HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN

Studienarbeit

Maschinenbau

Effiziente Bayessche Optimierung durch Berücksichtigung A-priori-Wissens in Form eines physikalischen Dynamikmodells

vorgelegt von

Oliver Schön
Matr.-Nr.: 7026058

Betreuer:

Prof. Dr.-Ing. habil. Ansgar Trächtler
Dr.-Ing. Julia Timmermann
M.Sc. Michael Hesse

Paderborn, 06.12.2019

Studienarbeit

Effiziente Bayessche Optimierung durch Berücksichtigung A-priori-Wissens in Form eines physikalischen Dynamikmodells
am: 06.12.2019

Heinz Nixdorf Institut
Universität Paderborn
Regelungstechnik und Mechatronik
Prof. Dr.-Ing. habil. Ansgar Trächtler
Fürstenallee 11
33102 Paderborn

HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN

Studienarbeit

Effiziente Bayessche Optimierung durch Berücksichtigung A-priori-Wissens in Form eines physikalischen Dynamikmodells

Motivation

Nahezu alle Maschinen und Prozesse unterliegen strengen Zeit-, Qualitäts- und Kostenvorgaben. Die Entwicklung moderner Produkte erfordert daher die zielgerichtete Anpassungsfähigkeit involvierter Systeme. Dem Stand der Technik nach ist hierfür eine zeit- und kostenintensive Einstellung durch den Maschinenbediener notwendig. Zur automatisierten und effizienten Bestimmung der optimalen Einstellparameter soll im Rahmen dieser Arbeit die Bayessche Optimierung aus dem Bereich des maschinellen Lernens eingesetzt werden.

Aufgabenstellung

Das Ziel dieser Arbeit ist es, die Bayessche Optimierung mit einem exemplarischen System zu verknüpfen und einen Vergleich zwischen einer Optimierung mit und ohne Modellwissen bzgl. der Anzahl an benötigten Gütefunktionsauswertungen durchzuführen. Im Vorfeld ist hierfür eine intensive Einarbeitung in die Funktionsweise und Implementierung von Gaußprozessen notwendig. Anschließend soll ein Algorithmus entwickelt und implementiert werden, welcher die Bayessche Optimierung unter Einbeziehung des physikalisch-motivierten Simulationsmodells realisiert. Dabei ist im Hinblick auf die Anwendung eine geeignete Konfiguration zu wählen. Abschließend ist der Algorithmus hinsichtlich Dateneffizienz und Robustheit zu untersuchen und zu analysieren. Dabei sollen zwei Simulationsmodelle mit unterschiedlichen Parametrierungen verwendet werden, wobei ein Modell stellvertretend für das reale System und das andere Modell für die Bayessche Optimierung eingesetzt werden soll.

Die Aufgabenstellung gliedert sich damit wie folgt:

- Einarbeitung in die Funktionsweise und Implementierung von Gaußprozessen
- Literaturrecherche zum Einsatz von Bayesscher Optimierung, insbesondere im Kontext von Einrichtprozessen und in Kombination mit Modellwissen
- Entwicklung und Implementierung von Bayesscher Optimierung mit Modellwissen in MATLAB oder Python
- Simulationsbasierte Analyse und Vergleich der Optimierung mit und ohne Einbringung von Modellwissen

Über Vorgehen und Ergebnisse ist eine schriftliche Ausarbeitung anzufertigen.

Arbeitsaufwand: 450 h	Bearbeiter: Oliver Schön, B.Sc.
Bearbeitungszeit: 6 Monate	Betreuer: Michael Hesse, M.Sc.

Beginn der Bearbeitung: 01.07.2019

Prof. Dr.-Ing. habil. Ansgar Trächtler

Erklärung:

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne unerlaubte fremde Hilfe angefertigt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Paderborn, 06.12.2019

Zusammenfassung

Bayessche Optimierung ist ein mächtiges Werkzeug zur Lösung komplexer Optimierungsprobleme, z.B. im Rahmen von Einrichtprozessen. Besonders dann, wenn die Erhebung von Daten mit hohem Zeit- und Kostenaufwand verbunden ist, ermöglicht Bayessche Optimierung die automatisierte Einstellung grundlegender Parameter. Dabei erfolgt die Optimierung von Prozessen und Anlagen meist ohne jegliches Systemverständnis in Form einer Black-Box-Optimierung. Im Rahmen technischer Systeme liegt in der Regel jedoch Vorwissen in Form eines physikalischen Dynamikmodells vor, das u.U. zu einer gesteigerten Effizienz des Optimierungsverfahrens führen könnte. Um diesen Zusammenhang näher zu untersuchen, wird anhand eines Anwendungsbeispiels der Effekt A-priori-Wissens im Rahmen des Optimierungsprozesses genauer beleuchtet. Es kann gezeigt werden, dass es bei der Berücksichtigung hinreichend genauen Vorwissens zu einer deutlichen Effizienzsteigerung des Bayesschen Optimierers kommt. Dabei spielt neben der Konditionierung des Optimierungsproblems auch besonders die Dimension des Problems eine Rolle. Im Allgemeinen sollte aber nicht auf die Nutzung A-priori-Wissens verzichtet werden, da bereits durch Vorgabe grober Zusammenhänge die Zahl der benötigten Iterationen reduziert werden kann und es durch die Einbringung fehlleitender Informationen nur in besonderen Fällen zu Performanceeinbußen kommt.

Abstract

Bayesian Optimization is a powerful tool for providing solutions to complex optimization problems, e.g. as part of setup processes. It enables the automated setting of basic tuning parameters, especially when the collection of data is very time- and cost-intensive. The optimization of both processes and devices usually takes place without any understanding of the regarding systems in the form of a black box optimization. In the context of technical systems, however, prior knowledge is typically available in the form of a physical dynamics model, which could potentially result in greater efficiency of the optimization process. In order of investigating this connection in more detail, the effect of a priori knowledge in the context of the optimization process is examined on the basis of an exemplary application. It can be shown that the utilization of sufficiently accurate prior knowledge results in a significant increase in the efficiency of the Bayesian optimizer. In addition to the conditioning of the optimization problem, the dimension of the problem also plays a particularly important role. In general, however, the use of a priori knowledge should not be neglected, since the number of required iterations can already be reduced by specifying rough baselines and performance losses only occur in special cases due to the introduction of misleading information.

Effiziente Bayessche Optimierung durch Berücksichtigung A-priori-Wissens in Form eines physikalischen Dynamikmodells

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Motivation 1
- 1.2 Zielsetzung 2
- 1.3 Vorgehensweise 2

2 Grundlagen **3**
- 2.1 Wahrscheinlichkeitstheorie 3
  - 2.1.1 Rechenregeln 3
  - 2.1.2 Wahrscheinlichkeitsverteilungen 6
- 2.2 Optimierung 11
- 2.3 Energiebasierte Regelung dynamischer Systeme 12
- 2.4 Regression 14
- 2.5 Gauß-Prozess-Regression 17
  - 2.5.1 Mittelwert und Kovarianz 20
  - 2.5.2 Kernel-Funktionen 22
  - 2.5.3 Interpretation der Hyperparameter 24
  - 2.5.4 Bestimmung der Hyperparameter 27

3 Bayessche Optimierung **31**
- 3.1 Algorithmus 33
- 3.2 Akquisitionsfunktionen 34
  - 3.2.1 Upper/Lower Confidence Bound (UCB/LCB) 34
  - 3.2.2 Max-value Entropy Search (MES) 34

4 Anwendungsbeispiel: Aufschwung eines starren Pendels **37**
- 4.1 Physikalische Modellbildung 37
- 4.2 Steuerungsentwurf 38
- 4.3 Akquisitionsfunktion 40
  - 4.3.1 Gradient der Akquisitionsfunktion 40
  - 4.3.2 Zustandsdiskretisierung 41
- 4.4 Untersuchungen 42
  - 4.4.1 Eindimensionaler Fall 42
  - 4.4.2 Dreidimensionaler Fall 46
  - 4.4.3 Effekt von Ausreißern 49

5 Zusammenfassung und Ausblick **51**
5.1 Stückweise Glättung der Kovarianz 52
5.2 Adaptive Akquisitionsfunktionen 52
5.3 Gewichteter Prior 53
5.4 Parameteridentifikation 53

Literaturverzeichnis **55**

Anhang

A1 Systemverhalten des starren Pendels **59**

A2 Code **63**
A2.1 Main 63
A2.2 Prozedur 64
A2.3 Bestimmung der Hyperparameter 66
A2.4 Akquisitionsfunktion 68
A2.5 Prior 69

Abkürzungsverzeichnis

AQF	Akquisitionsfunktion
ARD	Automatic Relevance Determination
BO	Bayessche Optimierung
CDF	kumulierte Verteilungsfunktion (engl. cumulative distribution function)
EI	Expected Improvement
ES	Entropy Search
GP	Gaußprozess
HP	Hyperparameter
i.i.d.	unabhängig, identisch verteilt (engl. independent, identically distributed)
LCB	Lower Confidence Bound
M52	Matern-52
MES	Max-value Entropy Search
MESmin0	Max-value Entropy Search mit Minimum 0
ML	Maschinelles Lernen
PDF	Wahrscheinlichkeitsdichtefunktion (engl. probability density function)
PES	Predictive Entropy Search
PI	Probability of Improvement
SQExp	Squared-Exponential
UCB	Upper Confidence Bound

Symbolverzeichnis

Name	Beschreibung	Einheit
E	Energie	J
F	Kugelfarbe	–
J	Gütemaß	–
P, p	Wahrscheinlichkeit	–
T	Zeit	s
U	Urne	–
Δ	Intervallgröße	–
Ω	Menge aller θ bzw. $\boldsymbol{\theta}$	–
Ψ	kumulierte Normalverteilung	–
β	Präzision	–
$\boldsymbol{E}$	Einheitsmatrix	–
$\boldsymbol{L}$	untere Dreiecksmatrix	–
$\boldsymbol{\Lambda}$	Inverse der Kernel-Matrix	–
$\boldsymbol{\Sigma}$	Kovarianzmatrix	–
δ	Kronecker-Delta	–
$\mathbb{R}$	Menge der reellen Zahlen	–
$\mathcal{D}$	Daten	–
$\mathcal{N}, \Phi$	Normalverteilung	–
$\mathcal{W}$	Menge aller w bzw. $\boldsymbol{w}$	–
$\mathfrak{R}$	Region	–
$\mathfrak{X}$	Menge unstetiger Stellen	–
μ^+	Posterior-Mittelwert	–
μ^-	Prior-Mittelwert	–
μ	Mittelwert	–
ω	Winkelgeschwindigkeit	$\mathrm{rad\,s^{-1}}$
ϕ	Winkel	rad
π	Kreiszahl	–
σ^2	Varianz	–
σ_f^2	Signal Variance	–
σ_n^2	Noise Variance	–
σ	Standardabweichung	–
$\theta, \boldsymbol{\theta}$	Optimierungsparameter	–
ε	Zufallsvarible; i.i.d. Rauschen	–
$f, \boldsymbol{f}$	Funktionswert; Funktion	–
g	Erdbeschleunigung	$\mathrm{m\,s^{-2}}$
h	Schrittweite	s
k	Designparameter	–
$l_i, \boldsymbol{l}$	Lengthscale-Parameter	–
l	Länge des Pendels	m
m	Masse des Pendels	kg
u	Stellgröße	–

1 Einleitung

1.1 Motivation

In der Zeit von Big Data, Condition Monitoring und einer durch Industrie 4.0 global vernetzten Industrielandschaft wird die Erhebung und Verarbeitung von Daten immer bedeutsamer. Durch Auswertung gesammelter Daten können fundiert Entscheidungen gefällt und Prozesse optimiert werden. Dabei ermöglichte gerade das Aufkommen von Maschinellem Lernen eine effiziente Analyse der anfallenden Datenmengen. In vielen Disziplinen gilt es jedoch auch Aussagen trotz geringer Datenverfügbarkeit zu treffen. Ist die Erhebung der Daten beispielsweise mit hohem Aufwand in Form von Zeit und Kosten verbunden, ist eine sinnvolle Nutzung nur im kleinen Rahmen wirtschaftlich. Besonders in diesen Fällen ist eine effiziente Verarbeitung der Daten essentiell (vgl. [DLvht]). Dafür müssen für große Teile der relevanten Daten Prädiktionen gemacht werden.

Gerade bei der Einrichtung von Prozessen und Maschinen wird Maschinelles Lernen nur spärlich eingesetzt. Viele Anlagen erfordern die manuelle Einstellung durch Fachpersonal, da eine hinreichend genaue Modellierung des Systemverhaltens in der Regel nicht in Betracht kommt. Daher werden besonders komplexe Systeme, bei denen das simple Testen von Einstellungen der Bildung eines Systemverständnisses überlegen ist, als Black-Box betrachtet. Infolge stetig steigender Anforderungen an Produkte und der daraus resultierenden Komplexität der Systeme wurde Black-Box-Optimierung in den letzten Jahren immer bedeutsamer. Da die erhobenen Daten insbesondere einer gewissen Ungenauigkeit unterliegen, ist ohnehin keine exakte Modellierung technischer Systeme möglich und der Einsatz eines Prädiktionsverfahrens erforderlich. [GSM+17]

Ein bekanntes Verfahren zur automatisierten Modellierung und Einstellung von Prozessen ist das Optimierungsverfahren der Bayesschen Optimierung. Dabei erfolgt ein selbstständiges Lernen des Systemverhaltens hinsichtlich einer optimalen Einrichtung des Prozesses. Anhand stichprobenartiger Auswertungen des betrachteten Systems werden Prädiktionen über das gesamte Systemverhalten angestellt. Der Einsatz Bayesscher Optimierung ist allerdings nur dann sinnvoll, wenn die folgenden drei Bedingungen erfüllt sind. Zum einen sollte sich das zu untersuchende System nicht in geschlossener Form darstellen lassen und es sollten keine Informationen über den Gradienten vorliegen. Die Betrachtung des Systems als Black-Box ist somit gerechtfertigt. Darüber hinaus sind besonders die Systeme interessant, deren Auswertung mit großem Aufwand verbunden sind oder die nur durch verrauschte Messdaten beobachtbar sind. Diese Vorgaben gelten insbesondere für die Automatisierung von Einrichtprozessen. [Hea17]

Es existieren bereits diverse Anwendungen Bayesscher Optimierung im Rahmen von Einrichtprozessen. So nutzen z.B. Frameworks von Rechnerverbundsystemen die Anwendbarsowie Skalierbarkeit auf komplexe Systeme zur Einstellung wesentlicher Konfigurationsparameter (vgl. [FGB15]). Sowohl die Steigerung der Qualität von Produkten, als auch der produktivere Einsatz von Human Resources wird durch den Einsatz Bayesscher Optimierung realisiert. Dabei geht es oft lediglich um die Optimierung einzelner Prozessparameter

und die Automatisierung von Prozessen durch Ausschluss der „Einflussgröße Mensch[1]“ (vgl. [SSW+16]). Doch bei keiner der betrachteten Anwendungen wird Gebrauch von vorhandenem Vorwissen über das zu optimierende System gemacht. Obwohl es sich bei der Bayesschen Optimierung de facto um ein Black-Box-Optimierungsverfahren handelt, ist es möglich, dabei Vorwissen einfließen zu lassen. Gerade im Rahmen technischer Systeme liegt in vielen Fällen Vorwissen in Form eines physikalischen Dynamikmodells vor. Obwohl es durch die Berücksichtigung zu einer signifikanten Steigerung der Effizienz des Algorithmus kommen könnte, wurde eine derartige Konfiguration noch nie untersucht.

1.2 Zielsetzung

Im Rahmen dieser Arbeit soll der Einfluss von Vorwissen auf das Konvergenzverhalten der Bayesschen Optimierung technischer Systeme untersucht werden. Dabei ist besonders vorhandenes Wissen in Form eines physikalischen Dynamikmodells interessant. Es gilt weiterführend, die dabei wirkenden Vorgänge vollständig zu erschließen und mögliche Einflussfaktoren zu identifizieren, die Auswirkung auf die Effizienz des Algorithmus haben können. Im Zuge dessen soll eine geeignete Implementierung der betrachteten Inhalte in Matlab erfolgen und anschließend in Form eines Optimierers vorliegen.

