen/nichtlinearen Optimierung mit CasADi
  - Workshop: Selbstständige Bearbeitung von akademischen Modellbildungsaufgaben
  - Ziele:
    - Modellbildung als Optimierungsproblem verstehen
    - Lösen einfacher Optimierungsprobleme mit CasADi
- Prozessoptimierung mittels numerischer Optimalsteuerung

Agenda

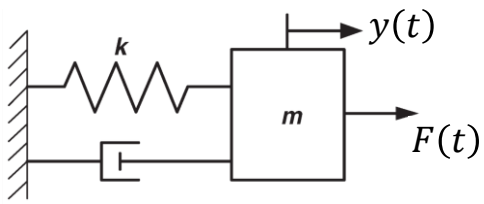
EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- **Grundlagen der datengetriebenen Modellbildung**
  - Physikalische vs. datengetriebene Modellbildung
  - Ablauf der datengetriebenen Modellbildung
  - Designentscheidungen
- **Grundlagen der Optimierung**
  - Lineare vs. nichtlineare Optimierungsprobleme
  - Lineare Optimierung
  - Nichtlineare Optimierung
- **Einführung in das CasADi Framework**
  - Automatisches Differenzieren
  - Was ist CasADi? Warum CasADi?
  - CasADi Basics
  - Differenzieren mit CasADi
  - Lösen quadratischer und nichtlinearer Programme
- **Rechnerübung**

Modellbildung – phys. vs. datengetrieben

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

Physikalische Modellbildung

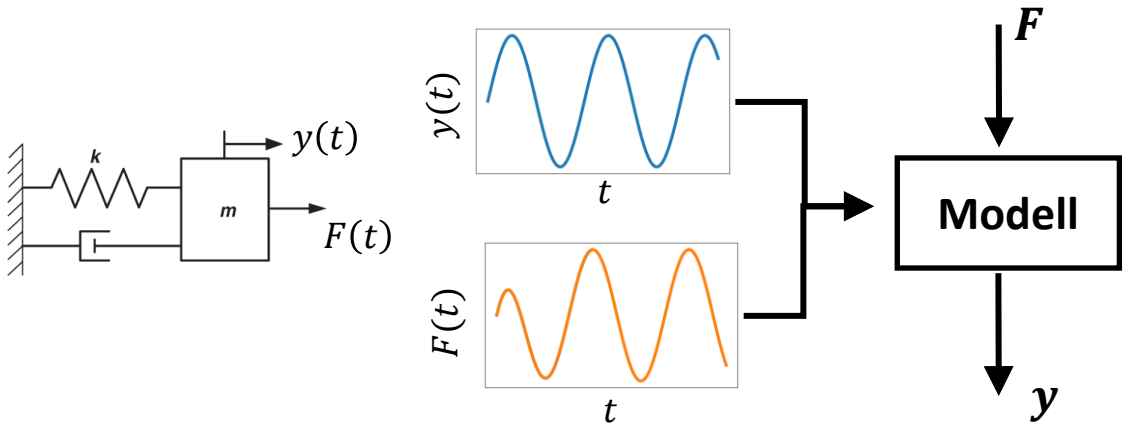


$$m\ddot{y}(t) = \sum F_i = -d\dot{y}(t) - ky(t) + F(t)$$

$$\rightarrow y(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t} + F(t)$$

- Bildung eines mathematischen Modells eines (dynamischen) Systems durch Anwendung physikalischer Prinzipien (Bilanzgleichungen und Komponentengleichungen)
- Erfordert
  - weitgehend vollständiges physikalisches Verständnis des zu modellierenden Systems
  - Kenntnis oder experimentelle Bestimmbarkeit der physikalischen Parameter des Systems

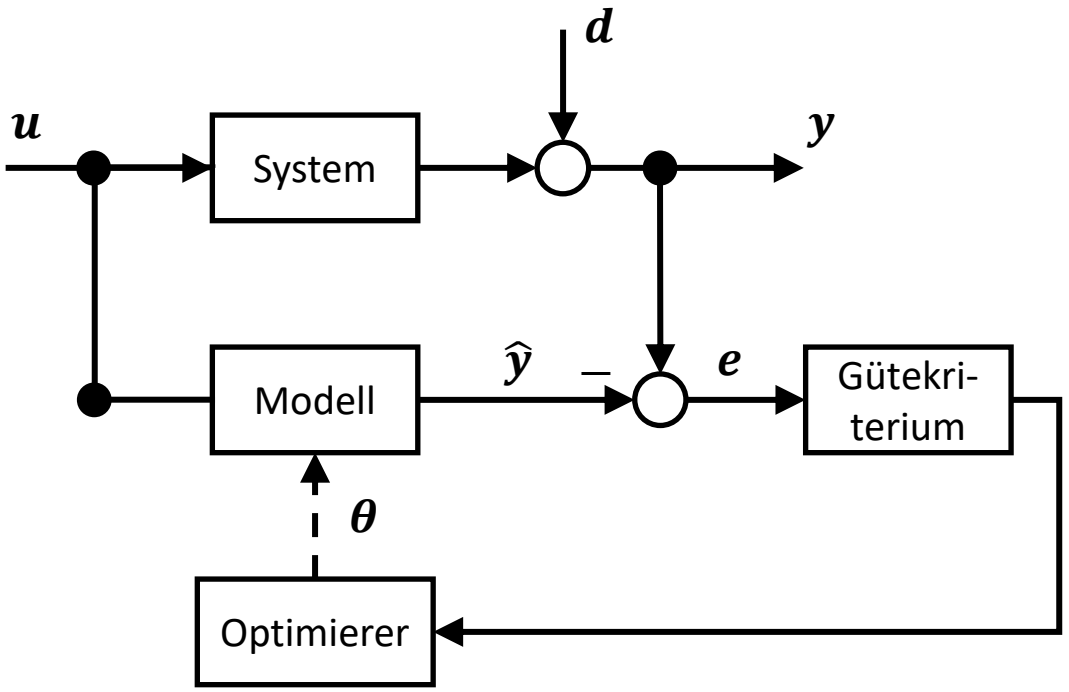
Datengetriebene Modellbildung



- Bildung eines mathematischen Modells eines (dynamischen) Systems unter Verwendung von Messungwerten der Ein- und Ausgangsgrößen des Systems
- Die Struktur des Modells kann physikalisch motiviert, aber auch ein Black-Box Ansatz, wie ein Neuronales Netz sein.
- Erfordert
  - Messwerte der Ein- und Aussgangsgrößen des Systems (im Zeit- und/oder Frequenzbereich)

Ablauf der datengetriebenen Modellbildung

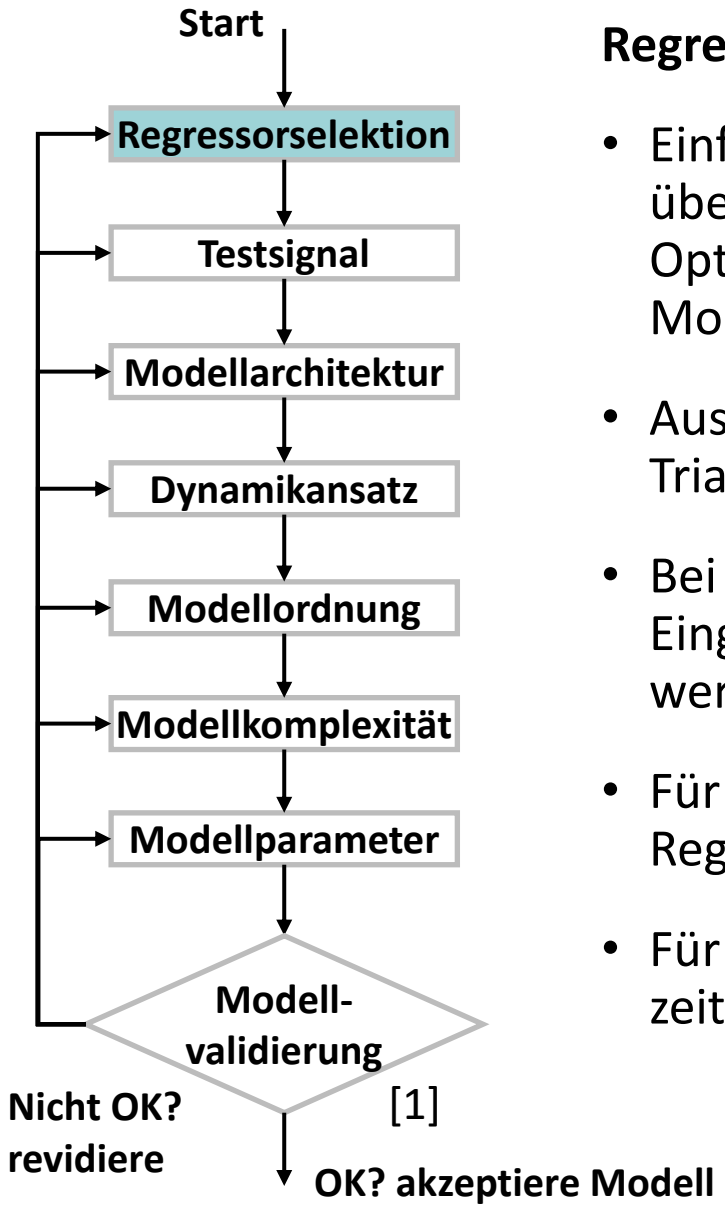
EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung



- System und Modell werden mit geeigneten Testsignalen angeregt
- Der Modellansatz kann aus theoretischen Überlegungen (physikalisch) folgen, muss aber nicht (Black-Box)
- Aus den Ausgangssignalen (System $\boldsymbol{y}$ und Modell $\hat{\boldsymbol{y}}$) wird die Differenz $\boldsymbol{e}$ gebildet und anhand eines Gütekriteriums (Kostenfunktion) bewertet
- Das Ergebnis der Bewertung wird von einem Optimierungsverfahren verwendet, um die Modellparameter anzupassen
- Das Optimierungsverfahren wird beendet, sobald eine gewünschte Güte erreicht oder ein anderes Abbruchkriterium erfüllt ist

Designentscheidungen

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung



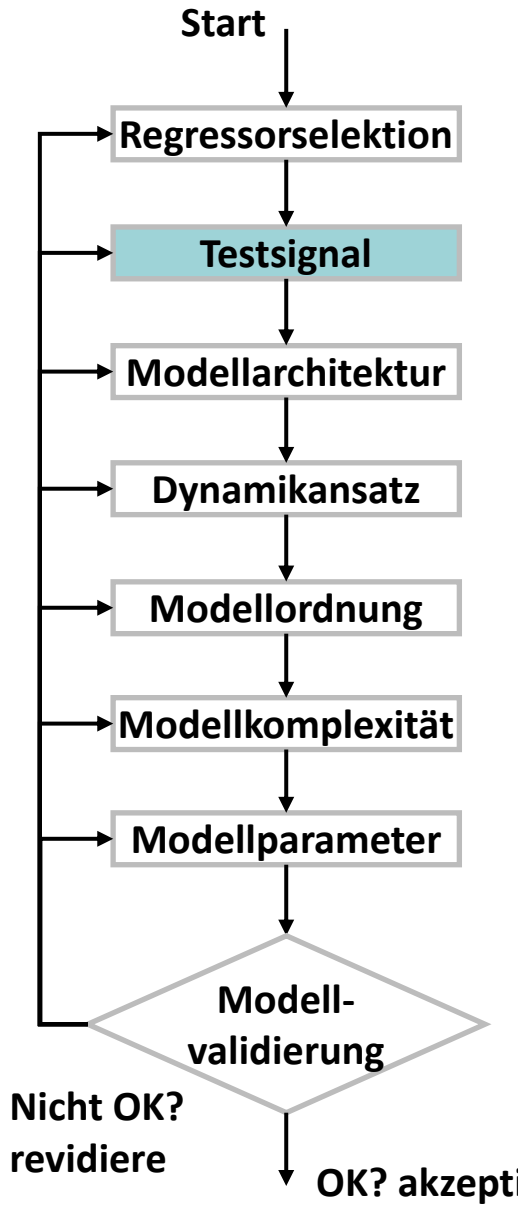
[1]

Regressorselektion

- Einfach alle potentiellen Eingangsgrößen zu verwenden ist möglich, kann aber zu übermäßig vielen Modelleingängen (und –parametern) führen und das Optimierungsproblem erheblich erschweren. Daher ist eine Selektion relevanter Modelleingänge notwendig.
- Auswahl der Modelleingangsgrößen $\boldsymbol{u}$ meist eine Kombination aus Vorwissen und Trial-and-Error
- Bei gut verstandenen (z.B. mechanischen) Prozessen können die relevanten Eingangsgrößen aufgrund des hohen physikalischen Verständnisses bestimmt werden.
- Für lineare Modelle existieren effiziente rechnergestützte Methoden zur Regressorselektion (Korrelationsanalyse, schrittweise Regression, ...)
- Für nichtlineare Modelle ist die Regressorselektion ein sehr rechen- und zeitaufwändiges Opimierungsproblem.

Designentscheidungen

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

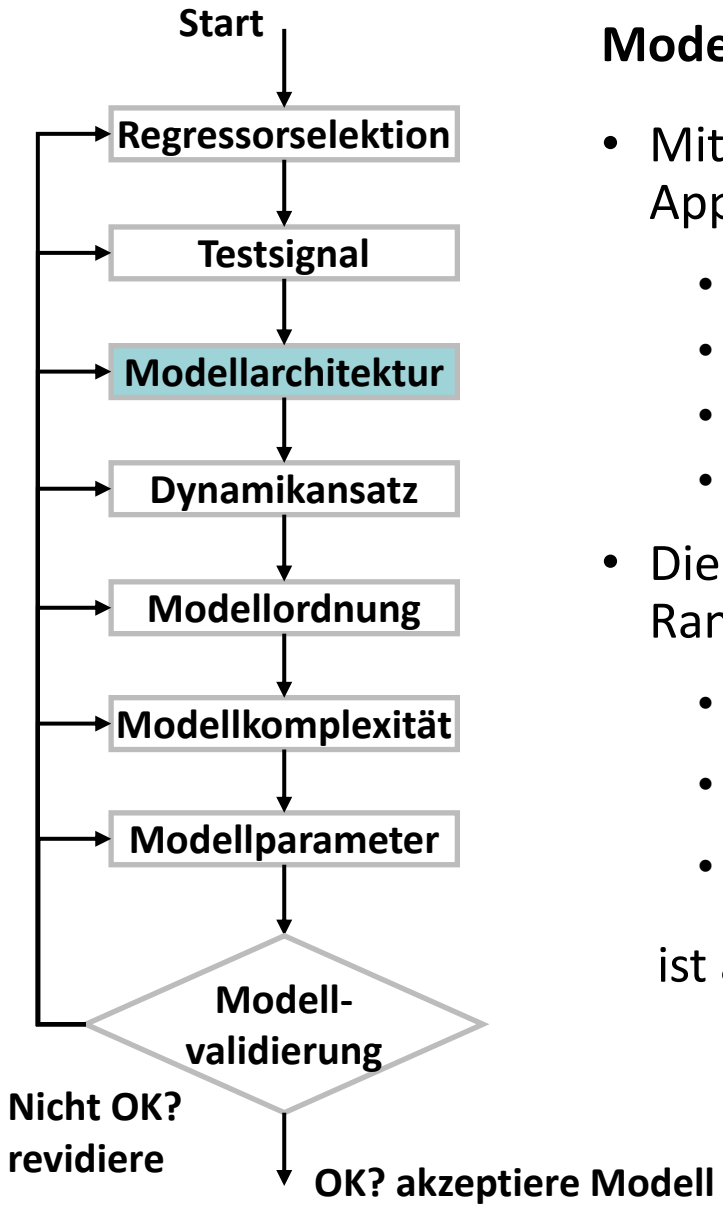


Testsignal

- Systemverhalten, das nicht in den Daten „enthalten” ist kann vom Modell nicht abgebildet werden
- Die Wahl des Testsignals ist daher ausschlaggebend für die maximal erreichbare Modellgüte
- Die Wahl des Testsignals ist stark anwendungsspezifisch, es existieren aber einige grundlegende Prinzipien die zu beachten sind:
  - Das Testsignal sollte die größtmögliche realisierbare Amplitude aufweisen, um ein bestmögliches Signal-Rausch-Verhältnis zu erreichen
  - Das Testsignal sollte den relevanten Frequenzbereich abdecken, in dem das System betrieben wird
  - Die meistgenutzten Testsignale sind pseudobinäre Signale (PRBS, für lineare Systeme) und amplitudenmodulierte pseudobinäre Signale (APRBS, für nichtlineare Systeme) sowie Multi-Sinus-Signale.
  - Die minimale Haltezeit des APRBS sollte etwa der größten Zeitkonstante des Systems entsprechen. Nur dann hat der Prozess nach jeder Stufe etwas Zeit, sich seinem neuen Gleichgewichtszustand anzunähern, sodass das nichtlineare statische Verhalten in den Messdaten enthalten ist.
  - Geeignete Wahl der Abtastrate