1.3 Vorgehensweise

Zunächst sollen die für die Bayessche Optimierung relevanten Grundlagen erarbeitet werden. Das umfasst neben der Wahrscheinlichkeitstheorie auch die Grundlagen der Optimierung und Funktionsregression. Darauf aufbauend sollen erste Implementierungen der Gauß-Prozess-Regression in Matlab erfolgen. Aufgrund des baukastenprinzipartigen Aufbaus soll in diesem Zug ebenfalls der Einfluss der Wahl diverser Komponenten und die Funktionsweise eben dieser genau betrachtet werden. Dies ist insbesondere hinsichtlich einer geeigneten Komponentenwahl für die Implementierung der Bayesschen Optimierung von Bedeutung, die im nächsten Schritt genauer analysiert werden soll. Im Anschluss soll der implementierte Optimierungsalgorithmus hinsichtlich diverser Fragestellungen untersucht werden. So soll im ersten Schritt zunächst nur ein eindimensionales Optimierungsproblem anhand eines adäquaten Anwendungsbeispiels behandelt werden. Daraus sollen Rückschlüsse auf den Einfluss von Vorwissen in Form eines physikalischen Dynamikmodells auf das Konvergenzverhalten des Algorithmus gezogen werden. Die vorerst eindimensionale Betrachtung soll in diesem Zuge ein besseres Verständnis der dabei auftretenden Effekte ermöglichen. Erst im Anschluss ist ein multidimensionales Anwendungsbeispiel zu untersuchen. Durch die gesammelten Ergebnisse sind finale Schlussfolgerungen für das Optimierungsverhalten allgemeiner technischer Systeme hinsichtlich der Berücksichtigung A-priori-Wissens zu ziehen.

[1] engl. human factor

2 Grundlagen

Die Grundlagen, die zum Verständnis der Bayesschen Optimierung erforderlich sind, sind umfangreich. Daher werden in diesem Kapitel die grundlegendsten Elemente aufgegriffen und erläutert. Dazu zählen die Grundlagen der Wahrscheinlichkeitstheorie, sowie ein grober Überblick über Regressionsprobleme, die Optimierung und den Steuerungsentwurf für dynamische Systeme. Darüber hinaus wird insbesondere auf die Gauß-Prozess-Regression eingegangen.

2.1 Wahrscheinlichkeitstheorie

Machine Learning[2] (ML) wird genutzt, um Muster in gesammelten Daten zu erkennen. So können z.B. auf Basis umfangreicher Patientendaten, wie CT-Daten[3], Diagnosen gemacht werden. Dabei treten Unsicherheiten auf, die aufgrund der endlichen Menge an Daten und dem Messrauschen aufkommen. Die Unsicherheiten werden in Form von Wahrscheinlichkeiten ausgedrückt. Um optimale Aussagen infolge der gesammelten Informationen treffen zu können, ist daher ein Verständnis der inbegriffenen Wahrscheinlichkeiten und der damit verbundenen Regeln notwendig. Die nachfolgenden Ausführungen folgen [Bis06].

2.1.1 Rechenregeln

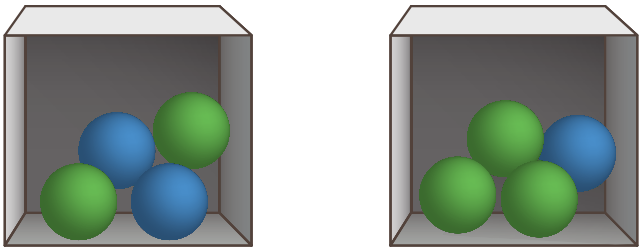


Bild 2-1: *Die beiden Wahlurnen A und B enthalten jeweils Kugeln mit definiertem Anteil grüner und blauer Kugeln.*

Zur Veranschaulichung der nachfolgenden Inhalte soll vorab ein Beispiel eingeführt werden. Es werden dazu zwei Wahlurnen betrachtet, die jeweils eine gewisse Anzahl blauer und grüner Kugeln beinhalten (Abb. 2-1). Der Anteil blauer Kugeln entspricht für Urne A 50% und für Urne B 25%. Im Rahmen einer Ziehung wird nun vorab eine der beiden

[2] dt. Maschinelles Lernen
[3] Computertomografie: bildgebendes Verfahren in der Medizintechnik

Urnen gewählt und danach aus selbiger eine Kugel gezogen. Nach der Ziehung wird die Kugel wieder in die entsprechende Urne zurückgelegt. Sowohl die Wahl der Urnen als auch der Kugeln erfolgt dabei zufällig. Würde dieser Ablauf unendlich oft wiederholt, so würde die Wahl gleich oft auf Urne A und B fallen. Die Wahrscheinlichkeit p für ein Ereignis x, z.B. die Wahl einer grünen Kugel, kann dabei nicht kleiner null werden

$$p(x) \geq 0, \quad \forall x \in X,$$

und die Summe der Wahrscheinlichkeiten aller möglichen Ereignisse, z.B. die Wahl einer grünen oder blauen Kugel ergibt immer eins bzw. 100%.

$$\sum_{x \in X} p(x) = 1$$

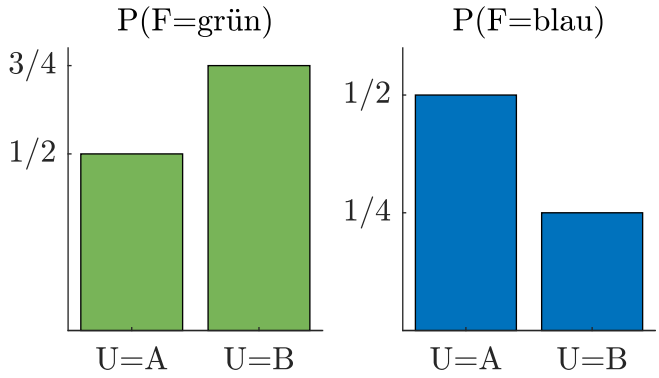


Bild 2-2: *Die Wahrscheinlichkeit für die Ziehung einer grünen oder blauen Kugel ist abhängig von der Wahl der Urne U. Es ergibt sich die diskrete Wahrscheinlichkeitsverteilung für die Ziehung der jeweiligen Farbe F.*

Produktregel der Wahrscheinlichkeitstheorie

Die Wahrscheinlichkeit, mit der eine blaue Kugel aus Urne A gezogen wird, ergibt sich aus der Produktregel der Wahrscheinlichkeit. Ist $F \in \{\text{grün,blau}\}$ die Farbe der gezogenen Kugel und $U \in \{\text{A,B}\}$ die gewählte Urne (siehe dazu Abb. 2-2), so kann die Wahrscheinlichkeit einer gezogenen blauen Kugel aus Urne A berechnet werden durch

$$P(U = \mathsf{A}, F = \mathsf{blau}) = P(F = \mathsf{blau}|U = \mathsf{A}) \cdot P(U = \mathsf{A}).$$

Werden nun nur all diejenigen blauen Kugeln betrachtet, die bei oft wiederholter Ziehung aus Urne A gezogen werden, so ergibt sich die Wahrscheinlichkeit mit der eine gezogene blaue Kugel aus Urne A stammt. Dies wird als *bedingte Wahrscheinlichkeit* bezeichnet - $p(F|U)$ ist „die Wahrscheinlichkeit von F gegeben U“.

Summenregel der Wahrscheinlichkeitstheorie

Soll die Wahrscheinlichkeit für die Wahl einer grünen Kugel, ungeachtet der im Zuge dessen gewählten Urne, bestimmt werden, so handelt es sich um eine *marginale Wahr-*

scheinlichkeit. Die Wahrscheinlichkeit, dass eine grüne Kugel gezogen wird ergibt sich durch Summation über alle möglichen Urnen U:

$$P(F = \text{grün}) = \sum_{U \in \{\text{A,B}\}} p(F = \text{grün}, U).$$

Dabei spricht man bei der *Verbundwahrscheinlichkeitsverteilung*[4] $p(F, U)$ einer gezielten Ziehung einer Kugel mit Farbe F aus Urne U von „der Wahrscheinlichkeit von F und U“.

Satz von Bayes

Unter Zuhilfenahme der Symmetrie-Regel $p(U, F) = p(F, U)$ folgt durch Anwendung der Produktregel der Satz von Bayes:

$$\begin{aligned} p(U, F) = p(F|U) \cdot p(U) &\overset{!}{=} p(U|F) \cdot p(F) = p(F, U) \\ \Leftrightarrow p(F|U) &= \frac{p(U|F) \cdot p(F)}{p(U)}. \end{aligned}$$

Der Nenner kann als Normalisierungskonstante betrachtet werden, indem $p(U)$ durch Summen- und Produktregel ausgedrückt wird als

$$p(U) = \sum_{F \in \{\text{blau,grün}\}} p(U|F) \cdot p(F).$$

Da der Normalisierungsfaktor in der Regel gar nicht oder nur sehr aufwändig bestimmt werden kann, wird der Satz von Bayes häufig umformuliert zu der Proportionalitätsbeziehung

$$p(F|U) \propto p(U|F) \cdot p(F).$$

Im Rahmen der Wahrscheinlichkeitstheorie gibt es zwei differenzierte Ansätze zur Beschreibung von Wahrscheinlichkeiten. Der *frequentistische Ansatz* beschreibt Wahrscheinlichkeiten als Frequenz wiederholter Ereignisse. Vorangegangene Betrachtungen basieren auf diesem Ansatz. *Bayessche Ansätze* hingegen charakterisieren Wahrscheinlichkeit als Maß der Sicherheit oder Unsicherheit, mit der gewisse Ereignisse eintreten werden. Dabei werden vorab Annahmen über das Verhalten der untersuchten Systeme getroffen - das sogenannte *A-Priori-Wissen*. Bisher wurde die Urnenziehung lediglich als Black-Box-System betrachtet. Der Unterschied der beiden Ansätze wird jedoch bereits bei einem einfachen Beispiel deutlich. Wird eine einmalige Ziehung durchgeführt, so ist die gezogene Kugel entweder blau oder grün und stammt aus Urne A oder B. Angenommen eine grüne Kugel wird aus Urne A gezogen und es werden auf Basis der einmaligen Auswertung Wahrscheinlichkeiten für die Ziehung von blauen oder grünen Kugeln sowie Wahl von Urne A und B aufgestellt, so ist die Auftrittswahrscheinlichkeit für grüne Kugeln und Urne A 1 und für blaue Kugeln und Urne B 0. Der Bayessche Ansatz hingegen bezieht

[4] engl. joint probability distribution

Vorwissen über das System in die Betrachtung ein. Die bedingte Auftrittswahrscheinlichkeit der unterschiedlichen Kugeln bezogen auf die beiden Urnen ist (siehe dazu auch Abb. 2-2)

$$P(F = \text{blau}|U = \text{A}) = 1/2,$$
$$P(F = \text{grün}|U = \text{A}) = 1/2,$$
$$P(F = \text{blau}|U = \text{B}) = 1/4,$$
$$P(F = \text{grün}|U = \text{B}) = 3/4.$$

Daraus folgt die über beide Urnen marginalisierte Wahrscheinlichkeit für die Ziehung einer grünen Kugel:

$$\begin{aligned} P(F = \text{grün}) &= P(F = \text{grün}|U = \text{A}) \cdot P(U = \text{A}) + P(F = \text{grün}|U = \text{B}) \cdot P(U = \text{B}) \\ &= 1/2 \cdot 1/2 + 3/4 \cdot 1/2 \\ &= 5/8. \end{aligned}$$

Mithilfe des Satzes von Bayes lässt sich nun die Wahrscheinlichkeit berechnen, mit der eine gezogene grüne Kugel aus Urne A stammt:

$$P(U = \text{A}|F = \text{grün}) = \frac{P(F = \text{grün}|U = \text{A}) \cdot P(U = \text{A})}{P(F = \text{grün})} = \frac{1/2 \cdot 1/2}{5/8} = 2/5.$$

Im Gegensatz zum frequentistischen Ansatz liefern bayessche Ansätze auch bei geringer Datenverfügbarkeit gute Ergebnisse durch die Berücksichtigung von A-Priori-Informationen. Dieser Zusammenhang wird im Rahmen Bayesscher Optimierung genutzt, um dateneffiziente Prädiktionen zu generieren (siehe Kap. 3).

Zusammenfassung: Rechenregeln für den diskreten Fall

Positivität:	$p(x) \geq 0 \quad \forall x \in X$
Normierung:	$\sum_{x \in X} p(x) = 1$
Produktregel:	$p(x, y) = p(y\|x) \cdot p(x)$
Summenregel:	$p(x) = \sum_{y \in Y} p(x, y)$
Satz von Bayes:	$p(y\|x) = \frac{p(x\|y) \cdot p(y)}{p(x)}$

2.1.2 Wahrscheinlichkeitsverteilungen

Die im vorangegangenen Kapitel beschriebenen Regeln der Wahrscheinlichkeit lassen sich vom diskreten Fall auf den kontinuierlichen übertragen. Dabei weist die nun kontinuierliche Größe x Eigenschaften auf, die es ferner zu beachten gilt. Die Funktion $p(x)$

spannt im kontinuierlichen Fall eine *Wahrscheinlichkeitsdichte* (PDF)[5] über x auf. Der Unterschied zur diskreten Wahrscheinlichkeitsverteilung wird deutlich, sobald die Wahrscheinlichkeit für das Auftreten eines konkreten $x \in X$ bestimmt werden soll. Im diskreten Fall ist die Variable x in Intervalle Δ untergliedert. Da die kontinuierliche Wahrscheinlichkeitsdichteverteilung unendlich viele Funktionswerte x überspannt, kann dies als Grenzfall $\Delta \to 0$ betrachtet werden. Die Wahrscheinlichkeit $P(x)$ für ein einzelnes, ganz konkretes $x \in X$ wird gleich null und bezüglich vorangegangener Wahrscheinlichkeitsregeln werden aus Summen Integrale. Um die Wahrscheinlichkeit zu bestimmen, mit der x in einem Intervall $[a, b]$ liegt, wird die Wahrscheinlichkeitsdichtefunktion über $x \in [a, b]$ integriert (Abb. 2-3):

$$P(x \in [a, b]) = \int_a^b p(x)\,\mathrm{d}x.$$

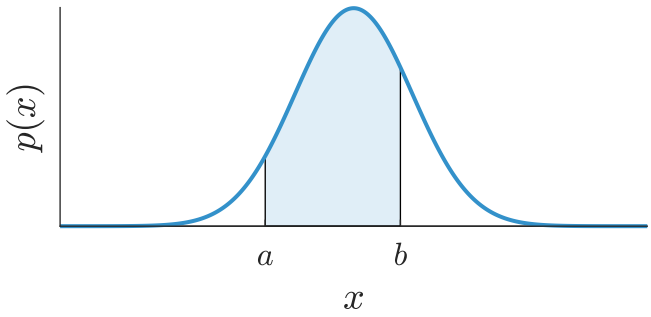


Bild 2-3: *Um die Wahrscheinlichkeit von $x \in [a, b]$ zu bestimmen, gilt es über die zugehörige Fläche unter der Wahrscheinlichkeitsdichteverteilung zu integrieren.*

Grundsätzlich gilt auch im kontinuierlichen Fall, dass Wahrscheinlichkeiten größer gleich null sind und die Gesamtheit aller möglichen Ereignisse 1 ergeben muss:

$$p(x) \geq 0, \quad \forall x \in X,$$

$$\int_{-\infty}^{\infty} p(x) = 1.$$

Die Regeln der Wahrscheinlichkeitsrechnung geben sich entsprechend wie folgt:

$$\text{Produktregel:} \quad p(x, y) = p(y|x) \cdot p(x)$$

$$\text{Summenregel:} \quad p(x) = \int_{y \in Y} p(x, y)\,\mathrm{d}y$$

$$\text{Satz von Bayes:} \quad p(y|x) = \frac{p(x|y) \cdot p(y)}{p(x)}$$

[5] engl. probability density function

Soll die Wahrscheinlichkeit berechnet werden, mit der ein Funktionswert x kleiner als ein Grenzwert z ist, spricht man von einer *kumulierten Wahrscheinlichkeit* $P(x \leq z)$, die durch die kumulierte Verteilungsfunktion (CDF)[6] (Abb. 2-4) beschrieben wird:

$$P(x \leq z) = \int_{-\infty}^{z} p(x)\,\mathrm{d}x.$$

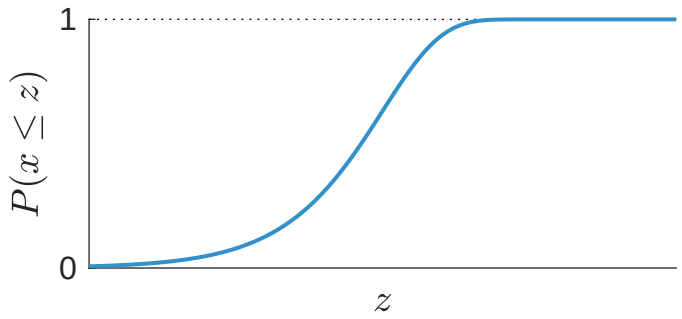


Bild 2-4: *Die kumulierte Wahrscheinlichkeitsdichtefunktion beschreibt die Wahrscheinlichkeit für $x \leq z$ und konvergiert für $z \to \infty$ gegen 1.*

Erwartungswert, Varianz und Kovarianz

Der durchschnittliche Wert einer Funktion $f(x)$ mit Wahrscheinlichkeitsverteilung $p(x)$ nennt sich *Erwartungswert* von $f(x)$. Für diskrete Verteilungen ist dieser gegeben durch

$$\mathbb{E}\left[f(x)\right] = \sum_{x} p(x) \cdot f(x),$$

und für kontinuierliche Verteilungen durch

$$\mathbb{E}\left[f(x)\right] = \int p(x) \cdot f(x)\,\mathrm{d}x.$$

Der Erwartungswert ist ein durch relative Wahrscheinlichkeiten für x gewichteter Wert. Kommt also ein konkretes $x^* \in X$ besonders häufig vor (hohe relative Auftrittswahrscheinlichkeit $p(x^*)$), so wird der zugehörige Wert $f(x^*)$ stärker gewichtet als Werte an anderen Stellen $x \in \{X | x \neq x^*\}$.