Designentscheidungen

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung



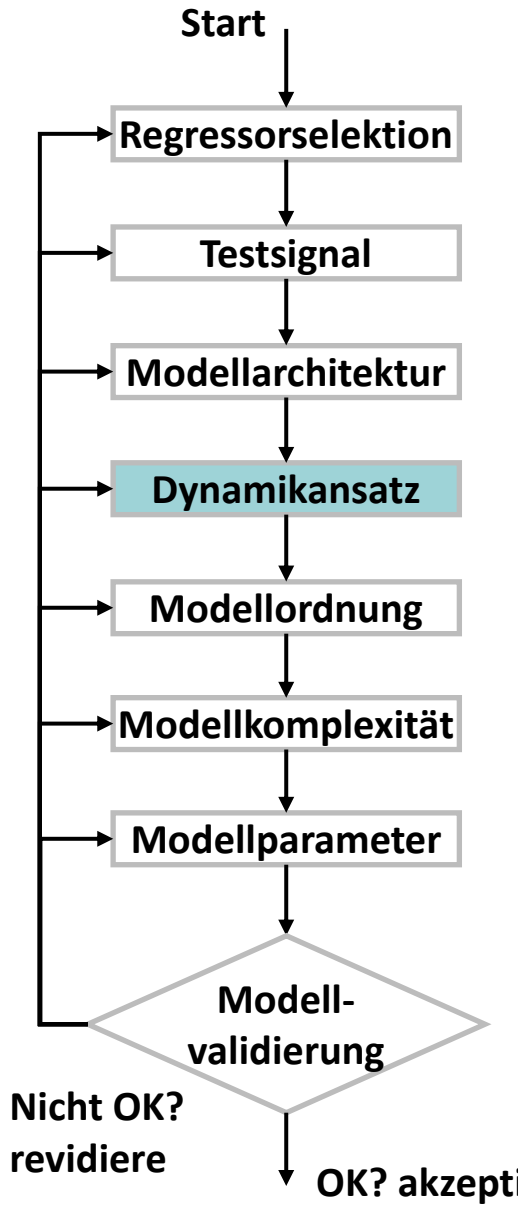
Modellarchitektur

- Mit Modellarchitektur ist der grundlegende funktionale Ansatz, der zur Approximation des Systems verwendet werden soll, gemeint. Bspw.:
  - Multilayer Perceptron (MLP)
  - Polynom
  - Takagi Sugeno Modell (TS)
  - Übertragungsfunktion
- Die Wahl der Modellarchitektur wird durch die Anforderungen und Randbedingungen des Modellierungsproblems bestimmt
  - Problemtyp: linear-nichtlinear, dynamisch-statisch, Anzahl relevanter Modelleingänge
  - Anwendung des Modells: Prädiktion vs. Reglerentwurf
  - Verfügbare Datenmenge und -qualität

ist aber auch stark von den subjektiven Erfahrungen des Nutzers abhängig

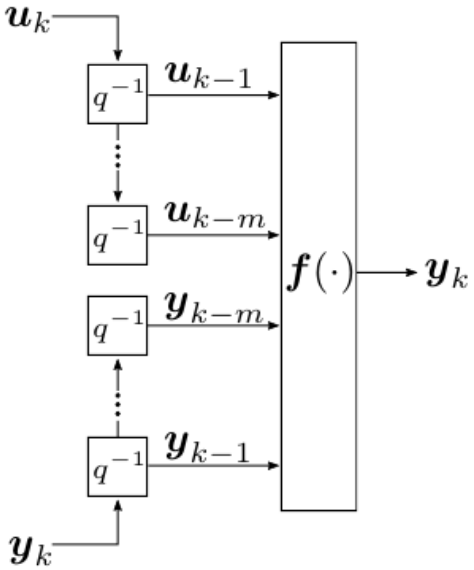
Designentscheidungen

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

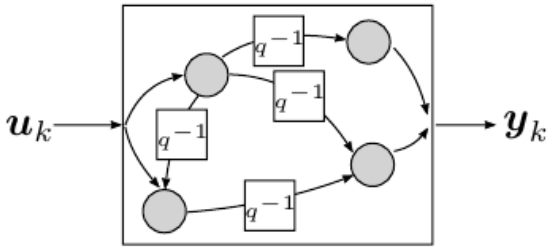


Dynamikansatz

- Es existieren zwei verschiedene Ansätze, um dynamisches Verhalten durch ein Modell abzubilden:



Externer Dynamikansatz

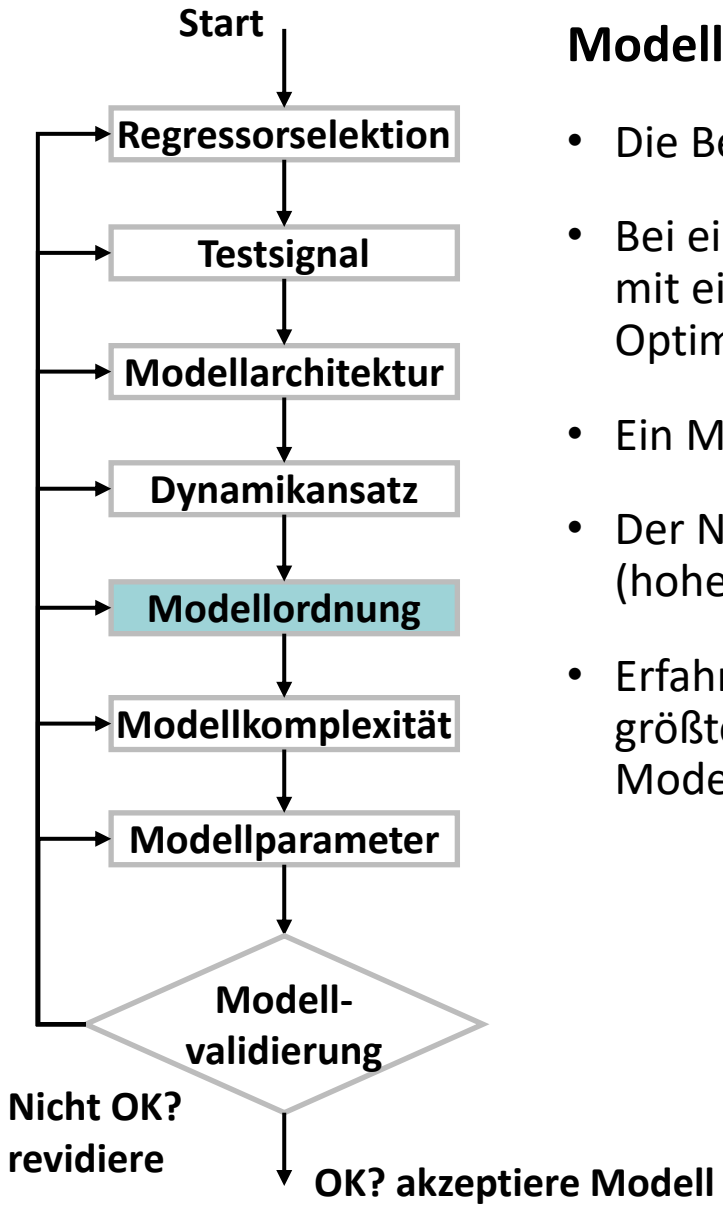


Interner Dynamikansatz

- Der externe Dynamikansatz wird präferiert, da er zu einfacheren Optimierungsproblemen führt.

Designentscheidungen

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

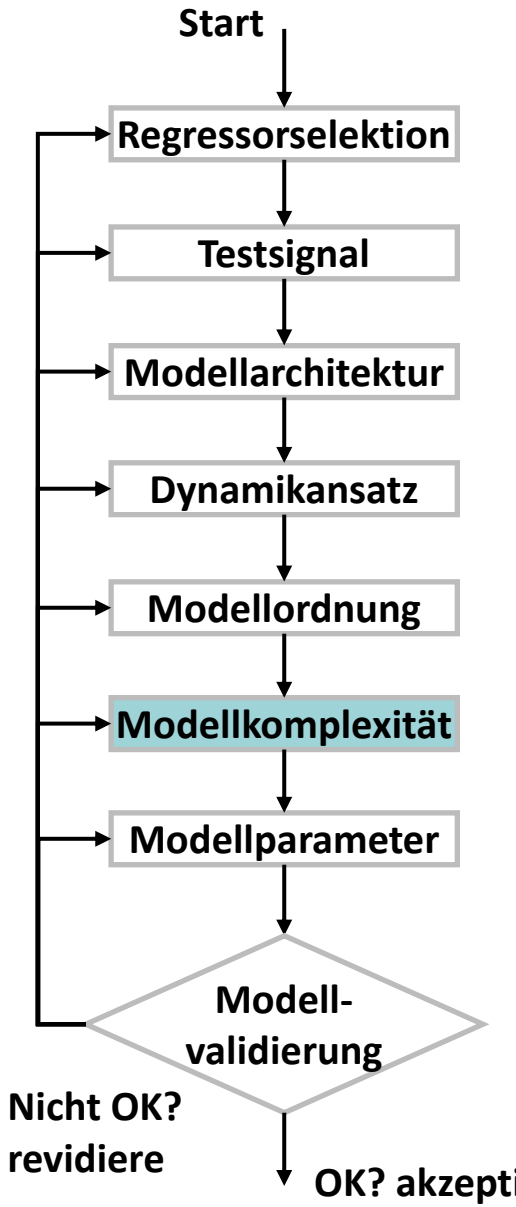


Modellordnung

- Die Bestimmung der Modellordnung geschieht basierend auf Vorwissen sowie Trial and Error
- Bei einem Modell mit externer Dynamik geht eine Erhöhung der Modellordnung zwangsläufig mit einer größeren Anzahl an Modelleingängen einher. Dies erhöht die Komplexität des Optimierungsproblems und die erforderliche Datenmenge (Curse of Dimensionality)
- Ein Modell kann aufgrund des Bias-Varianz-Dilemmas nicht beliebig komplex sein.
- Der Nutzer muss daher einen Kompromiss zwischen der Modellierung dynamischer Effekte (hohe Modellordnung) und Nichtlinearitäten (mächtiger Funktionsapproximator) finden.
- Erfahrungsgemäß sind es oft nicht unmodellierte dynamische Effekte hoher Ordnung, die den größten Modellfehler verursachen, sondern unmodellierte Nichtlinearitäten. Nichtlineare Modelle mit niedriger Ordnung sind daher oft ausreichend.

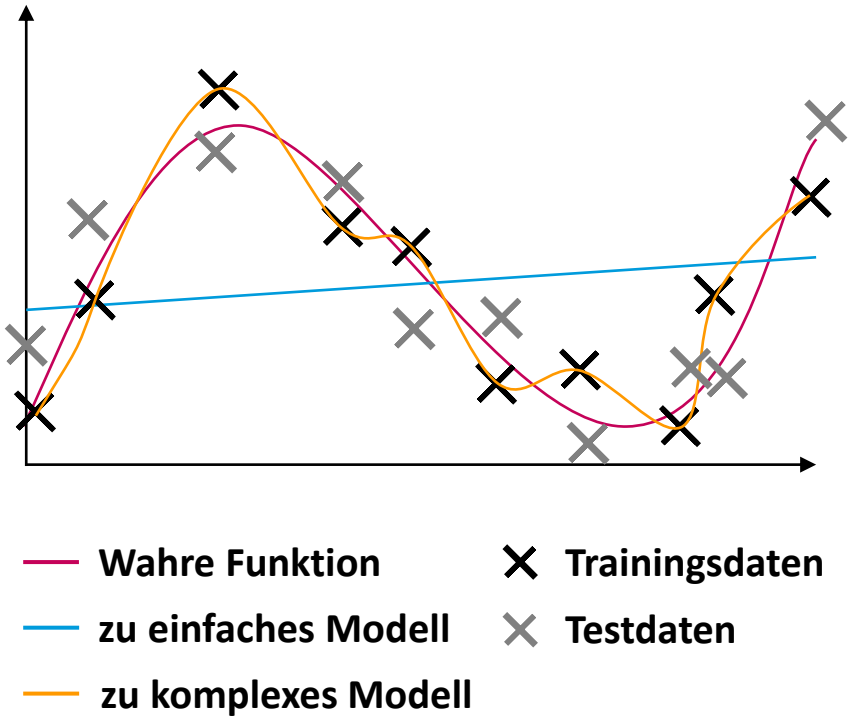
Designentscheidungen

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung



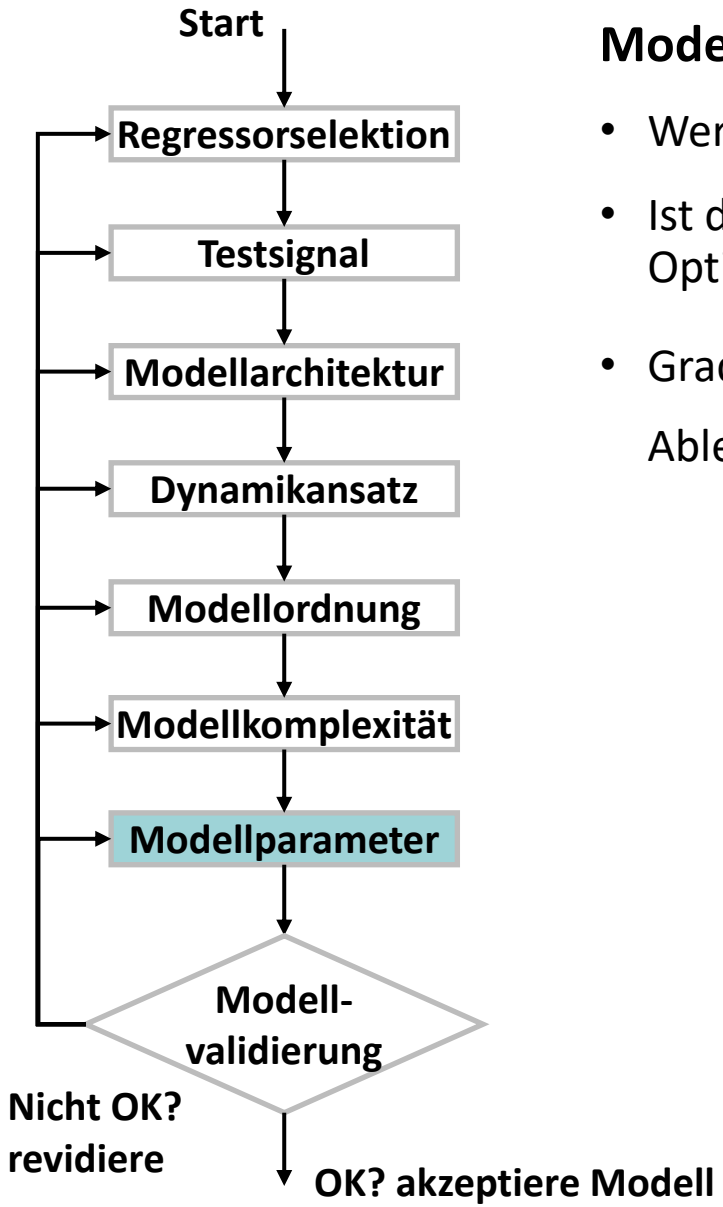
Modellkomplexität

- Die Komplexität eines Modells beschreibt dessen Fähigkeit, bestimmtes Prozessverhalten abbilden zu können
- Ein Polynom 3. Grades ist bspw. komplexer als ein Polynom 2. Grades. Ein Neuronales Netz mit 20 Neuronen ist komplexer als ein NN mit 10 Neuronen in der verdeckten Schicht
- Die Komplexität eines Modells wird für gewöhnlich über die Anzahl an Modell-parametern gemessen
- Ist ein Modell nicht complex genug, kann es das Prozessverhalten nicht akkurat abbilden (underfitting, hoher Biasfehler)
- Ist es hingegen zu komplex, kann es sich ungewollt an stochastische Variationen in den Trainingsdaten anpassen (overfitting, hoher Varianzfehler)