Die *Varianz* von $f(x)$ ist das Maß der Streuung von f an der Stelle x um den Mittelwert $\mathbb{E}\left[f(x)\right]$ und ist definiert durch

$$\mathrm{var}\left[f(x)\right] = \mathbb{E}\left[\left(f(x) - \mathbb{E}\left[f(x)\right]\right)^2\right].$$

[6] engl. cumulative distribution function

Darüber hinaus beschreibt die *Kovarianz* zweier zufällig verteilter Variablen x und y des gleichen Wahrscheinlichkeitsraums Ω die Abhängigkeit der Verteilungen der beiden Variablen. Sie ist gegeben durch

$$\begin{aligned}\mathrm{cov}\,[x, y] &= \mathbb{E}_{x,y}\left[\{x - \mathbb{E}[x]\}\{y - \mathbb{E}[y]\}\right] \\ &= \mathbb{E}_{x,y}[xy] - \mathbb{E}[x] \cdot \mathbb{E}[y].\end{aligned}$$

Der Index „x, y“ der Erwartungswertfunktion verdeutlicht, dass an dieser Stelle der Erwartungswert hinsichtlich beider Variablen x, y gemeint ist. In Abbildung 2-5 sind in blau Höhenlinien der Verteilung zweier beispielhafter Variablen x und y gegeneinander aufgetragen. Höhenlinien, die eine höhere Auftrittswahrscheinlichkeit darstellen, liegen dabei weiter innen, als die für weniger wahrscheinliches Auftreten. Wäre die Verteilung der individuellen Variablen unabhängig von der jeweils anderen, so würden sich Höhenlinien in Form gleichförmiger Kreise ergeben. Die dargestellten Ellipsen ergeben sich aufgrund einer vorherrschenden Kovarianz der beiden Variablen. Wird $x = a$ angenommen, so ergibt sich die in Abbildung 2-5 in rot dargestellte Verteilung für y. Für höherdimensionale Variablen $\boldsymbol{x}$ und $\boldsymbol{y}$ kann die Kovarianz entsprechend als Matrix dargestellt werden durch

$$\begin{aligned}\mathrm{cov}\,[\boldsymbol{x}, \boldsymbol{y}] &= \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\left[\{\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}]\}\{\boldsymbol{y}^T - \mathbb{E}[\boldsymbol{y}^T]\}\right] \\ &= \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}[\boldsymbol{x}\boldsymbol{y}^T] - \mathbb{E}[\boldsymbol{x}] \cdot \mathbb{E}[\boldsymbol{y}^T].\end{aligned}$$

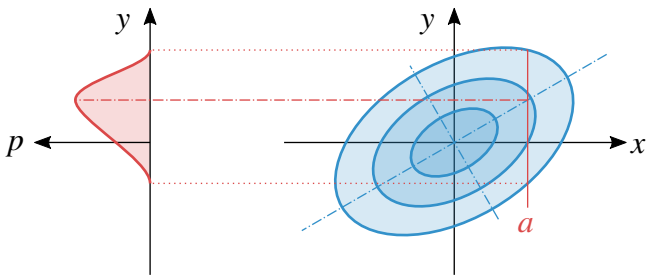


Bild 2-5: *Trägt man die Verteilung zweier Zufallsvariablen x, y gegeneinander auf, ergibt sich ein Bild aus Höhenlinien, das die Kovarianz der beiden Variablen darstellt (rechts in blau). Wird nun $x = a$ gewählt, ergibt sich für y eine entsprechende Wahrscheinlichkeitsverteilung (links in rot).*

Normalverteilungen

Normalverteilungen, oder auch *Gaußverteilungen* genannt, sind eine besonders gängige Verteilungsform kontinuierlicher Variablen. Ihr charakteristisches Erscheinungsbild wird auch „Glockenkurve “ genannt (siehe Abb. 2-6). Sie wird durch nur zwei Parameter eindeutig beschrieben - Mittelwert μ und Varianz σ^2 - und ist definiert als

$$\mathcal{N}(x|\mu, \sigma^2) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}.$$

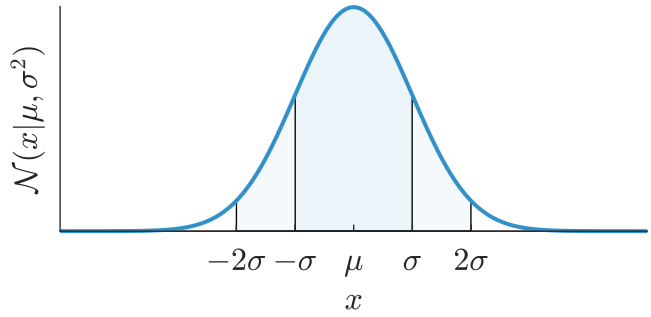


Bild 2-6: *Die Normalverteilung beschreibt die Verteilung einer zufälligen Variable x durch Mittelwert μ und Varianz σ^2 als charakteristische Glocke. Anhand der Auftrittswahrscheinlichkeit für x ergeben sich sogenannte „Sigma-Mengen“.*

Die charakteristischen „Sigma-Mengen“ stellen Teilmengen der Normalverteilung dar und können direkt in Anteile der Gesamtfläche der Verteilung überführt werden. Die „Ein-Sigma-Menge“ ($\mathcal{N}(x|\mu,\sigma^2)$ für alle $x \in [-\sigma,\sigma]$) entspricht ca. 68% der Gesamtfläche und die „Zwei-Sigma-Menge“ ($\mathcal{N}(x|\mu,\sigma^2)$ für alle $x \in [-2\sigma,2\sigma]$) sogar 95% (vgl. [Six19]). Dabei wird σ auch als Standardabweichung bezeichnet. Im weiteren Verlauf wird Varianz in der Regel als „Ein-“ oder „Zwei-Sigma-Menge“ bzw. 68%- oder 95%-Menge dargestellt. D.h., bildet $\mathcal{N}(x|\mu,\sigma^2)$ die Verteilung der Auftrittswahrscheinlichkeit von x ab, liegt ein zufällig gewähltes x in 68% bzw. 95% aller Fälle in dieser Menge.

Ein besonderes Derivat der Normalverteilungen ist die *Standard-Normalverteilung* mit Mittelwert null und Varianz gleich eins:

$$\mathcal{N}(x|0,1) := \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}x^2\right\}.$$

Jede Normalverteilung kann durch Normierung auf die Standard-Normalverteilung zurückgeführt werden. Es wird angenommen ein Sample[7] x_i stammt von einer Normalverteilung mit Mittelwert μ und Varianz σ^2:

$$x_i \sim \mathcal{N}(\mu,\sigma^2).$$

Dann kann x auch dargestellt werden als

$$x_i = \mu + \sigma\varepsilon_i,$$

$$\varepsilon_i \sim \mathcal{N}(0,1).$$

Die multivariate Normalverteilung für $\boldsymbol{x} \in \mathbb{R}^D$ wird entsprechend durch einen D-dimensionalen Mittelwertsvektor $\boldsymbol{\mu}$ und eine $D \times D$-dimensionale symmetrische, positiv definite Kovarianzmatrix $\boldsymbol{\Sigma}$ vollständig beschrieben:

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) := \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{2}\right\}.$$

[7] dt. zufällig gewählte Stichprobe

Um wie im eindimensionalen Fall einzelne Samples einer normalverteilten Zufallsvariable zu erhalten, muss die Kovarianzmatrix mittels der Cholesky-Zerlegung[8] zerlegt werden:

$$\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$
$$\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T.$$

Mit der $D \times D$-dimensionalen Einheitsmatrix $\boldsymbol{E}$ ergibt sich $\boldsymbol{x}_i$ zu

$$\boldsymbol{x}_i = \boldsymbol{\mu} + \boldsymbol{L}\boldsymbol{\varepsilon}_i, \tag{2-1}$$
$$\boldsymbol{\varepsilon}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{E}).$$

2.2 Optimierung

Die Motivation für die Optimierung von Prozessen ist mit festgelegten Mitteln das bestmögliche Resultat erzielt werden, um Kosten und Zeit einzusparen. Das Ganze geschieht unter Einhaltung diverser Randbedingungen. Speziell bei der Prozessparametrierung können dies z.B. Begrenzungen von Stellgrößen sein. Es besteht zwar auch für komplexe hochdimensionale Problemstellungen die Möglichkeit des stupiden Erratens einer optimalen Lösung, auch bekannt als *Random Search*[9], nichtsdestotrotz haben sich im Laufe der Zeit diverse Optimierungsverfahren etabliert und auch heute noch werden immer bessere Verfahren entwickelt, um eine effiziente Lösung zu erzielen. Angenommen es sollen an einer Maschine 10 Parameter optimal eingestellt werden, die alle einen Anteil an der Güte des resultierenden Produkts haben, so ist der Suchraum von 10-dimensionaler Art. Eine Identifikation der optimalen Parametrierung ist nur mit modernsten wahrscheinlichkeitsbasierten Optimierungsverfahren, wie der Bayesschen Optimierung, effizient möglich.

Sei J das Gütemaß für die Güte eines Prozesses, dann ist $J(\boldsymbol{\theta})$ die Gütefunktion, die die Güte einer Parametrierung $\boldsymbol{\theta}$ abbildet. $\boldsymbol{\theta}^*$ entspricht der optimalen Parametrierung für die gilt:

$$\boldsymbol{\theta}^* := \arg\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}). \tag{2-2}$$

Da es sich bei den meisten Optimierungsverfahren um Minimierungsverfahren handelt, ist es nützlich Maximierungs- in äquivalente Minimierungsprobleme, vice versa, umformen zu können. Dies geschieht durch simple Negierung des Optimierungsproblems: [Tim19]

$$\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = -\min_{\boldsymbol{\theta}}(-J(\boldsymbol{\theta})).$$

Im Rahmen von Minimierungsproblemen gilt es, das globale Minimum der Gütefunktion zu finden. Oft sind Gütefunktionen Konstrukte abstrakter Natur und bilden keine interpretierbaren Werte ab. Daher ist im Allgemeinen vor allem die optimale Parametrierung $\boldsymbol{\theta}^*$ von Interesse (Formel (2-2)).

[8] Zerlegung einer symmetrischen, positiv definiten Matrix in das Produkt einer unteren und oberen Dreiecksmatrix

[9] dt. zufällige Suche

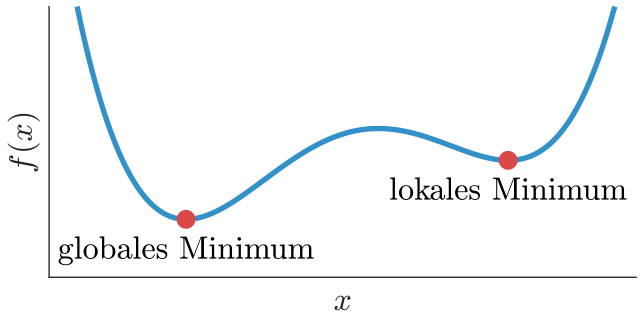


Bild 2-7: *Die dargestellte Kurve hat ein lokales und ein globales Minimum. Im Gegensatz zum lokalen ist der Funktionswert des globalen Minimums kleiner als der Wert von $f(x)$ an jeder Stelle x.*

Minima können untergliedert werden in lokale und globale Minima. Für *lokale Minima* gilt $J(\boldsymbol{\theta}^*) \leq J(\boldsymbol{\theta})$ für alle $\boldsymbol{\theta}$ in hinreichend kleiner Umgebung von $\boldsymbol{\theta}^*$ (Abb. 2-7). Bei einem lokalen Minimum muss es sich deshalb nicht um die optimale Lösung handeln. Als optimale Lösung wird das *globale Minimum* der Gütefunktion bezeichnet, das eine Untermenge der lokalen Minima bildet. Für globale Minima gilt $J(\boldsymbol{\theta}^*) \leq J(\boldsymbol{\theta})$ für alle $\boldsymbol{\theta}$ der zulässigen Menge Ω. Oft handelt es sich bei Optimierungsproblemen um beschränkte Optimierungsprobleme. Hierbei gilt es bestimmte Gleichungs- und Ungleichungsbeschränkungen $g_j(\boldsymbol{\theta}) = 0$ und $h_i(\boldsymbol{\theta}) \leq 0$ einzuhalten. Demnach kann es sich bei einem Minimum auch um einen Randwert der zulässigen Menge $\boldsymbol{\theta} \in \Omega$ handeln. [Tim19]

Für die effiziente Lösung von Optimierungsproblemen nutzen viele Optimierungsverfahren Informationen über das Gefälle der Gütefunktion im aufgespannten Parameterraum, sofern vorhanden[10]. Bei diesem sogenannten *Gradienten* handelt es sich um eine Verallgemeinerung der eindimensionalen Ableitung. Falls $J(\boldsymbol{\theta})$ stetig differenzierbar ist, kann der Gradient von $J(\boldsymbol{\theta})$ an der Stelle $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_n]^T$ geschrieben werden als

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) := \frac{\partial J}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial J}{\partial \theta_1} \\ \vdots \\ \frac{\partial J}{\partial \theta_n} \end{bmatrix}.$$

2.3 Energiebasierte Regelung dynamischer Systeme

Allgemeine nichtlineare dynamische Systeme dienen der Beschreibung dynamischer Systeme, die nichtlineares Zustands- und oder Ausgangsverhalten aufweisen. Das Systemverhalten wird dabei abgebildet durch:

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

$$\boldsymbol{y}(t) = g(\boldsymbol{x}(t), u(t)).$$

Die systemimmanente Dynamik $f(\boldsymbol{x}, u)$ basiert auf innerem Zustand $\boldsymbol{x}(t)$ und Steuergröße $u(t)$. Die von außen messbare Ausgangsgröße $y(t)$ ergibt sich durch die Systemausgabefunktion $g(\boldsymbol{x}, u)$ (vgl. Abb. 2-8).