Designentscheidungen

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

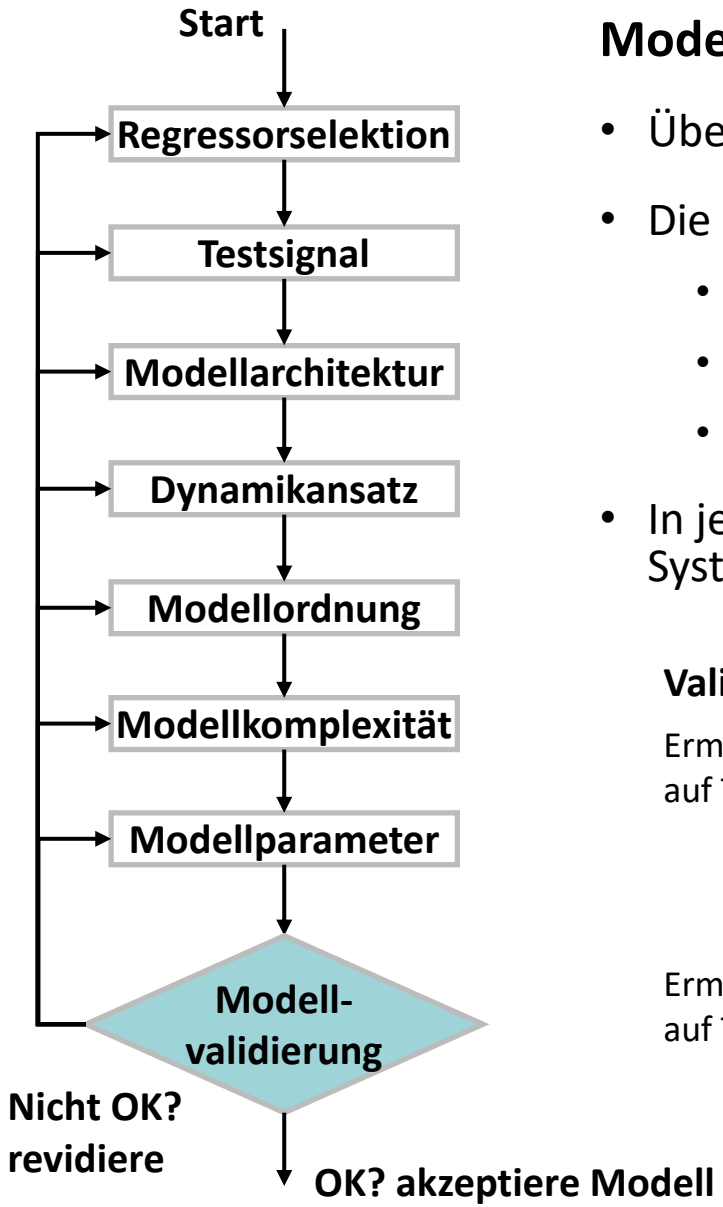


Modellparameter

- Werden durch Anwendung linearer oder nichtlinearer Optimierungsmethoden berechnet
- Ist die zu minimierende Funktion differenzierbar, können gradientenbasierte Optimierungsmethoden angewendet werden.
- Gradientenbasierte Optimierungsverfahren nutzen die erste Ableitung $\boldsymbol{J} = \frac{\partial f}{\partial \theta}$ (und zweite Ableitung $\boldsymbol{H} = \frac{\partial^2 f}{\partial \theta^2}$) um ein lokales Minimum der Kostenfunktion $f(\theta)$ zu finden

Designentscheidungen

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung



Modellvalidierung

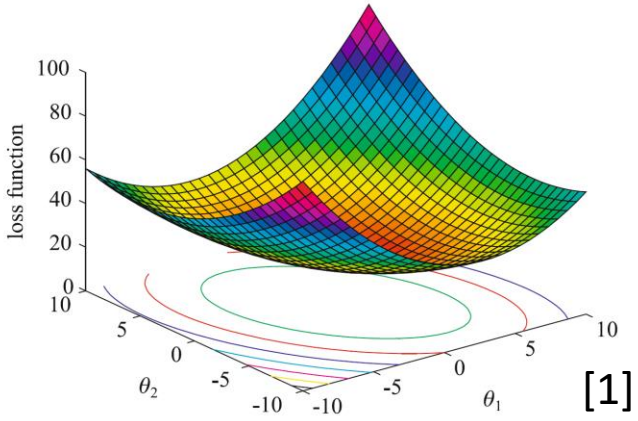
- Überprüfung, ob das resultierende Modell den Anforderungen genügt.
- Die Kriterien sind abhängig von der Verwendung des Modells:
  - Reglerauslegung: Führungsverhalten, Störverhalten.
  - Fehlerdetektion: Anzahl Falschalarme / Fehlalarme
  - ...
- In jedem Fall wird aber die Modellgüte direkt untersucht, um den Schritt der Systemidentifikation von nachfolgenden Schritten soweit wie möglich zu entkoppeln:

Validierungsschritt	Ursachen für geringe Modellgüte
Ermittlung der Modellgüte auf Trainingsdaten	• Das Modell ist nicht fllexibel genug, um die Beziehungen zwischen Eingang und Ausgang abbilden zu können. Die Modellkomplexität muss erhöht werden. • Dem Modell fehlen Informationen, um die Beziehungen zwischen Eingang und Ausgang abbilden zu können, z.B. eine Eingangsgröße. • Das Optimierungsverfahren ist nicht in der Lage ein gutes lokales Minimum zu finden.
Ermittlung der Modellgüte auf Testdaten	• Das Modell ist zu komplex und generalisiert daher schlecht • Die Trainingsdaten beinhalten nicht alle Phänomene, die in den Testdaten enthalten sind (decken nicht denselben Betriebsbereich ab, regen nicht alle Frequenzbereiche an)

Lineare vs. nichtlineare Optimierungsprobleme

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

Lineares Optimierungsproblem



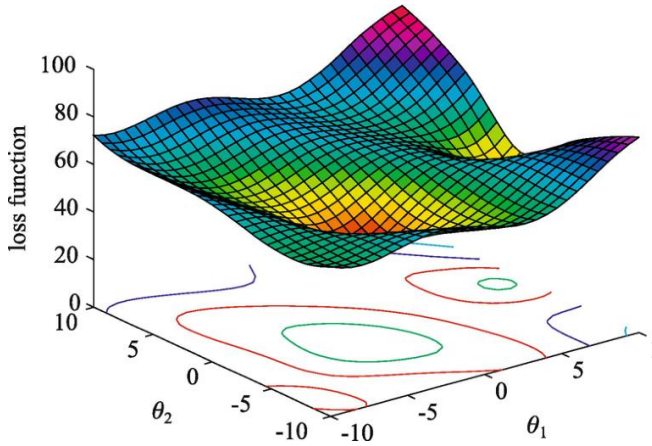
[1]

- Die Kostenfunktion ist eine (Hyper-) Parabel der Form

$$\frac{1}{2}\theta^T H\,\theta + \; g^T\theta + f_0$$

- Es existiert ein eindeutiges Optimum
- Es existiert eine analytische Lösung, die in einem Schritt berechnet werden kann
- Die Lösungsverfahren sind stabil und schnell, sodass eine online Anwendung einfach möglich ist

Nichtlineares Optimierungsproblem



[1]

- Die Kostenfunktion kann in der Nähe eines lokalen Optimums durch eine (Hyper-) Parabel angenähert werden

$$\frac{1}{2}\Delta\theta^T H\,\Delta\theta + \; g^T\Delta\theta + f_0$$

- Es existieren viele lokale Optima
- Es existiert keine analytische Lösung
- Es müssen iterative Lösungsverfahren angewendet werden
- Die Lösungsverfahren sind sehr rechenaufwändig, was eine online Anwendung erschwert.

Lineare Optimierung

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- Gegeben seien N Datenpunkte eines Prozesses mit einer Eingangsgröße $u \in \mathbb{R}$ und einer Ausgangsgröße $y \in \mathbb{R}$

$$D = \begin{bmatrix} u_1 & y_1 \\ \vdots & \vdots \\ u_N & y_N \end{bmatrix}$$

- Es soll ein einfaches statisches Modell mit einem Parameter geschätzt werden

$$\hat{y} = \theta u$$

- Einsetzen der Daten in die Modellgleichung ergibt

$$\hat{y}_1 = \theta u_1$$
$$\hat{y}_2 = \theta u_2$$
$$\dots$$
$$\hat{y}_N = \theta u_N$$

- Zwecks einer kompakteren Schreibweise werden Regressionsmatrix X und Regressandenvektor Y definiert

$$X = \begin{bmatrix} u_1 \\ \vdots \\ u_N \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

- Verwendet wird eine quadratische Kostenfunktion

$$L = \frac{1}{2} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

- Einsetzen der Modellgleichung ergibt

$$L = \frac{1}{2}\left[(\theta u_1 - y_1)^2 + (\theta u_2 - y_2)^2 + \cdots + (\theta u_N - y_N)^2\right]$$

- Ausmultiplizieren und Einsetzen von Regressionsmatrix X und Regressorvektor Y

$$L = \frac{1}{2}\theta\underbrace{\left[X^T X\right]}_{H}\theta - \underbrace{X^T \mathrm{Y}}_{g^T}\theta + \frac{1}{2}\underbrace{\mathrm{Y}^T \mathrm{Y}}_{l_0}$$

- Die Kostenfunktion ist in diesem Fall eine quadratische Funktion. Deren Minimum kann analytisch und in einem Schritt ermittelt werden:

$$\hat{\theta} = \left[X^T X\right]^{-1} X^T \mathrm{Y}$$

- Die angegebene Lösung gilt für alle Modelle, die linear in den Parametern (LiP) sind, d.h. deren Modellgleichungen die Form

$$y = \sum_{j=1}^{n} \theta_j u_j$$

haben.

Nichtlineare Optimierung

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- Ist ein Modell nicht LiP, sondern tauchen die Modellparameter in nichtlinearen Modellgleichungen auf

$$\hat{y} = f(u;\theta)$$

  dann ist die zu minimierende Kostenfunktion L keine quadratische, sondern eine allgemeine nichtlineare Funktion.

- Es existiert dann keine analytische Lösung für das Optimierungsproblem mehr, sodass iterative Optimierungsverfahren angewendet werden müssen.

- Lokale Optimierungsverfahren, wie z.B. das **Newtonverfahren**, approximieren die Kostenfunktion in der Umgebung $\triangle\,\theta_k$ des aktuellen Wertes θ_k als quadratische Funktion

$$L(\theta_k + \triangle\,\theta_k) \approx L(\theta_k) + L'(\theta_k)\,\triangle\,\theta_k + \frac{1}{2}L''(\theta_k)\,\triangle\,\theta_k^2$$

- $L'(\theta_k)$ und $L''(\theta_k)$ sind die erste und zweite Ableitung der Kostenfunktion nach den gesuchten Parametern.

- Die quadratische Gleichung kann nach der Parameteränderung $\triangle\,\theta_k$ gelöst werden, sodass sich der neue Wert für die gesuchten Parameter zu

$$\theta_{k+1} = \theta_k + \triangle\,\theta_k$$

  ergibt.

Nichtlineare Optimierung

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- Oft wird das Newtonverfahren um eine Lernrate / Schrittweite $0 < \gamma < 1$ erweitert

$$\theta_{k+1} = \theta_k + \gamma \triangle \theta_k$$

- Damit soll sichergestellt werden, dass die Parameteränderung $\gamma \triangle \theta_k$ klein genug ist, sodass $L(\theta_{k+1}) < L(\theta_k)$.
- Die lokale Approximation von $L(\theta_k)$ als quadratische Funktion ist umso akkurater, je kleiner die betrachtete Umgebung gewählt wird.
- Nebem dem Newtonverfahren existieren zahlreiche weitere gradientenbasierte Optimierungsverfahren, bspw.

	Beschreibung	**Vorteile**	**Nachteile**
Gradientenverfahren	Nehme Schritte in entgegensetzte Richtung des Gradienten $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \gamma \boldsymbol{J}$	• Einfach zu implementieren • Rechenaufwand pro Iteration sehr gering	• Sehr langsame Konvergenz, falls $f(\theta)$ unterschiedlich stark gekrümmt in Richtungen von $\boldsymbol{\theta}$ • Praktisch von geringer Relevanz
Newtonverfahren	Approximiert $f(\theta)$ lokal als Parabel und berechnet deren Minimum $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \gamma \boldsymbol{H}^{-1} \boldsymbol{J}$	• Schnelle Konvergenz	• Berechnung von $\boldsymbol{H}$ bei großen Problemen sehr aufwändig • $\boldsymbol{H}$ muss invertierbar sein • Divergenz wg. numerischer Instabilität
Quasi Newton Methoden, z.B. Levenberg-Marquardt	Approximation von H und Addition einer skalierten Einheitsmatrix, um Invertierbarkeit sicherzustellen $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \gamma \left[\boldsymbol{J}^T \boldsymbol{J} + \lambda \boldsymbol{I}\right]^{-1} \boldsymbol{J}^T \boldsymbol{f}(\boldsymbol{\theta}_k)$	• Schnelle Konvergenz • Geringerer Rechenaufwand als Newton • Numerisch stabil	• Vielzahl von Strategien zur Wahl von λ

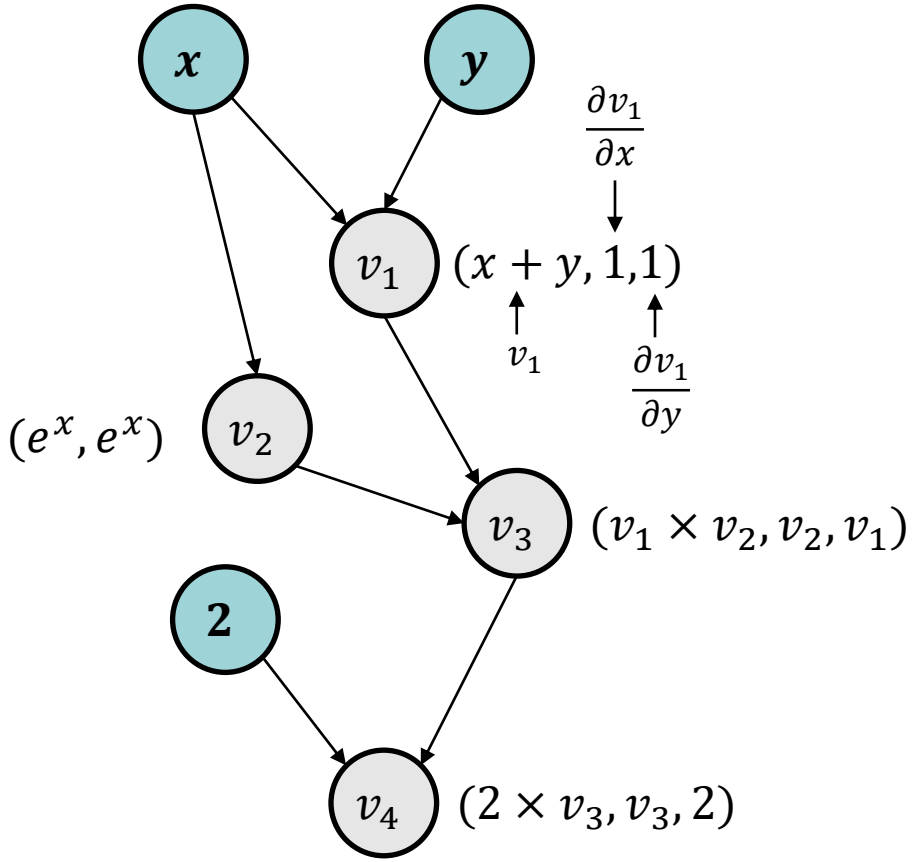
Automatisches Differenzieren

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- Zur effizenten Lösung von Optimierungsproblemen ist die Bildung des Gradienten der Kostenfunktion erforderlich
- Je schneller dieser gebildet werden kann, desto schneller kann das Optimierungsproblem gelöst werden

--> Automatisches Differenzieren!

- Beschreibe die Funktion $f(x)$, bezüglich der die Ableitung gebildet werden soll, als Berechnungsgraphen
- Der Berechnungsgraph beschreibt $f(x)$ als Abfolge von einfachen Rechenoperationen
- Für jede dieser elementaren Rechenoperationen ist die Ableitung bekannt
- Auf Basis des Berechnungsgraphen wird von dem Automatischen Differenzierer (AD) ein neuer Graph zur Berechnung der Jacobi-Matrix gebildet
- Dieses kann bereits bei der Vorwärtsauswertung des Graphen berechnet werden!
- Bsp.: $f(x, y) = 2 \times ((x + y) \times e^x)$
- Durch Einsetzen konkreter Werte für x, y können der Funktionswert und deren Ableitung in einem Schritt berechnet werden!



Was ist Casadi?

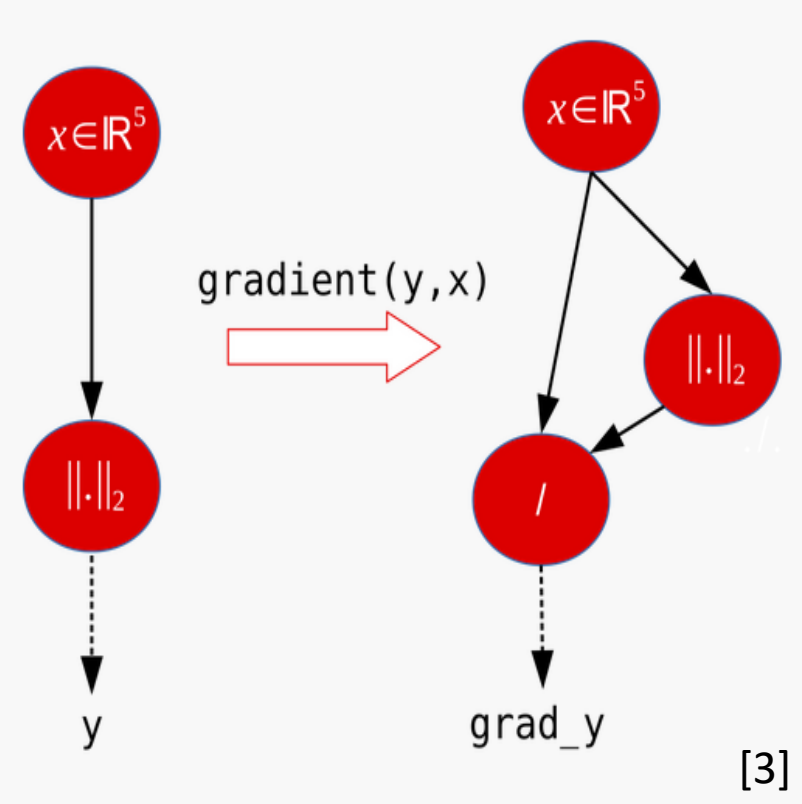
EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- Casadi ist ein symbolisches Framework zur Automatischen Differentiation auf Berechnungsgraphen
- Die Berechnungsgraphen werden in sogenannten Casadi-Funktionen gekapselt
- CasADi-Funktionen können als eigenständiger C-Code exportiert werden