[10] Die Rede ist hier von gradientenbasierten Suchverfahren

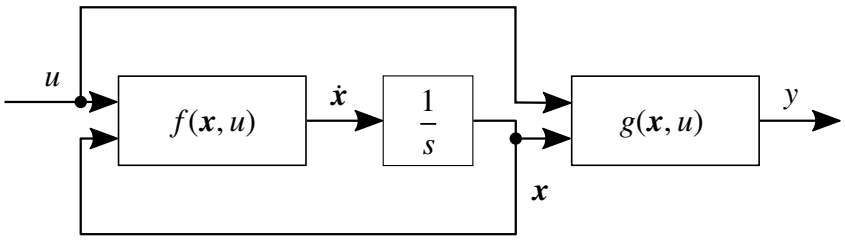


Bild 2-8: *Ein nichtlineares Dynamikmodell kann als Blockschaltbild dargestellt werden und verdeutlicht die systemimmanente Dynamik sowie den Zusammenhang von Ein- und Ausgangsgröße.*

Zur Überführung des Systemzustands $\boldsymbol{x}(t)$ von einem Anfangszustand $\boldsymbol{x}_0 := \boldsymbol{x}(t = t_0)$ in eine Endlage $\boldsymbol{x}_f := \boldsymbol{x}(t = t_f)$ in der Zeit $T := t_f - t_0$ gilt es eine Regelung zu entwerfen (siehe Abb. 2-9). Ein möglicher Ansatz für eine derartige Regelung, bei der lediglich Anfangs- und Endzustand festgeschrieben sind, ist beispielsweise eine Energy Control[11]. Diese spezielle Regelungsvariante macht sich den Energieerhaltungssatz zunutze. Sind Anfangs- und Endzustand bekannt, so können die korrespondierenden Energielevel ermittelt werden. Es gilt nun die nötige Energie zuzuführen, die sich als Differenz von Anfangsenergie $E_0 := E(t = t_f)$ und Energie der Endlage $E_f := E(t = t_f)$ ergibt:

$$\Delta E_{0f} = E_f - E_0.$$

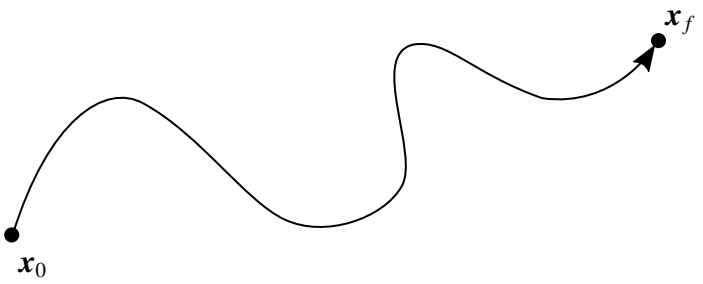


Bild 2-9: *Für die gesuchte Regelung sind lediglich Anfangs- und Endzustand der Zustandstrajektorie relevant. Der resultierende Trajektorienverlauf ist für den Regelungsentwurf nicht von Bedeutung.*

Da die Stellgröße $u(t)$ in der Regel beschränkt ist ($u(t) \leq u_{max}$), kann diese Energie nicht einfach schlagartig zugeführt werden. Im Verlauf dieser Arbeit soll die Energy Control zum Aufschwung eines Pendels genutzt werden (vgl. Abb. 2-10). Daher wird im Weiteren davon ausgegangen, dass es sich um ein schwingendes System handelt. Die zuzuführende Energie ΔE wird daher durch wiederholtes Hin- und Herschwingen ins System

[11] dt. Energieregelung

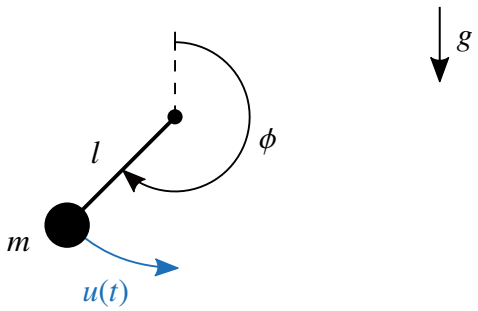


Bild 2-10: *Ein Beispiel für ein schwingendes System ist das starre Pendel. Die Systemparameter sind Länge l und Masse m. Die Regelgröße greift in Form einer Fußpunktanregung an und induziert die Bewegung des Pendels.*

eingebracht und somit auf mehrere Schwünge aufgeteilt. Diese Art der Regelung, bei der abwechselnd in entgegengesetzte Richtungen maximale Stellgrößen aufgebracht werden, wird auch als *Bang-Bang-Strategie* bezeichnet. Um das Pendel in der aufrechten Position zu fangen, reagiert die Regelung sensitiv auf die Energiedifferenz $\Delta E(t) = E(t) - E_f$. Die Stellgröße reduziert sich somit sukzessive bei Annäherung an die obere Ruhelage. Ein möglicher Regelungsansatz dafür ist gegeben durch

$$u(t) = \mathrm{sat}_{u_{max}} \left[k \cdot \left(E(t) - E_f \right) \cdot \mathrm{sign} \left(\dot{\phi}(t) \cos \phi(t) \right) \right],$$

$$E(t) = mgl \left(\frac{1}{2} \left(\frac{\dot{\phi}(t)}{\omega_0} \right)^2 + \cos \phi(t) - 1 \right),$$

$$\omega_0 = \sqrt{mgl/J}.$$

Das Trägheitsmoment J entspricht für das starre Pendel ml^2. Dabei wird zu jedem Zeitpunkt $t \in [t_0, t_f]$ die aktuelle Differenz zwischen Systemenergie $E(t)$ und Energie der Endlage E_f betrachtet. Die Signumfunktion ruft ein oszillierendes Systemverhalten hervor. Allgemein betrachtet, gleicht die Regelung für große Regelfehler einer Bang-Bang-Regelung und für kleine Regelfehler einem Proportionalregler. [ÅF96]

2.4 Regression

Bei *Überwachtem Lernen*[12] handelt es sich um einen Teilbereich des Maschinellen Lernens, der wiederum in zwei Bereiche untergliedert werden kann. Zum einen gibt es Klassifikationsprobleme, die die Identifikation von Mustern und die damit einhergehende Zuordnung zu diskreten Klassen fordern, wohingegen sich Regressionsprobleme mit der Prädiktion von Größen in kontinuierlichen Mengen beschäftigen. Gaussprozesse dienen der Lösung von Regressionsproblemen und daher soll in diesem Kapitel zunächst ein allgemeiner Überblick über Funktionsregression und Curve Fitting[13] gegeben werden. Für den Einstieg soll sich einer Gewichtsraum-Betrachtungweise bedient werden, um später in Kapitel 2.5 zu einer Funktionsraum-Betrachtungsweise überzugehen. [Ras06]

[12] engl. supervised learning
[13] dt. Ausgleichungsrechnung

Angenommen, es werden Daten $\mathcal{D}$ erhoben, die einem reellwertigen Eingangsvektor $\boldsymbol{x} = [x_1, \cdots, x_D]^T$ eine reellwertige Zielvariable y zuordnen, dann bilden n Datenpunkte den Trainings-Datensatz $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)|i = 1, \cdots, n\}$. Die Eingangsvektoren $\boldsymbol{x}_i$ werden zur Design-Matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n]^T$ zusammengefasst und die Daten ergeben sich zu $\mathcal{D} = \{(\boldsymbol{X}, \boldsymbol{y})\}$ mit $\boldsymbol{y} = [y_1, \cdots, y_n]^T$. [Ras06],[Bis06]

Ziel ist es, die Beziehung zwischen Eingaben $\boldsymbol{x}$ und zugehörigen Ausgaben y zu bestimmen. Zur Beschreibung einer derartigen Beziehung wird sich eines Modells bedient. Je nach Anwendungsfall sind hierbei diverse Modelle denkbar. So gibt es z.B. lineare Modelle, quadratische Modelle etc.. Grundsätzlich entspricht ein Modell $f(\boldsymbol{x}, \boldsymbol{w})$ einer Funktion der gewichteten Eingabeparameter x_i mittels Gewichten w_i. Die Gewichte spannen dabei einen c-dimensionalen Raum $\mathcal{W} \subseteq \mathbb{R}^c$ auf. Angenommen die gemessenen Daten $\mathcal{D}$ unterliegen einem unabhängigen, identisch verteilten[14] Rauschen ε_i, ergeben sich die beobachteten Zielwerte zu (vgl. [Ras06])

$$y_i = f(\boldsymbol{x}_i, \boldsymbol{w}) + \varepsilon_i,$$

$$\varepsilon_i \sim \mathcal{N}(0, \sigma_n^2).$$

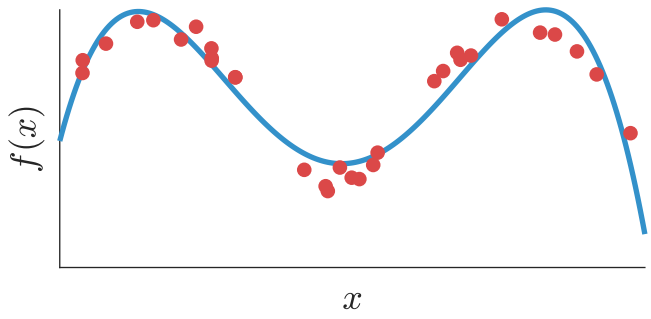


Bild 2-11: Die 30 in rot dargestellten Messpunkte können bei geeigneter Wahl der Gewichte durch eine Polynomfunktion vierten Grades (in blau) approximiert werden.

Zur Veranschaulichung soll das nachfolgende Beispiel dienen. In Abbildung 2-11 ist ein Trainingssatz von $n = 30$ Datenpunkten dargestellt, der mittels eines Polynoms vierter Ordnung approximiert wird[15]. Das entsprechende 5-dimensionale Modell lautet

$$f(x, \boldsymbol{w}) = \sum_{j=0}^{4} w_j x^j.$$

Um eine bestmögliche Anpassung an die Trainingsdaten zu generieren, müssen die Koeffizienten w_i passend gewählt werden. Zur Bewertung der Anpassung dient die Residuenquadratsumme $E(\boldsymbol{w})$, die als Summe der Quadrate der Fehler von Mess- und Datenpunkten zu verstehen ist.

$$E(\boldsymbol{w}) := \sum_{n=1}^{N} [f(x_n, \boldsymbol{w}) - y_n]^2$$

[14] kz. i.i.d. (engl. independent, identically distributed)
[15] engl. polynominal curve fitting

Zur Lösung des Curve-Fitting-Problems muss $\boldsymbol{w}$ so gewählt werden, dass $E(\boldsymbol{w})$ minimal wird. Die optimale Funktion $f(x, \boldsymbol{w}^*)$ ergibt sich durch Lösen eines linearen Gleichungssystems als optimale Parametrierung $\boldsymbol{w}^* = (\boldsymbol{\Upsilon}^T\boldsymbol{\Upsilon})^{-1}\boldsymbol{\Upsilon}^T\boldsymbol{y}$ (vgl. [Bis06]) mit

$$\boldsymbol{\Upsilon} := \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & x_1^4 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 & x_n^4 \end{bmatrix}.$$

Diese Lösung resultierte aus einer diskreten Betrachtung des Ausgangsraums. So wurden nur die Punkte des Ausgangsraums in dem Gleichungssystem berücksichtigt, die in Form von Daten vorliegen. Das Konzept der Regression lässt sich allerdings auch auf den kontinuierlichen Ausgangsraum erweitern und es ergibt sich die gleiche Lösung des Curve-Fitting-Problems.

Das Modell, gegeben durch Gleichung (2-3), beschreibt die Verteilung einer Zielgröße t für jede Stelle $\boldsymbol{x} \in X$ als Normalverteilung mit Mittelwertfunktion $f(\boldsymbol{x}, \boldsymbol{w})$ und Präzision[16] β. Als Pendant der diskreten Residuenquadratsumme wird als geeignetes Maß für die Anpassung des Modells an die Daten im kontinuierlichen Fall die *Likelihood* $p(\mathcal{D}|\boldsymbol{w})$ als Produkt der lokalen Wahrscheinlichkeiten $p(t|\boldsymbol{x}, \boldsymbol{w}, \beta^{-1})$ formuliert (siehe Gleichung (2-4)). Sie drückt die Wahrscheinlichkeit für beobachtete Daten $\mathcal{D}$ angesichts einer Parametrierung $\boldsymbol{w}$ aus. Zur Lösung des Curve-Fitting-Problems gilt es nun, die Likelihood der vorliegenden Daten zu maximieren. Das Argument der auf diese Weise resultierenden *Maximum Likelihood* $\max_{\boldsymbol{w}\in\mathcal{W}} p(\mathcal{D}|\boldsymbol{w})$ ist die optimale Parametrierung $\boldsymbol{w}^*$.

$$p(t|\boldsymbol{x}, \boldsymbol{w}, \beta^{-1}) = \mathcal{N}(t|f(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1}), \tag{2-3}$$

$$p(\mathcal{D}|\boldsymbol{w}) = \prod_{min}^{max} p(t|\boldsymbol{x}, \boldsymbol{w}, \beta^{-1}), \tag{2-4}$$

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}\in\mathcal{W}} p(\mathcal{D}|\boldsymbol{w})$$

Nun, da die Lösung des Curve-Fitting-Problems als Maximierung einer Wahrscheinlichkeitsverteilung formuliert werden kann, kommt die Überlegung über die Einbindung von Vorwissen in Form einer Prior-Verteilung der Gewichte auf. Durch Anwendung des Satzes von Bayes aus Kapitel 2.1.1 kann beispielsweise die Bestimmung möglichst kleiner Gewichte w_i durch Wahl einer geeigneten *Prior-Verteilung* $p(\boldsymbol{w})$ veranlasst werden. Es ergibt sich die sogenannte *Posteriori-Verteilung* und der zugehörige wahrscheinlichste Wert - *Maximum Posteriori*:

$$\text{Posteriori-Verteilung:} \qquad p(\boldsymbol{w}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{w}) \cdot p(\boldsymbol{w}),$$

$$\text{Maximum Posteriori:} \quad \arg\max_{\boldsymbol{w}\in\mathcal{W}} p(\boldsymbol{w}|\mathcal{D}) \propto \arg\max_{\boldsymbol{w}\in\mathcal{W}} p(\mathcal{D}|\boldsymbol{w}) \cdot p(\boldsymbol{w}).$$

Auch für den diskreten Fall gilt, sollen möglichst kleine Gewichte w_i bestimmt werden, kann dies durch Erweiterung des diskreten Modells erfolgen:

[16] Kehrwert der Varianz: $\beta^{-1} := \sigma^2$

$$E(\boldsymbol{w}) := \sum_{n=1}^{N} \left[f(x_n, \boldsymbol{w}) - y_n \right]^2 + \lambda \cdot \|\boldsymbol{w}\|,$$

$$\boldsymbol{w}^* = (\boldsymbol{\Upsilon}^T \boldsymbol{\Upsilon})^{-1} \boldsymbol{\Upsilon}^T \boldsymbol{y} + \lambda \cdot \boldsymbol{E}.$$

Die Größe der Gewichte wird durch die Euklidische Norm $\|\boldsymbol{w}\|$ berücksichtigt, wobei der Einfluss auf die Optimierung durch λ mehr oder weniger priorisiert werden kann.

Wie in Abbildung 2-11 zu erkennen ist, können die Trainingsdaten (in rot) gut durch ein Polynom vierter Ordnung (in blau) approximiert werden. Um den Zusammenhang zwischen Ein- und Ausgangsgrößen noch besser darstellen zu können, kann z.B. ein alternatives Modell gewählt werden.

2.5 Gauß-Prozess-Regression

Bisher wurde im Rahmen von Funktionsregression lediglich eine Gewichtsraumdarstellung betrachtet, bei der sich Funktionswerte $f(\boldsymbol{x}, \boldsymbol{w})$ für den Eingangsraum X mithilfe eines festen Parametersatzes $\boldsymbol{w}$ ergeben. Bei der Funktionsraumdarstellung wird nun die Verteilung von Funktionen explizit betrachtet, ohne dabei eine Verteilung von Gewichten zu nutzen (zumindest in erster Linie (siehe Hyperparameter 2.5.3)). Dazu wird für jeden Punkt $\boldsymbol{x} = [x_1, \cdots, x_D]^T$ des Eingangsraums $X \subseteq \mathbb{R}^D$ der zugehörige Funktionswert f als normalverteilte Zufallsvariable aufgefasst (siehe Abb. 2-12). An jedem Punkt $\boldsymbol{x} \in X$ kann $f(\boldsymbol{x})$ als Zufallswert einer Normalverteilung mit Mittelwert $\mu(\boldsymbol{x})$ und Varianz $\sigma^2(\boldsymbol{x})$ dargestellt werden:

$$f|\boldsymbol{x} = \mathcal{N}(\mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x})).$$

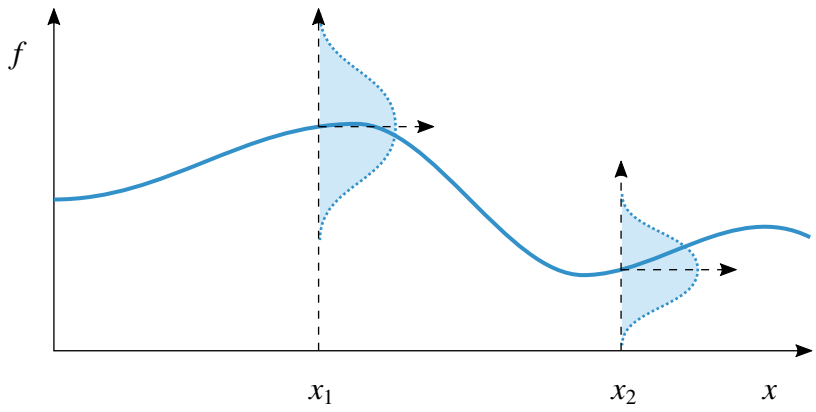


Bild 2-12: Ein Gaußprozess spannt an jeder Stelle x eine Wahrscheinlichkeitsverteilung für f auf. Dieser wird durch Vorgabe einer Mittelwert- und Kovarianzfunktion vollständig definiert .

Das Konglomerat der Normalverteilungen über den Eingangsraum X wird als Gaußprozess (GP) bezeichnet und ist vollständig über seine Mittelwerts- und Kovarianzfunktion $\mathrm{m}(\boldsymbol{x})$ und $\mathrm{k}(\boldsymbol{x}, \boldsymbol{x}')$ definiert:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mathrm{m}(\boldsymbol{x}), \mathrm{k}(\boldsymbol{x}, \boldsymbol{x}')),$$

$$\mathrm{m}(\boldsymbol{x}) := \mathbb{E}[f(\boldsymbol{x})],$$

$$\mathrm{k}(\boldsymbol{x}, \boldsymbol{x}') := \mathbb{E}[(f(\boldsymbol{x}) - \mathrm{m}(\boldsymbol{x}))(f(\boldsymbol{x}') - \mathrm{m}(\boldsymbol{x}'))].$$

Ziel von Gaußprozessen ist es, Prädiktionen über f an Stellen $\boldsymbol{x}_*$ zu generieren, dessen zugehöriger Funktionswert f_* unbekannt ist. Dazu wird der GP zunächst mit Trainingsdaten $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n]^T$ trainiert. Dafür werden die Daten vom GP in Form von Mittelwert- und Kovarianzfunktion abgebildet:

$$\boldsymbol{f}|\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{m}(\boldsymbol{X}), \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})),$$

$$\boldsymbol{m}(\boldsymbol{X}) := \begin{bmatrix} \mathrm{m}(\boldsymbol{x}_1) \\ \vdots \\ \mathrm{m}(\boldsymbol{x}_n) \end{bmatrix},$$

$$\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) := \begin{bmatrix} \mathrm{var}(\boldsymbol{x}_1) & \mathrm{cov}(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & \mathrm{cov}(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \mathrm{cov}(\boldsymbol{x}_2, \boldsymbol{x}_1) & \mathrm{var}(\boldsymbol{x}_2) & \cdots & \mathrm{cov}(\boldsymbol{x}_2, \boldsymbol{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{cov}(\boldsymbol{x}_n, \boldsymbol{x}_1) & \mathrm{cov}(\boldsymbol{x}_n, \boldsymbol{x}_2) & \cdots & \mathrm{var}(\boldsymbol{x}_n) \end{bmatrix}$$

$$= \begin{bmatrix} \mathrm{k}(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & \mathrm{k}(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ \mathrm{k}(\boldsymbol{x}_n, \boldsymbol{x}_1) & \cdots & \mathrm{k}(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix}.$$

Um nun Prädiktionen für Punkte $\boldsymbol{X}_* = [\boldsymbol{x}_{*1}, \cdots, \boldsymbol{x}_{*n_*}]^T$ zu generieren, wird die Verteilung von $\boldsymbol{f}_* = [f_{*1}, \cdots, f_{*n_*}]^T$ auf Basis der Daten gegeben durch

$$\boldsymbol{f}_*|\boldsymbol{X}_*, \boldsymbol{X}, \boldsymbol{f} \sim \mathcal{N}(\boldsymbol{\mu}^+, \boldsymbol{\Sigma}^+),$$

$$\boldsymbol{\mu}^+ := \boldsymbol{m}(\boldsymbol{X}_*) + \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \cdot \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1}(\boldsymbol{f}(\boldsymbol{X}) - \boldsymbol{m}(\boldsymbol{X})),$$

$$\boldsymbol{\Sigma}^+ := \boldsymbol{K}(\boldsymbol{X}_*\boldsymbol{X}_*) - \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \cdot \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*),$$

$$\boldsymbol{f}(\boldsymbol{X}) := \begin{bmatrix} f(\boldsymbol{x}_1) \\ \vdots \\ f(\boldsymbol{x}_n) \end{bmatrix},$$

$$\boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) = \begin{bmatrix} \mathrm{cov}(\boldsymbol{x}_{*1}, \boldsymbol{x}_1) & \mathrm{cov}(\boldsymbol{x}_{*1}, \boldsymbol{x}_2) & \cdots & \mathrm{cov}(\boldsymbol{x}_{*1}, \boldsymbol{x}_n) \\ \vdots & \vdots & & \vdots \\ \mathrm{cov}(\boldsymbol{x}_{*n_*}, \boldsymbol{x}_1) & \mathrm{cov}(\boldsymbol{x}_{*n_*}, \boldsymbol{x}_2) & \cdots & \mathrm{cov}(\boldsymbol{x}_{*n_*}, \boldsymbol{x}_n) \end{bmatrix} = \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*)^T.$$

Berücksichtigung von Rauschen

Da reale Systeme in der Regel verrauschte Messdaten liefern bzw. die involvierten Messverfahren eine gewisse Ungenauigkeit nicht ausschließen lassen, soll auch für die Mess-

werte ein Messrauschen berücksichtigt werden. Dabei wird angenommen, dass das Rauschen durch i.i.d. gaußsches Rauschen ε abgebildet werden kann. Das Rauschen kommt von einer Normalverteilung mit Mittelwert null und Varianz σ_n^2:

$$y = f(\boldsymbol{x}) + \varepsilon,$$
$$\varepsilon \sim \mathcal{N}(0, \sigma_n^2).$$

Mit additivem Rauschen ergeben sich folgende Erweiterungen von Kovarianzmatrix und GP: [Ras06]

$$\operatorname{cov}(\boldsymbol{y}) = K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{E},$$
$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}^-(\boldsymbol{X}), \\ \boldsymbol{\mu}^-(\boldsymbol{X}_*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{E} & \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*) \\ \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) & \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) \end{bmatrix}\right),$$
$$\boldsymbol{f}_* | \boldsymbol{X}_*, \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}^+(\boldsymbol{X}_*), \boldsymbol{\Sigma}^+(\boldsymbol{X}_*)),$$
$$\boldsymbol{\mu}^+(\boldsymbol{X}_*) = \boldsymbol{m}(\boldsymbol{X}_*) + \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \cdot (\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{E})^{-1} (\boldsymbol{f}(\boldsymbol{X}) - \boldsymbol{m}(\boldsymbol{X})),$$
$$\boldsymbol{\Sigma}^+(\boldsymbol{X}_*) = \boldsymbol{K}(\boldsymbol{X}_* \boldsymbol{X}_*) - \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \cdot (\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{E})^{-1} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*).$$

Neue Notation

Um im Folgenden eine übersichtlichere Darstellung zu ermöglichen, wird eine neue Notation eingeführt. Deshalb soll im weiteren Verlauf eine reduzierte Schreibweise für die Kovarianzmatrix genutzt werden:

$$\boldsymbol{K} := \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}),$$
$$\boldsymbol{K}_* := \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*),$$
$$\boldsymbol{K}_{**} := \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*).$$

Im Falle, dass nur ein Testpunkt $\boldsymbol{x}_*$ ausgewertet werden soll, ergibt sich für $\boldsymbol{K}_*$ ein Spaltenvektor und für $\boldsymbol{K}_{**}$ ein Skalar, im Weiteren notiert als $\boldsymbol{k}_*$ und k_{**}:

$$\boldsymbol{k}_* := \boldsymbol{k}(\boldsymbol{X}, \boldsymbol{x}_*),$$
$$k_{**} := k(\boldsymbol{x}_*, \boldsymbol{x}_*).$$

Darüber hinaus soll auch die Notation der Mittelwertmatrizen eindeutiger gestaltet werden. Um den Prior-Mittelwert $\mathbb{E}[\boldsymbol{f}_* | \boldsymbol{X}_*]$ im Folgenden klar vom Posterior-Mittelwert $\mathbb{E}[\boldsymbol{f}_* | \boldsymbol{X}_*, \boldsymbol{X}, \boldsymbol{y}]$ abgrenzen zu können wird der Prior mit einem hochgestellten ‚$-$‘ und der Posterior mit einem ‚$+$‘ markiert:

$$\mu^-(\boldsymbol{X}_*) := \mathbb{E}[\boldsymbol{f}_* | \boldsymbol{X}_*],$$
$$\mu^+(\boldsymbol{X}_*) := \mathbb{E}[\boldsymbol{f}_* | \boldsymbol{X}_*, \boldsymbol{X}, \boldsymbol{y}].$$

Zentrale Formeln

$$\boldsymbol{f}_*|\boldsymbol{X}_*, \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}^+(\boldsymbol{X}_*), \boldsymbol{\Sigma}^+(\boldsymbol{X}_*))$$

$$\boldsymbol{\mu}^+(\boldsymbol{X}_*) = \boldsymbol{\mu}^-(\boldsymbol{X}_*) + \boldsymbol{K}_*^T \cdot (\boldsymbol{K} + \sigma_n^2 \boldsymbol{E})^{-1}(\boldsymbol{f}(\boldsymbol{X}) - \boldsymbol{\mu}^-(\boldsymbol{X}))$$

$$\boldsymbol{\Sigma}^+(\boldsymbol{X}_*) = \boldsymbol{K}_{**} - \boldsymbol{K}_*^T \cdot (\boldsymbol{K} + \sigma_n^2 \boldsymbol{E})^{-1} \boldsymbol{K}_*$$

2.5.1 Mittelwert und Kovarianz

Die Mittelwertfunktion m($\boldsymbol{x}$) eines GP beschreibt den Erwartungswert $\mathbb{E}[f(\boldsymbol{x})]$ für $\boldsymbol{x} \in X$. In vorangegangenen Teilen des Kapitels 2.5 wird die Beziehung zwischen Prior- und Posteriori-Mittelwert $\mu^-(\boldsymbol{x})$ und $\mu^+(\boldsymbol{x})$ genauer beleuchtet. Anschaulich betrachtet, werden Funktionen $\boldsymbol{f}_i$ von der Posteriori Verteilung realisiert und geplottet, ergibt sich ein Plot wie in Abbildung 2-13. Wird angenommen, dass die Daten kein Messrauschen beinhalten, so verlaufen alle Funktionen (in hellblau) und das arithmetische Mittel zu jeder Stelle $\boldsymbol{x}$ (in dunkelblau) durch die einzelnen Datenpunkte (in rot). Der Verlauf der Funktionen in Abbildung 2-13 ist stetig. Dies ist ein Phänomen der gewählten Kovarianzfunktion. Es gibt jedoch auch Kovarianzfunktionen, die keinen stetigen Verlauf der einzelnen Samples ergeben. Werden unendlich viele Funktionssamples vom GP gezogen, ergibt sich der Posteriori-Mittelwert als arithmetisches Mittel an jeder Stelle $\boldsymbol{x}$. Die Posteriori-Mittelwertfunktion kann darüber hinaus als Funktion der Prior-Mittelwertfunktion betrachtet werden. Die Prior-Mittelwertfunktion an sich wird für einen GP vorgegeben. Im Regelfall wird dieser sogenannte Prior gleich null gesetzt. Da diese Arbeit sich jedoch besonders auf das durch den Prior eingebrachte Vorwissen stützen soll, ist dieser im weiteren Verlauf von hoher Bedeutung und kann, im Gegensatz zur Verfahrensweise in vielen wissenschaftlichen Arbeiten, nicht vernachlässigt werden. [Bis06]

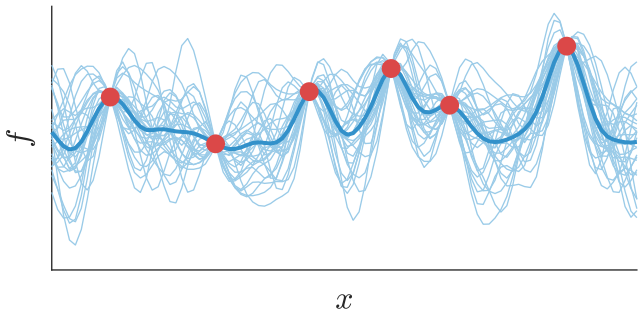


Bild 2-13: *Dargestellt sind 30 zufällig gezogene Samples der Posterior-Verteilung (in hellblau) mit arithmetischem Mittel (in dunkelblau) auf Basis vorgegebener Messpunkte (in rot). Durch die Verteilung der Funktionsverläufe kann eine gute Approximation der resultierenden Posterior-Verteilung gebildet werden.*

Die Kovarianzfunktion k($\boldsymbol{x}, \boldsymbol{x}'$) beschreibt, wie stark sich der Funktionswert $f(\boldsymbol{x})$ an zwei Stellen $\boldsymbol{x}, \boldsymbol{x}' \in X \subseteq \mathbb{R}^D$ gegenseitig bedingt (siehe Kapitel 2.1.2 für Grundlagen der Kovarianz). Grundsätzlich gilt bei Gaußprozessen, dass sich ähnliche Punkte stärker beeinflussen als weniger ähnliche - sie haben eine höhere Kovarianz. Ähnlichkeit bedeutet in

diesem Zusammenhang Distanz. Liegen zwei Punkte nah beieinander (die zugehörigen Stützvektoren ähneln sich), gleicht sich der jeweilige Funktionswert eher, als bei zwei weit voneinander entfernten Punkten. Die Kovarianzfunktion stellt in diesem Fall eine Art Abstandsfunktion dar. Ein renommiertes Beispiel für eine derartige Kovarianzfunktion, auch *Kernel* genannt, ist der *Squared Exponential Kernel*[17] (SQExp-Kernel) (Formel (2-5) (vereinfacht ohne Rauschen, für Weitergehendes siehe Kapitel 2.5.2)), der aufgrund seiner Popularität zunächst als Standardkernel angenommen werden soll. Eine detaillierte Gegenüberstellung der in dieser Arbeit verwendeten Kernel-Funktion ist in Kapitel 2.5.2 zu finden. [Ras06]

$$\text{Squared-Exponential-Kernel:}\quad \mathrm{k}(\boldsymbol{x},\boldsymbol{x}') = \sigma_f^2 \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}')^T \boldsymbol{M}^{-1}(\boldsymbol{x}-\boldsymbol{x}')\right\},$$
$$\boldsymbol{M} := \begin{bmatrix} l_1^2 & & \\ & \ddots & \\ & & l_D^2 \end{bmatrix} \tag{2-5}$$

GPs gelten als parameterfreie Prozesse, und doch besitzen sie Parameter, die angepasst werden müssen. Diese sogenannten Hyperparameter (hier σ_f und $[l_1, \cdots, l_D]$) haben Einfluss auf die Gestalt des GP und sollen vorerst als gegeben betrachtet werden. In Kapitel 2.5.3 soll ausführlich auf Interpretation und Bestimmung der Hyperparameter eingegangen werden.