Python Code

```
from casadi import *


# Create scalar/matrix symbols
x = MX.sym('x',5)

# Compose into expressions
y = norm_2(x)

# Sensitivity of expression -> new expression
grad_y = gradient(y,x);

# Create a Function to evaluate expression
f = Function('f',[x],[grad_y])

# Evaluate numerically
grad_y_num = f([1,2,3,4,5]);
```

Berechnungsgraphen



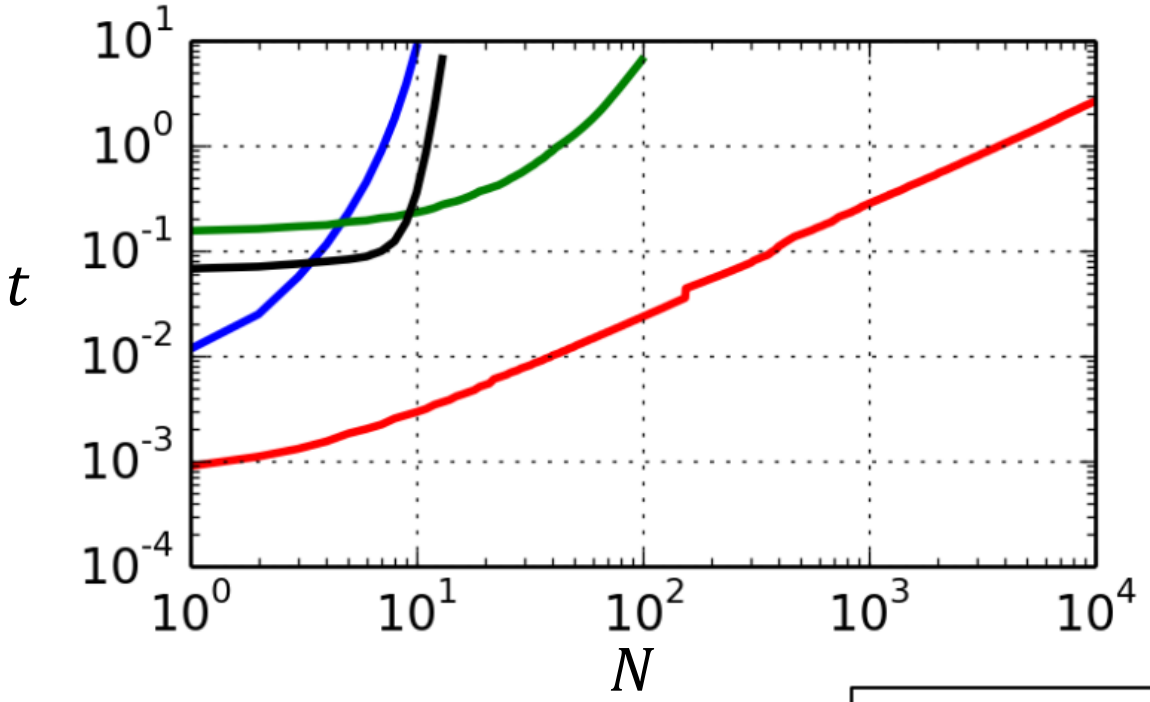
[3]

Warum Casadi?

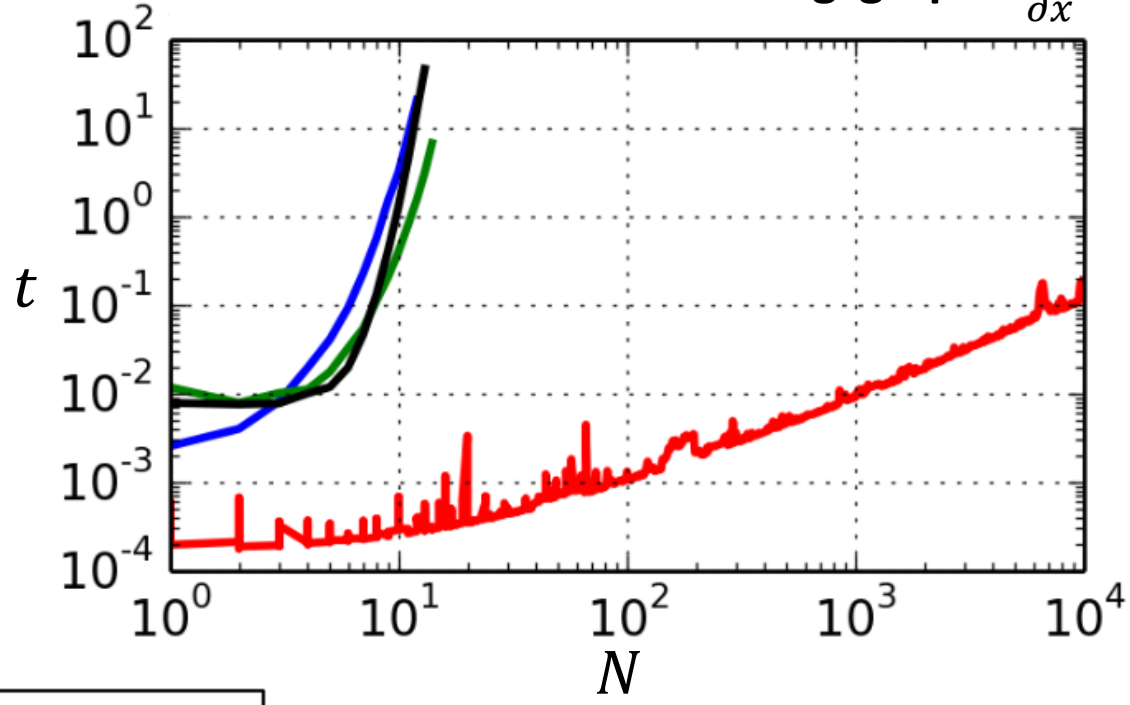
EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- Casadi ist effizienter in der Konstruktion des Berechnungsgraphen der Funktion und deren Ableitung als andere Frameworks

Konstruktionszeit Berechnungsgraph $f(x)$



Konstruktionszeit Berechnungsgraph $\frac{\partial f(x)}{\partial x}$



[2]

- CasADi ist linear in der Auswertung der Berechnungsgraphen (nicht bekannt bei anderen Frameworks aber vermutlich identisch)

Casadi Basics

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- Es gibt zwei verschiedene Symboltypen in CasADi: MX und SX
- MX-Graphen sind speicheroptimiert, während SX-Graphen geschwindigkeitsoptimiert sind

```
import casadi as cs


x = cs.MX.sym('x',1,2)          # Definition einer symbolischen MX Variable x mit Dimension 1x2
y = cs.SX.sym('y',1,2)          # Definition einer symbolischen SX Variable y mit Dimension 1x2
```

- Die beiden Typen sind nicht miteinander kompatibel

```
print(x+x)      -->     MX((2.*x))
print(y+y)      -->     SX([[(y_0+y_0), (y_1+y_1)]])
print(x+y)      -->     Fehler!
```

- Beachte den vereinfachten (und damit speichereffizienteren) Ausdruck der MX Variable gegenüber der SX Variable

Casadi Basics

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- Um einen MX oder SX Graphen für eine Funktion $f()$ aufzubauen, müssen zunächst deren Veränderliche als symbolische MX oder SX Variablen definiert werden, auf denen dann alle entsprechenden Rechenoperationen ausgeführt werden:

```
x = cs.MX.sym('x',1,1)          # Definition einer symbolischen MX Variable x mit Dimension 1x2
y = cs.MX.sym('y',1,1)          # Definition einer symbolischen MX Variable y mit Dimension 1x2

f = 2*((x+y)*cs.exp(x))         # f(x,y) = 2 × ((x + y) × e^x)
```

- Der entstandene Ausdruck für f wird dann in einer CasADi Funktion gekapselt

```
f_fun = cs.Function('f_fun',[x,y],[f],['x', 'y'],['f'])          # Syntax: cs.Function('name',[inputs],[outputs],[names_in], [names_out])
print(f_fun)          -->          Function(f_fun:(x,y)->(x[1x2]) MXFunction)
```

- Die Ein- und Ausgangsgrößen der Funktion zu benennen ist nicht erforderlich, aber empfehlenswert weil
  - man sich bei Aufruf der Funktion keine Gedanken über die Anzahl und Reihenfolge der Argumente machen muss
  - es die Lesbarkeit erhöht und weniger fehleranfällig ist
  - nicht belegte Ein- oder Ausgänge können beim Funktionsaufruf einfach ausgelassen werden (werden automatisch auf Null gesetzt)

Casadi Basics

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- Eine CasADi Funktion wird aufgerufen, indem die Argumente in der richtigen Reihenfolge übergeben werden

  z = f_fun(1.0,2.5) --> DM(19.028)

- Wurden Ein- und Ausgangsgrößen benannt, ist folgender Aufruf möglich

  z = f_fun(x=1.0,y=2.5) --> {'z': DM(19.028)}

  z = f_fun(y=2.5, x=1.0) --> {'z': DM(19.028)}

  die Funktion gibt dann ein Dictionary mit Keys entsprechend den Namen der Ausgangsgrößen aus.