Eine zentrale Komponente der Posteriori-Verteilung ist die Kernel-Matrix $\boldsymbol{K}$. Die Einträge der sogenannten K-Matrix werden durch die gewählte Kernel-Funktion bestimmt. Eine charakteristische Eigenschaft der K-Matrix ist ihr symmetrischer Aufbau. Darüber hinaus muss die Matrix positiv definit sein, um eine Invertierung zu gewährleisten. Da gerade beim Training des GPs mit ähnlichen Datenpunkten die Gefahr einer Singularität der K-Matrix und damit einer numerischen Instabilität besteht, wird in der Praxis für die Berechnung der Inversen $\boldsymbol{\Lambda}$ zusätzlich ein sogenannter *Jitter-Term*[18] eingeführt, der künstlichem Rauschen entspricht. Bei der Berechnung der K-Matrix kann dieser z.B. in folgender Form vorliegen:

$$\boldsymbol{K} = \begin{bmatrix} \mathrm{k}(\boldsymbol{x}_1,\boldsymbol{x}_1) & \cdots & \mathrm{k}(\boldsymbol{x}_1,\boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ \mathrm{k}(\boldsymbol{x}_n,\boldsymbol{x}_1) & \cdots & \mathrm{k}(\boldsymbol{x}_n,\boldsymbol{x}_n) \end{bmatrix} + \underbrace{\sigma_f^2 10^{-10}\boldsymbol{E}}_{\text{Jitter-Term}},$$
$$\boldsymbol{\Lambda} := (\boldsymbol{K} + \sigma_n^2 \boldsymbol{E})^{-1},$$
$$\boldsymbol{\Sigma}^+(\boldsymbol{X}_*) = \boldsymbol{K}_{**} - \boldsymbol{K}_*^T \boldsymbol{\Lambda} \boldsymbol{K}_*.$$

[17] dt. quadratisch exponentieller Kernel
[18] jitter engl. für Schwankung

Um Funktionen $\boldsymbol{f}_i$ von der Posterior-Verteilung samplen zu können, bedarf es der Zerlegung der Kernel-Matrix (vgl. Kapitel 2.1.2 für Sampling von Verteilungen). Dazu wird die Matrix $\boldsymbol{K}$ mittels Cholesky-Zerlegung umgeformt zu

$$\boldsymbol{K} = \boldsymbol{L}\boldsymbol{L}^T.$$

Entspricht $\boldsymbol{\mu}$ dem Mittelwert für $\boldsymbol{x} \in X$, können entsprechende Funktionen $\boldsymbol{f}_i$ von folgender Verteilung gesampled werden:

$$\boldsymbol{f}_i = \boldsymbol{\mu} + \boldsymbol{L} \cdot \boldsymbol{\varepsilon}_i,$$
$$\boldsymbol{\varepsilon}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{E}).$$

2.5.2 Kernel-Funktionen

In diesem Kapitel soll ein Überblick über die Kernel-Funktionen gegeben werden, die im Rahmen dieser Arbeit Anwendung finden. Dabei wird neben Definition auch auf die Charakteristika eingegangen, die sie besonders zur Modellierung von Gütefunktionen eignen.

Squared-Exponential

Der Squared-Exponential-Kernel, kurz SQExp-Kernel, wurde bereits in Kapitel 2.5.1 eingeführt. Nun soll zudem ein gewisses Messrauschen mit Varianz σ_n^2 berücksichtigt werden, das durch das Kronecker-Delta δ modelliert wird:

$$\mathrm{k}(\boldsymbol{x}_q, \boldsymbol{x}_p) = \sigma_f^2 \exp\left\{-\frac{1}{2}(\boldsymbol{x}_q - \boldsymbol{x}_p)^T \boldsymbol{M}^{-1}(\boldsymbol{x}_q - \boldsymbol{x}_p)\right\} + \delta_{qp} \cdot \sigma_n^2,$$

$$\boldsymbol{M} := \begin{bmatrix} l_1^2 & & \\ & \ddots & \\ & & l_D^2 \end{bmatrix},$$

$$\delta_{qp} := \begin{cases} 1, & q = p \\ 0, & \text{sonst.} \end{cases}$$

Werden einzelne Funktionen einer Prior-Verteilung mit SQExp-Kernel mithilfe der Formel (2-1) gesampled, zeigen diese einen sehr glatten Verlauf (siehe Abb. 2-14), der an Polynomfunktionen oder überlagerte gaußsche Basisfunktionen erinnert. Es kann gezeigt werden, dass der Verlauf stetig und der SQExp-Kernel unendlich oft differenzierbar ist.

Die Schwäche des SQExp-Kernels ist seine mangelhafte Abbildung stückweise unstetiger Verläufe. In diesem Kontext bietet der *Matern-Kernel*, welcher nachfolgend betrachtet werden soll, eine deutlich bessere Abbildungsfähigkeit komplexer Verläufe.

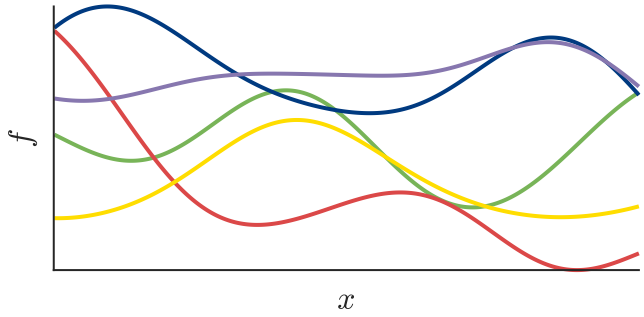


Bild 2-14: *Der SQExp-Kernel produziert Funktionssamples mit einem sehr glatten Verlauf.*

Matern

Matern bildet eine Klasse von Kernel-Funktionen, von denen besonders Matern-32- und Matern-52-Kernel für die Bayessche Optimierung relevant sind (vgl. [Ras06]). Nach [SLA12] eignet sich davon explizit der *Matern-52-Kernel* (M52-Kernel) für die Abbildung komplexer Verläufe. Er ist gegeben durch folgende Gleichungen:

$$\mathrm{k}(\boldsymbol{x}_q, \boldsymbol{x}_p) = \sigma_f^2 \left(1 + \sqrt{5}d + \frac{5}{3}d^2\right) \cdot \exp\left(-\sqrt{5}d\right) + \delta_{qp} \cdot \sigma_n^2,$$

$$d := \sqrt{(\boldsymbol{x}_q - \boldsymbol{x}_p)^T \boldsymbol{M}^{-1} (\boldsymbol{x}_q - \boldsymbol{x}_p)},$$

$$\boldsymbol{M} := \begin{bmatrix} l_1^2 & & \\ & \ddots & \\ & & l_D^2 \end{bmatrix},$$

$$\delta_{qp} := \begin{cases} 1, & q = p \\ 0, & \text{sonst.} \end{cases}$$

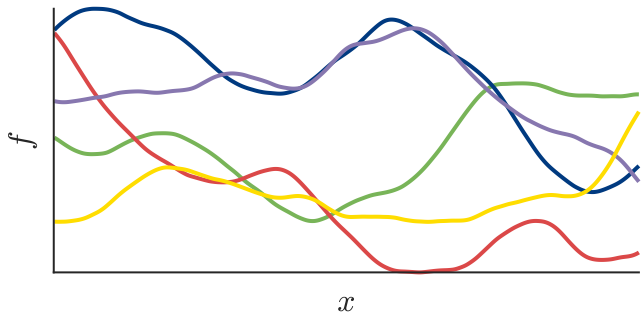


Bild 2-15: *Der deutlich weniger glatte Verlauf der durch M52-Kernel erzeugten Funktionen ist charakteristisch für die Klasse der Matern-Kernel.*

Im Gegensatz zum SQExp-Kernel ist der Verlauf des M52-Kernels deutlich weniger glatt (siehe Abb. 2-15). Diese Eigenschaft ermöglicht es mittels M52-Kernel komplexe Verläufe abzubilden. Dabei ist der Verlauf einzelner Samples sowie der GP-Verlauf stetig.

Um die Überlegenheit von M52 zu SQExp zu verifizieren (bereits nachgewiesen in [SLA12]), wurden Testreihen für beide Kernelfunktionen aufgenommen (siehe Abb. 2-16. Es zeigt sich, dass M52 bei dem betrachteten Beispiel nur geringfügige Vorteile ggü. SQExp verzeichnen lässt. Die Performance von M52 bleibt dennoch unübertroffen.

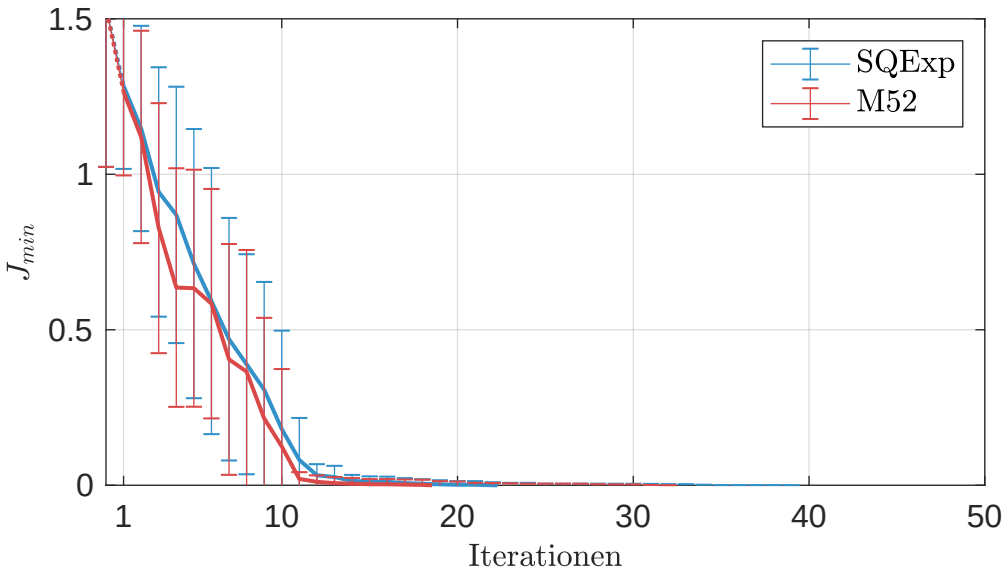


Bild 2-16: *Ein Vergleich der beiden Kernelfunktionen zeigt die leicht überlegene Performance von M52. Dafür wurden Mittelwert und Standardabweichung über 20 Durchläufe gebildet.*

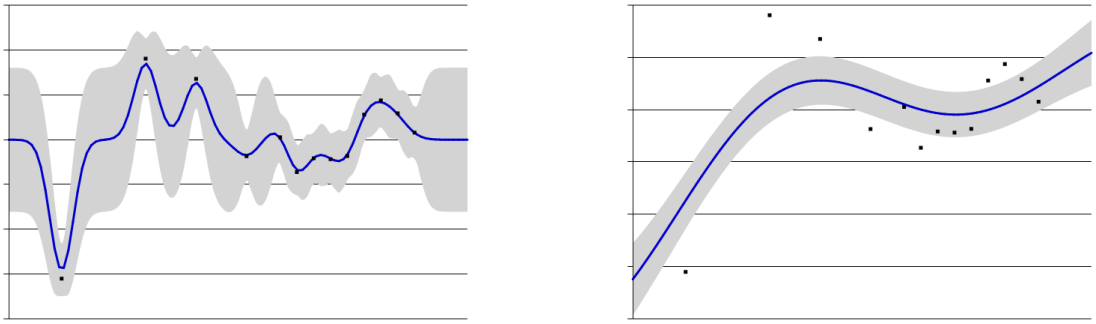
2.5.3 Interpretation der Hyperparameter

Bis zum jetzigen Zeitpunkt wurden die Hyperparameter als gegeben angesehen. Nun soll ein Verständnis für die Eigenschaften dieser Parameter geschaffen werden. Da ein GP über direkte Modellparameter marginalisiert, wird dieser faktisch nur von einer handvoll Hyperparameter gesteuert (vgl. [DLvht]). Dadurch kommt den Hyperparametern besondere Bedeutung zu. Die in dieser Arbeit betrachteten Kernel haben drei unterschiedliche Arten von Hyperparametern - Signal Variance σ_f^2, ein Lengthscale-Parameter l_i je Dimension und Noise Variance σ_n^2.

Lengthscale-Parameter

Lengthscale-Parameter[19] dienen der Skalierung des Eingangsraums. Um jede einzelne Dimension des Eingangsraums unabhängig voneinander skalieren zu können, gibt es für jede Dimension einen Parameter $l_i \geq 0$. Mithilfe des Längenskala-Parameters wird die Korrelation der Funktionswerte zweier Punkte $\boldsymbol{x}$, $\boldsymbol{x}'$ variiert (vgl. Gleichungen (2-6)). Wird eine lange Längenskala gewählt, also l_i groß, so korrelieren die zugehörigen Funktionswerte $f(\boldsymbol{x})$, $f(\boldsymbol{x}')$ stärker miteinander (siehe Abb. 2-17b). Es ergibt sich ein relativ monotoner GP-Verlauf. Kurze Längenskalen dagegen (l_i klein) ermöglichen vergleichsweise

[19] dt. Längenskala-Parameter



(a) *kurze Lengthscale* (b) *lange Lengthscale*

Bild 2-17: *Die Lengthscale beeinflusst die Skalierung des Eingangsraums. Hier eine Gegenüberstellung des Effekts bei Vorgabe kleiner (a) bzw. großer Werte (b). [Gab04]*

starke Fluktuation der Funktionswerte des GPs (siehe Abb. 2-17a). Funktionswerte korrelieren dann nur, wenn die entsprechenden Eingangsraumpunkte nah beieinander liegen. Dies ermöglicht deutliche stärkere Schwankungen des GP-Verlaufs und die Anpassung an mehr Datensätze. [DLvht],[Gab04]

$$\mathrm{k}(\boldsymbol{x}, \boldsymbol{x}') \equiv \mathrm{k}(d(\boldsymbol{x}, \boldsymbol{x}')),$$

$$\begin{aligned} d(\boldsymbol{x}, \boldsymbol{x}') &:= \sqrt{(\boldsymbol{x} - \boldsymbol{x}')^T \cdot \begin{bmatrix} l_1^2 & & \\ & \ddots & \\ & & l_D^2 \end{bmatrix}^{-1} \cdot (\boldsymbol{x} - \boldsymbol{x}')} \\ &= \sqrt{\frac{(x_1 - x_1')^2}{l_1^2} + \cdots + \frac{(x_D - x_D')^2}{l_D^2}}. \end{aligned} \tag{2-6}$$

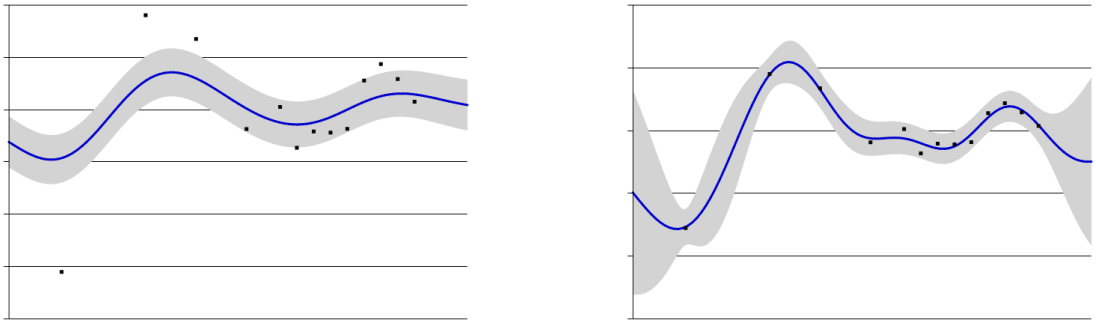
Darüber hinaus kann durch die unterschiedliche Skalierung der Eingangsdimensionen die Relevanz der jeweiligen Eingangsdimension bestimmt werden. Im Rahmen dieser sogenannten *Automatic Relevance Determination* [20] (ARD) deuten besonders kleine Längenskala-Parameter auf eine geringe Bedeutung der entsprechenden Eingangsdimension auf den Ausgang hin (vgl. [Ras06]). So können im Bezug auf die Einstellung von Prozessparametern Aussagen über die Sensitivität bestimmter Parameter gemacht werden. [Bis06]

Signal Variance

Kernel haben in der Regel einen *Signal-Variance-Parameter*[21], der direkte Auswirkung auf die Amplitude des GP hat. Er kann als Amplitudenskalierungsfaktor oder Amplitudenvarianzfaktor interpretiert werden. Sei eine normalisierte Kovarianzfunktion $\mathrm{k}(\tilde{\boldsymbol{x}}, \boldsymbol{x}') \in [0, 1]$ gegeben durch

$$\mathrm{k}(\boldsymbol{x}, \boldsymbol{x}') \equiv \sigma_f^2 \tilde{\mathrm{k}}(\boldsymbol{x}, \boldsymbol{x}') \equiv \sigma_f^2 \tilde{\mathrm{k}}(\boldsymbol{x} - \boldsymbol{x}').$$

[20] dt. automatischen Relevanzbestimmung
[21] dt. Signalvarianz-Parameter



(a) *kleine Signal Variance* (b) *große Signal Variance*

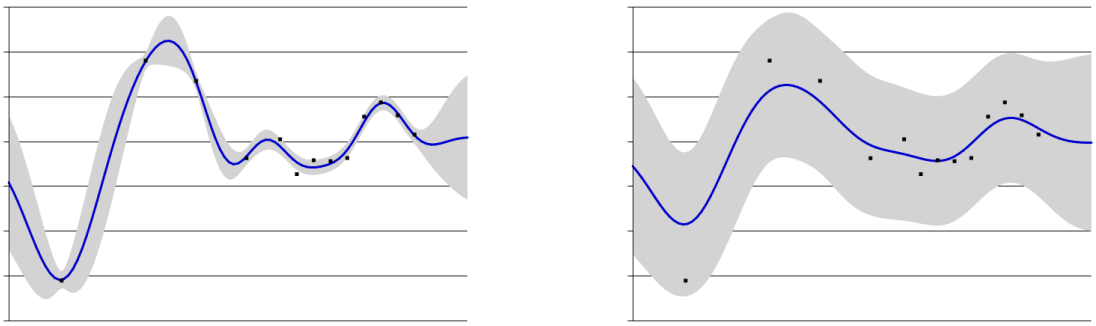
Bild 2-18: *Die Signal Variance dient zur Skalierung der Amplitude des GPs. Für niedrige Werte (a) ergibt sich ggü. großer Werte (b) ein vergleichsweise flacher Verlauf. [Gab04]*

Die Varianz des GPs an einer Stelle $\boldsymbol{x}$ ergibt dann die stationäre Signalvarianz $\sigma_f^2 > 0$:

$$\text{var}[f(\boldsymbol{x})] = \text{k}(\boldsymbol{x}, \boldsymbol{x}) \stackrel{!}{=} \sigma_f^2 \tilde{\text{k}}(\boldsymbol{x} - \boldsymbol{x}) = \sigma_f^2 \tilde{\text{k}}(0) = \sigma_f^2.$$

Auf den gesamten Eingangsraum X bezogen, implizieren große Werte für σ_f^2 also eine hohe stationäre Varianz der Funktionswerte und erlauben damit starke Abweichungen vom Mittelwert des GPs (vgl. Abbildung 2-18). [Gab04],[DLvht]

Noise Variance



(a) *kleine Noise Variance* (b) *große Noise Variance*

Bild 2-19: *Wird die Noise Variance besonders groß gewählt (b), verläuft der GP nur sehr grob entlang der Messpunkte. Bei Vorgabe besonders niedrigen Rauschens (a), wird der GP durch alle Punkte stark beeinflusst. [Gab04]*

Auch wenn es sich bei *Noise Variance*[22] $\sigma_n^2 \geq 0$ genau genommen nicht um einen Parameter der Kovarianzfunktion handelt, wird er doch als Teil eines GPs genutzt, um die Verrauschtheit der Messdaten zu modellieren. Angenommen die Messdaten unterliegen keinen Schwankungen ($\sigma_n^2 = 0$), so wird die Posteriori-Mittelwertfunktion durch alle

[22] dt. Rauschvarianz

Datenpunkte verlaufen. Dabei können sich beliebig komplexe Verläufe ergeben. Soll dagegen ein glatter Kurvenverlauf beschrieben werden, muss ein hohes σ_n^2 angenommen werden. Neben Ausreißern werden mit steigender Noise Variance immer mehr Kompromisse angesichts der Abbildung der Datenpunkte gemacht (vgl. Abbildung 2-19). Für $\sigma_n^2 \to \infty$ ergibt sich für den GP ein konstanter Mittelwert. [Gab04],[DLvht]

2.5.4 Bestimmung der Hyperparameter

Die Bestimmung der Hyperparameter (HP) und damit die Anpassung der Kovarianzfunktion an die gegebenen Daten stellt ein unterlagertes Optimierungsproblem dar. Der damit verbundene Rechenaufwand hat dabei einen großen Anteil am gesamten Rechenaufwand, der für die Berechnung des GPs notwendig ist. Die involvierten Vorgänge sollen an dieser Stelle nur grob beschrieben werden. Weiterführende Informationen sind in den angegebenen Quellen zu finden.

Maximum Likelihood

In Kapitel 2.4 wurde Maximum Likelihood zur optimalen Parametrierung eines Regressionsproblems eingeführt. Auch für einen GP kann die Wahrscheinlichkeit beobachteter Daten $\boldsymbol{X}, \boldsymbol{y}$ angesichts einer festen Parametrierung $\boldsymbol{\theta}$ in Form der Log-Likelihood-Funktion[23] angegeben werden (vgl. [Bis06] und [Ras06]):

$$\ln p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = -\frac{1}{2} \ln |\boldsymbol{K}| - \frac{1}{2} \boldsymbol{y}^T \boldsymbol{K}^{-1} \boldsymbol{y} - \frac{N}{2} \ln(2\pi).$$

Da die Maximierung der Likelihood-Funktion in der Regel mithilfe eines gradientenbasierten Multistart[24]-Optimierungsalgorithmus erfolgt, wird auch der Gradient der Likelihood-Funktion nach den Hyperparametern gebildet (vgl. [Ras06] und [DLvht]):

$$\begin{aligned}
\frac{\partial}{\partial \theta_i} \ln p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) &= -\frac{1}{2} \mathrm{Tr}\left(\boldsymbol{K}^{-1} \frac{\partial \boldsymbol{K}}{\partial \theta_i} \right) + \frac{1}{2} \left(\boldsymbol{y} - \boldsymbol{\mu}^- \right)^T \boldsymbol{K}^{-1} \frac{\partial \boldsymbol{K}}{\partial \theta_i} \boldsymbol{K}^{-1} \left(\boldsymbol{y} - \boldsymbol{\mu}^- \right) \\
&= -\frac{1}{2} \mathrm{Tr}\left(\boldsymbol{K}^{-1} \frac{\partial \boldsymbol{K}}{\partial \theta_i} \right) + \frac{1}{2} \mathrm{Tr}\left(\boldsymbol{\alpha}^T \frac{\partial \boldsymbol{K}}{\partial \theta_i} \boldsymbol{\alpha} \right), \quad \text{mit } \boldsymbol{\alpha} := \boldsymbol{K}^{-1} \left(\boldsymbol{y} - \boldsymbol{\mu}^- \right) \\
&= \frac{1}{2} \mathrm{Tr}\left(\left(\boldsymbol{\alpha} \boldsymbol{\alpha}^T - \boldsymbol{K}^{-1} \right) \frac{\partial \boldsymbol{K}}{\partial \theta_i} \right).
\end{aligned}$$

Unter den in dieser Arbeit erwähnten Methoden zur Bestimmung der Hyperparameter ist die Maximum-Likelihood-Methode die Methode, die mit dem niedrigsten Rechenaufwand gute Ergebnisse erzielt. Da es sich im Rahmen der Maximierung der Likelihood-Funktion oft aufgrund der gegebenen Daten um ein Optimierungsproblem mit lokalen Maxima handelt, können die Hyperparametersätze über mehrere Iterationen betrachtet

[23] Der Logarithmus stellt die Einhaltung der Beschränkungen des Suchraums sicher. Für weitere Informationen siehe [Bis06].

[24] Ermöglicht die Evaluierung des globalen Maximums.

zwischen diesen Maxima wechseln. Oft handelt es sich bei diesen konkurrierenden Maxima um ein Maximum mit hohem Rauschen und konservativem Verlauf und einem Maximum mit niedrigem Rauschen und starker Fluktuation des Funktionswerts (siehe dazu Kapitel 5.4 in [Ras06]). Zur besseren Approximation des wahren Systemverhaltens kann daher eine Superposition mehrerer GPs mit unterschiedlichen Hyperparametersätzen hilfreich sein. Dadurch können, gerade in frühen Iterationen, oft bessere Ergebnisse erzielt werden. Dieser Zusammenhang kann sich mittels HP-Sampling per Markov-Chain-Monte-Carlo-Sampling zunutze gemacht werden.

Markov-Chain-Monte-Carlo-Sampling

In diesem Kapitel soll ein grober Überblick über die Bestimmung der Hyperparameter eines GPs per Markov-Chain-Monte-Carlo-Sampling gegeben werden. Im vorangegangenen Kapitel wurden die Hyperparameter als Maximum der Likelihood-Verteilung $p(\boldsymbol{X}, \boldsymbol{y}|\boldsymbol{\theta})$ bestimmt. Auch in Kapitel 2.4 wurde bereits eine Ausweitung dieses Prinzips dargestellt. Durch Berücksichtigung einer Prior-Verteilung der Hyperparameter kann die Posteriori-Verteilung aufgestellt werden zu

$$p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) \propto p(\boldsymbol{X}, \boldsymbol{y}|\boldsymbol{\theta}) \cdot p(\boldsymbol{\theta}).$$

Im Rahmen des Markov-Chain-Monte-Carlo-Samplings sollen nun beliebig viele HP-Sätze von dieser Posteriori-Verteilung gezogen und superpositioniert werden. Das Prinzip von Markov-Chain-Monte-Carlo kann in drei grundsätzliche Teile unterteilt werden: Monte-Carlo-Simulation, Markov-Chain[25] und Acceptance-Rejection-Verfahren[26]. [Kat19]

Da die zu untersuchende Posteriori-Verteilung in der Regel nicht in geschlossener Form oder nur sehr aufwändig dargestellt werden kann, wird sie durch *Monte-Carlo-Simulation* modelliert. Dazu werden zufällige HP-Sätze generiert. Wird dieser Vorgang oft genug wiederholt, kann über die Auftrittshäufigkeit der Hyperparameter auf die zugrundeliegenden Wahrscheinlichkeitsverteilungen rückgeschlossen werden. [Kat19],[TRL11]

Markov-Chain stellt dabei die Repräsentation eines Agenten dar, der durch den Parameterraum "wandert". Für jeden Punkt in diesem Parameterraum können Wahrscheinlichkeiten für das Wandern des Agenten in eine bestimmte Richtung aufgestellt werden. Lässt man den Algorithmus für eine ausreichend große Anzahl von Iterationen im Parameterraum wandern, werden wahrscheinlichere Parametrierungen öfter durchlaufen als weniger wahrscheinliche. Dies geschieht ähnlich dem Durchlaufen der Trajektorie eines dynamischen Systems, wobei die korrespondierenden Zustände durch HP-Sätze verkörpert werden. Der Prozess pendelt sich so auf einen Bereich wahrscheinlicher Zustände ein. [Cul16],[Kat19]

Beim Wandern durch den Parameterraum müssen an jedem Punkt Entscheidungen getroffen werden, in welche Richtung der Agent weiter wandern soll. Das *Acceptance-Rejection-Verfahren* wird genutzt, um zu bestimmen, ob der Agent in die richtige Richtung wandert und ob er diese beibehalten oder ändern soll. [Cul16],[Kat19]

[25] dt. Markow-Kette
[26] dt. Verwerfungsmethode

Die auf diese Art und Weise generierten HP-Sätze spiegeln die Posteriori-Wahrscheinlichkeitsverteilung wieder und ermöglichen durch grobe Marginalisierung über den Parameterraum eine bessere Prädiktion. Ein Nachteil der Markov-Chain-Monte-Carlo-Methode ist der damit verbundene hohe Rechenaufwand.

3 Bayessche Optimierung

Die *Bayessche Optimierung* (BO) gehört zu den modernsten und effizientesten Optimierungsverfahren und erfährt auch in jüngster Zeit immer wieder Verbesserungen. BO wurde beispielsweise im Bereich der Parameterbestimmung Neuronaler Netze etabliert. Grundlage des Optimierungsverfahrens ist der, vom 1701 in London geborenen Mathematiker Thomas Bayes erfundene, Satz von Bayes. Dieser begründete die bayessche Wahrscheinlichkeitstheorie, bei der es im Rahmen von Wahrscheinlichkeitsberechnung zur Berücksichtigung von Vorwissen kommt (siehe Kap. 2.1.1). Im Kontext dieser Arbeit wird Bayessche Optimierung genutzt, um die Gütefunktion eines Systems zu lernen. Dazu wird sukzessive durch gezielte Auswertungen der Zusammenhang von Ein- und Ausgabegrößen erlernt (vgl. Funktionsregression in Kap. 2.4). [Bis06]

BO setzt sich grundsätzlich aus zwei Komponenten zusammen (siehe Abb. 3-1) - der GP bildet das aus Daten gewonnene Wissen ab und die Akquisitionsfunktion (AQF) motiviert progressiv die gezielte Auswertung neuer Punkte. Aufbau und Funktionsweise eines GPs wurden bereits in Kapitel 2.5 beschrieben. Das Konzept der für BO charakteristischen AQF soll nun im weiteren Verlauf dieses Kapitels näher beleuchtet werden.