- Wie bei jeder Python Funktion können auch hier der *- und **-Operator verwendet werden

  ```
  inputs = [1.0,2.5]
  z = f_fun(*inputs)
  ```

  ```
  inputs = {'y'=2.5, 'x'=1.0}
  z = f_fun(**inputs)
  ```

Ableiten mit Casadi

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- CasADi bietet vorimplementierte Funktionen zum Bilden von Gradient, Jacobi-Matrix und Hesse-Matrix:

```
x = cs.MX.sym('x',1,1)                  # Definition einer symbolischen MX Variable x mit Dimension 1x2
y = cs.MX.sym('y',1,1)                  # Definition einer symbolischen MX Variable y mit Dimension 1x2

f = 2*((x+y)*cs.exp(x))

f_x = cs.jacobian(f,x)          # Ableitung von f nach x, f_x.shape = (1,1)
f_y = cs. jacobian(f,y)         # Ableitung von f nach y, f_y.shape = (1,1)
f_xx = cs. hessian(f,y)         # Zweite Ableitung von f nach x, f_y.shape = (1,1)
```

- Es existieren auch Funktionen zur Bildung der Ableitungen auf CasADi Funktionen, Best-Practice ist jedoch wie oben die Ableitungen basierend auf CasADi-Ausdrücken zu bilden.
- Nur die Ableitung einer Funktion nach einem Spaltenvektor ist wohldefiniert. Soll nach mehreren Variablen abgeleitet werden, müssen diese deshalb immer als Spaltenvektor rearrangiert werden:

```
f_jac = cs.jacobian(f,cs.vertcat(x,y))      # Ableitung von f nach x und y, f_jac.shape = (1,2) !
f_hess = cs.hessian(f,cs.vertcat(x,y))      # Zweite Ableitung von f nach x und y, f_hess.shape = (2,2) !
```

Ableiten mit Casadi

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- Es ist oft zweckmäßig eine CasADi-Funktion zu definieren, welche den Wert der Funktion und deren Ableitungen in Abhängigkeit der Variablen zurückgibt:

```
f = 2*((x+y)*cs.exp(x))
f_jac = cs.jacobian(f,cs.vertcat(x,y))        # Ableitung von f nach x und y, f_xy.shape = (1,2) !
f_hess = cs.hessian(f,cs.vertcat(x,y))[0]     # Zweite Ableitung von f nach x und y


f_opt = cs.Function('f_opt',[x,y],[f,f_jac,f_hess],['x', 'y'],['f','f_grad','f_hess '])
```

- Handelt es sich bei f_opt um eine zu optimierende Kostenfunktion, kann so für beliebige Punkte (x,y) der Wert der Kostenfunktion und deren Ableitungen ermittelt und an ein Optimierungsverfahren übergeben werden.

Lösung quadratischer Programme mit CasADi

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- Casadi bietet zur Lösung quadratischer Opimierungsprobleme („Programme“) zwei Solver mit gleicher Funktionalität aber unterschiedlichem Interface an: conic und qpsol
- Conic bietet ein Low Level Interface zur Lösung quadratischer Programme der Form

$$\arg\min_x \frac{1}{2} x^T H\, x + \; g^T x$$

- Beispielhaftes Optimierungsproblem:

```
x = cs.MX.sym('x',1,1)              # Zu optimierende Variable
f = 2*x**2 - 3*x +2                 # Zu minimierende Funktion
f_jac = cs.jacobian(f, x)           # Ableitung von f nach x und y, f_xy.shape = (1,2) !
f_hess = cs.hessian(f, x)[0]        # Zweite Ableitung von f nach x, beachte [0], es handelt sich um einen CasADi-Bug
f_opt = cs.Function('f_opt',[x],[f,f_jac,f_hess],['x'],['f','f_grad','f_hess '])
```

- Initialisierung des Solvers

```
qp_struct = {'h': f_hess.sparsity()}        # Der Solver nutzt spärliche Strukturen aus, daher in dict übergeben
S = cs.conic('S','qpoases', qp_struct)      # Syntax: cs.conic('name',solver,dict)
```

Lösung quadratischer Programme mit CasADi

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- Durch den Aufruf des Solvers wird die Lösung des quadratischen Opimierungsproblems automatisch ermittelt und zurückgegeben.
- Erinnerung: f_hess und f_jac sind bisher nur symbolische Größen, es muss ein konkreter numerischer Wert eingesetzt werden, welcher ist egal, weil die Lösung eindeutig und damit nicht initialisierungsabhängig ist.

```
x_0 = 1                    # Zufällig gewählter initialer Wert für x
f,g,H = f_opt(x_0)         # Berechnung von Hesse-Matrix H und Jacobi-Matrix g in x_0
```

- Die Kostenfunktion wird in einer Umgebung $\triangle x$ um x_0 nun beschrieben als

$$f(x) = \frac{1}{2} \triangle x^T \, H \triangle x + \, g^T \triangle x$$

- Aufruf des Solvers durch Übergabe von H und g

```
result = S(h=H,g=g)        # Aufruf des Solvers
dx = result['x']           # Optimierungsergebnis in dictionary unter key 'x' abgelegt
```

- Wichtig: Da die Kostenfunktion in der Nachbarschaft um x_0 beschrieben wurde, muss das finale Optimierungsergebnis noch berechnet werden

```
x_opt = x_0 + dx
```

Lösung quadratischer Programme mit CasADi

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- Ein High-Level Interface bietet qpsol, hier muss der Nutzer nur die Kostenfunktion an den Solver übergeben, und sich sonst um nichts kümmern:

```
x = cs.MX.sym('x',1,1)          # Zu optimierende Variable
f = 2*x**2 - 3*x +2             # Zu minimierende Funktion

qp = {'x':x, 'f':f}             # Definiere zu optimierende Größen 'x' und zu minimierende Funktion 'f'
S = cs.qpsol('S', 'qpoases', qp) # Initialisiere Solver
r=S()                           # Rufe Solver auf

r['x']                          # Optimale Werte für Zielgrößen in dictionary unter key 'x'
```

Lösung nichtlinearer Programme mit CasADi

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- Zur Lösung nichtlinearer Opimierungsprobleme (Programme) kann der Solver conic in einer Iterationsschleife genutzt werden
- Beispielhaftes Optimierungsproblem:

```
x = cs.MX.sym('x',1,1)                  # Zu optimierende Variable
f = 10 + x**2-10*cs.cos(2*cs.pi*x)  # Zu minimierende Funktion
f_jac = cs.jacobian(f, x)               # Ableitung von f nach x und y, f_xy.shape = (1,2) !
f_hess = cs.hessian(f, x)[0]            # Zweite Ableitung von f nach x
f_opt = cs.Function('f_opt',[x],[f,f_jac,f_hess],['x'],['f','f_grad','f_hess '])
```

- Ausgehend von einem vom Nutzer spezifizierten Startwert x_0 kann dieser iterativ verbessert werden

```
x_it = x_0
for i in range(0,10):                   # Führe 10 Iterationen aus (willkürlich gewählt für dieses Beispiel)
    f,g,H = f_opt(x_it)
    result = S(h=H,g=g)                 # Aufruf des Solvers
    dx = result['x']                    # Optimierungsergebnis (Änderung) in dictionary unter key 'x' abgelegt
    x_it = x_it + dx                    # Berechne neuen Wert der Optimierungsgröße
```

Lösung nichtlinearer Programme mit CasADi

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

- Lässt sich die Zielfunktion lokal allerdings nicht gut als quadratische Funktion approximieren (Hesse-Matrix nicht invertierbar oder beinahe singulär), kann es vorkommen, dass die Optimierung mit conic fehlschlägt. Es ist dann empfehlenswert einen dedizierten nichtlinearen Solver zu verwenden

- CasADi bietet den High-Level Solver nlpsol

```
x = cs.MX.sym('x',1,1)                  # Zu optimierende Variable
f = 10 + x**2-10*cs.cos(2*cs.pi*x)      # Zu minimierende Funktion

nlp = {'x':x, 'f':f}                    # Definiere zu optimierende Größen 'x' und zu minimierende Funktion 'f'
S = cs.nlpsol('S', ipopt', nlp)         # Initialisiere Solver
r=S(x0=0.5)                             # Rufe Solver mit Startwert auf

r['x']                                  # Optimale Werte für Zielgrößen in dictionary unter key x
```

Quellenverzeichnis

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

[1] Nelles, Oliver. *Nonlinear system identification: from classical approaches to neural networks, fuzzy models, and gaussian processes*. Springer Nature, 2020.

[2] https://www.syscop.de/files/2015ws/events/Joris%20Gillis%20Talk%2019.07.16.pdf

[3] https://web.casadi.org/