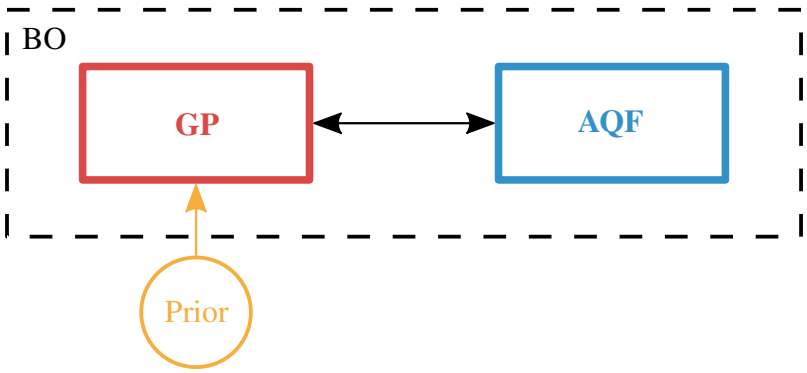
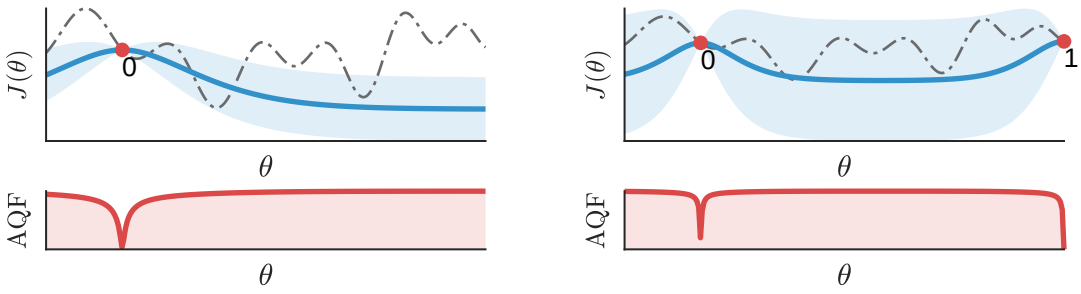


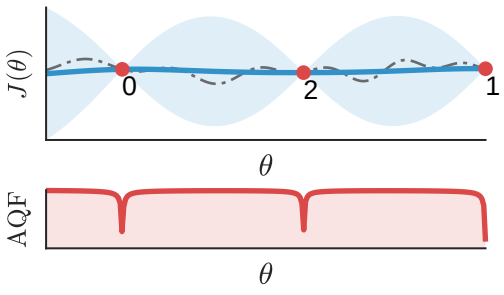
Bild 3-1: *Bayessche Optimierung kann in zwei grundlegende Komponenten untergliedert werden - GP und AQF. Der Prior wirkt in Form von Vorwissen extern auf den GP.*

Zum Einstieg soll anhand von Abbildung 3-2 der Ablauf einer BO veranschaulicht werden. In jeder Iteration wird zunächst ein neuer Messpunkt (rote Punkte) zum GP hinzufügt. Daraufhin wird in jeder Iteration der GP (jeweils blau im oberen Graph) als der auf Basis der zur Verfügung stehenden Daten wahrscheinlichste Verlauf des zu lernenden Systemverhaltens (gestrichelt in grau) prädiziert. Welcher Punkt schlussendlich am Ende einer Iteration ausgewertet wird, wird einzig von der AQF (jeweils rot im unteren Graph) vorgegeben. In diesem Fall entspricht die Stelle des Maximums der AQF der Stelle des GPs, der als nächstes evaluiert wird. Hier (Abb. 3-2) ist anschaulich zu erkennen, dass bereits nach wenigen Iterationen das Minimum hinreichend gut approximiert werden kann.

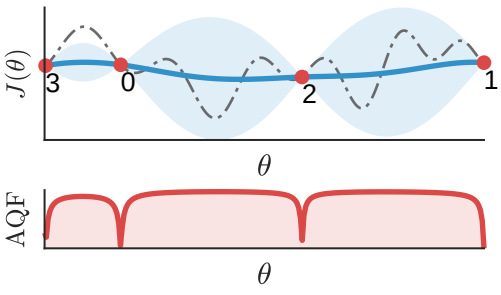


Iteration 1

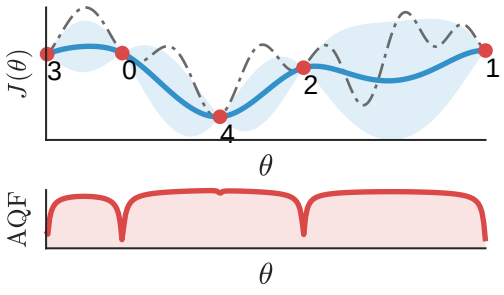
Iteration 2



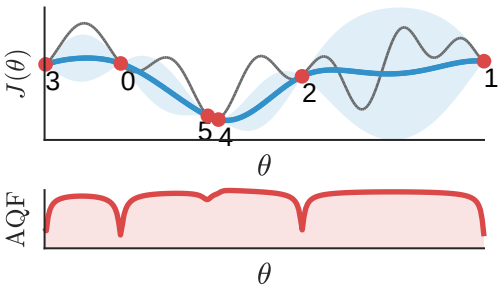
Iteration 3



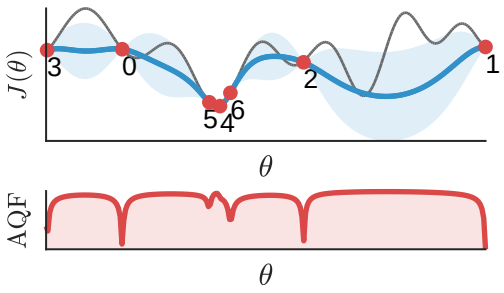
Iteration 4



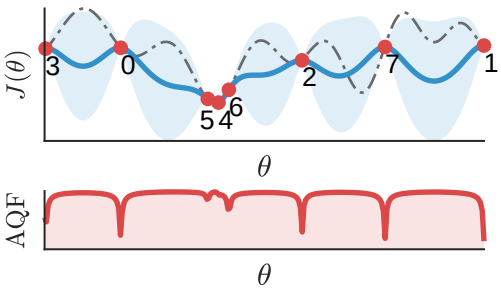
Iteration 5



Iteration 6



Iteration 7



Iteration 8

Bild 3-2: In jeder Iteration wird ein GP (in blau) gebildet, der auf Basis der Messdaten (rote Punkte) Prädiktionen über den Verlauf des realen Systems (gestrichelt in grau) macht. Durch Maximierung der AQF (in rot) wird der als nächstes auszuwertende Punkt bestimmt.

3.1 Algorithmus

In diesem Kapitel soll die Prozedur der BO detaillierter beschrieben werden. In Alg. 1 ist der zugehörige Pseudocode dargestellt. Gesucht ist eine optimale Parametrierung $\boldsymbol{\theta}^*$, die die Gütefunktion $J(\boldsymbol{\theta})$ minimiert. Zu Anfang wird eine Startparametrierung $\boldsymbol{\theta}_0$ initialisiert. Dies erfolgt entweder zufällig oder auf Basis des vorliegenden A-priori-Wissens (z.B. Prior-Minimum). In $\boldsymbol{\Theta}_i$ und $\boldsymbol{J}_i$ werden in jeder Iteration i die evaluierten Messpunkte $(\boldsymbol{\theta}_i, J_i)$ hinterlegt:

$$\boldsymbol{\Theta}_i := \begin{bmatrix} \boldsymbol{\theta}_0 \\ \vdots \\ \boldsymbol{\theta}_i \end{bmatrix}, \quad \boldsymbol{J}_i := \begin{bmatrix} J_0 \\ \vdots \\ J_i \end{bmatrix}.$$

Zur Berechnung des GPs werden zudem in jeder Iteration geeignete Hyperparameter σ_f^2, σ_n^2, $\boldsymbol{l}$ bestimmt und die Kernel-Matrix $\boldsymbol{K}$ errechnet. Durch Maximierung der Akquisitionsfunktion „aqf“ wird der als nächstes auszuwertende Punkt $\boldsymbol{\theta}_i$ gewählt. Im Laufe der Iterationen wird somit das gesammelte Datenwissen sukzessive umfangreicher, sodass zum Abschluss der Optimierung die optimale Lösung $\boldsymbol{\theta}^* := \arg\min_i J_i$ als Stelle minimaler bekannter Güte $J^* := \min_i J_i$ bestimmt werden kann.

Algorithm 1 Bayessche Optimierung (Minimierung)

```
init θ_0                                   // Wähle Startpunkt, z.B. zufällig
J_0 ← objective(θ_0)
Θ_0 ← θ_0; J_0 ← J_0
for i=1 to N do
   [σ_f^2, σ_n^2, l] ← getHypers(Θ_{i-1}, J_{i-1})
   K ← kernel(Θ_{i-1}, σ_f^2, σ_n^2, l)
   θ_i ← max_{θ∈Ω}(aqf(Θ_{i-1}, J_{i-1}, σ_f^2, σ_n^2, l, K))
   J_i ← objective(θ_i)
   Θ_i ← [Θ_{i-1}; θ_i]; J_i ← [J_{i-1}; J_i]
end for
return θ* ← Θ_N(min(J_N))
```

Da über die Iterationen betrachtet der Rechenaufwand, beispielsweise bei der Invertierung der Kernel-Matrix, immer weiter ansteigt, verlangsamt sich die Auswertung des Algorithmus sukzessive mit steigender Iterationszahl. Dies ist ein Resultat der steigenden Datenmenge. So expandiert die Anzahl der Einträge der Kernelmatrix $\boldsymbol{K}$ quadratisch (Formel (3-1)). Abhilfe können *Sparse Kernel Methods* liefern. Weitere Informationen dazu sind in [Bis06] zu finden.

$$\boldsymbol{K}_{i+1} = \begin{bmatrix} \boldsymbol{K}_i & \boldsymbol{k}^* \\ \boldsymbol{k}^{*T} & k^{**} \end{bmatrix} \tag{3-1}$$

3.2 Akquisitionsfunktionen

In diesem Kapitel sollen bekannte *Akquisitionsfunktionen*[27] (AQ-Funktionen oder kurz AQF) vorgestellt werden. Dabei soll im Hinblick auf diese Arbeit auf Vollständigkeit verzichtet werden. Umfangreiche Informationen zu anderen AQ-Funktionen wie z.B. Expected Improvement (EI), Entropy Search (ES) und Predictive Entropy Search (PES) sind in [Moc74], [HS12] und [HHG14] zu finden. Im Folgenden sollen besonders die AQ-Funktionen betrachtet werden, die entweder zum Einstieg besonders einfach nachzuvollziehen sind oder den Stand der Technik abbilden und somit zu den innovativsten und effizientesten AQ-Funktionen zum Zeitpunkt dieser Arbeit gehören.

3.2.1 Upper/Lower Confidence Bound (UCB/LCB)

Der wohl trivialste Ansatz einer AQ-Funktion $\alpha(\boldsymbol{x})$ besteht in einer gewichteten Summe aus den beiden GP-Größen Mittelwert μ und Standardabweichung σ. Dieser sogenannte *Lower Confidence Bound* (LCB) (für Maximierungsprobleme *Upper Confidence Bound* (UCB)) berücksichtigt die Unsicherheit des GPs mit einem Koeffizienten $\beta \geq 0$, der sich von Iteration zu Iteration ändern kann und so das Konvergieren des Algorithmus ermöglicht. Man spricht beim Konflikt von Erkundung[28] des Eingangsraums und Konvergenz[29] zum Minimum von *Exploration vs. Exploitation*. Für $\beta = 0$ entspricht das Minimum $\boldsymbol{x}^*$ der LCB-Funktion dem Minimum des GPs. Durch Erhöhung von β wird nun eine gewisse Exploration angeregt. [Aue02],[SKKS10]

$$\text{Lower Confidence Bound:} \quad \alpha_{LCB}(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \beta \cdot \sigma(\boldsymbol{x}),$$
$$\boldsymbol{x}^* = \min_{\boldsymbol{x} \in X} \alpha_{LCB}(\boldsymbol{x}) = \min_{\boldsymbol{x} \in X} \mu(\boldsymbol{x}) + \beta \cdot \sigma(\boldsymbol{x})$$

Auch wenn die Auswertung und Interpretation der LCB-Funktion besonders intuitiv ausfällt, steht die Performance der AQ-Funktion anderen Prinzipien deutlich nach. So spielt beispielsweise die Wahl einer geeigneten Funktion $\beta(i)$ für Iterationen $i \in [1, N]$ eine bedeutende Rolle für die Effizienz des Algorithmus.

3.2.2 Max-value Entropy Search (MES)

Zum Zeitpunkt dieser Arbeit stellt die *Max-value Entropy Search* (MES) AQ-Funktion den Stand der Technik dar. Ihre Performance ist äquivalent oder größer als die der effizientesten AQ-Derivate (siehe dazu [WJ17]). Das informationstheoretische Konzept von MES ist leicht zu veranschaulichen. Es wird davon ausgegangen, dass der Funktionswert y^* des Optimums bekannt ist bzw. ‚dass anhand einer Wahrscheinlichkeitsverteilung die wahrscheinlichsten Optima $y_1^*, \cdots, y_K^* \in Y_*$ bestimmt werden können. Für jede Stelle $\boldsymbol{x}$ des GPs mit Daten $\boldsymbol{X}, \boldsymbol{y}$ kann $f(\boldsymbol{x})$ als Wert einer Normalverteilung mit Mittelwert $\mu(\boldsymbol{x})$ und Varianz $\sigma^2(\boldsymbol{x})$ dargestellt werden. Dann approximiert $\alpha_{MES}(\boldsymbol{x})$ für jede Stelle $\boldsymbol{x}$ den

[27] engl. acquisition functions
[28] engl. exploration
[29] engl. exploitation

Flächenanteil der aufgespannten Normalverteilung $\mathcal{N}(y|\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{y})$, der durch den Funktionswert y^* "geköpft wird". Man spricht im Englischen von *truncated distribution*, die einer kumulierten Normalverteilung (CDF) mit Grenzwert y^* entspricht. Anders gesagt, wird der Anteil der an Punkten $\boldsymbol{x}$ aufgespannten Verteilungen maximiert, der „zum gesuchten Optimum y^* führt". Dabei ist $\gamma_{y_*}(\boldsymbol{x})$ hinsichtlich eines Minimierungsproblems gegeben durch

$$\gamma_{y_*}(\boldsymbol{x}) = \frac{-y_* - (-\mu(\boldsymbol{x}))}{\sigma(\boldsymbol{x})} = \frac{\mu(\boldsymbol{x}) - y_*}{\sigma(\boldsymbol{x})}. \tag{3-2}$$

und dient zur Normierung des Arguments $\boldsymbol{x}$. $\Phi(\gamma_{y_*}(\boldsymbol{x}))$ ist die Standardnormalverteilung (PDF) für y und $\Psi(\gamma_{y_*}(\boldsymbol{x}))$ die kumulierte Standardnormalverteilung (CDF). [WJ17]

$$\begin{aligned}\text{Max-value Entropy Search:}\quad \alpha_{MES}(\boldsymbol{x}) &= H(p(-y|\boldsymbol{x}, \boldsymbol{X}, -\boldsymbol{y})) - \mathbb{E}[H(p(-y|\boldsymbol{x}, \boldsymbol{X}, -\boldsymbol{y}, -y_*))] \\ &\approx \frac{1}{K}\sum_{y_* \in Y_*}\left[\frac{\gamma_{y_*}(\boldsymbol{x}) \cdot \Phi(\gamma_{y_*}(\boldsymbol{x}))}{2\Psi(\gamma_{y_*}(\boldsymbol{x}))} - \log\left(\Psi(\gamma_{y_*}(\boldsymbol{x}))\right)\right], \\ \boldsymbol{x}^* &= \max_{\boldsymbol{x} \in X} \alpha_{MES}(\boldsymbol{x})\end{aligned}$$

Zudem entspricht $\alpha_{MES}(\boldsymbol{x})$ der Überlagerung von K Verteilungen mit jeweiligem y_i^*. Es wird an dieser Stelle also auch wieder über mögliche Optimalwerte marginalisiert. Die y_i^* können z.B. über eine Wahrscheinlichkeitsverteilung gesampled werden. Zur weiteren Beschleunigung der Evaluierung der y_i^* wird eine Approximation der Wahrscheinlichkeitsverteilung durch eine Gumbel-Verteilung[30] vorgenommen. Weitere Informationen sind [WJ17] zu entnehmen.

Da in manchen Fällen der Optimalwert für das betrachtete System bekannt ist, ist es interessant $\alpha_{MES}(\boldsymbol{x})$ für ein einzelnes Optimum zu berechnen. Nach [WJ17] ist der reduzierte Ausdruck (3-3) proportional zu $\gamma_{y_*}(\boldsymbol{x})$ (Formel (3-2)) und die Maximierung des Ausdrucks entspricht der Minimierung von $\gamma_{y_*}(\boldsymbol{x})$:

$$\text{für K=1:}\quad \alpha_{MES}(\boldsymbol{x}) = \left(\frac{\gamma_{y_*}(\boldsymbol{x}) \cdot \Phi(\gamma_{y_*}(\boldsymbol{x}))}{2\Psi(\gamma_{y_*}(\boldsymbol{x}))} - \log\left(\Psi(\gamma_{y_*}(\boldsymbol{x}))\right)\right) \propto -\gamma_{y_*}(\boldsymbol{x}), \tag{3-3}$$

$$\begin{aligned}\boldsymbol{x}^* &= \max_{\boldsymbol{x}} \alpha_{MES}(\boldsymbol{x}) = \max_{\boldsymbol{x}}\left(\frac{\gamma_{y_*}(\boldsymbol{x}) \cdot \Phi(\gamma_{y_*}(\boldsymbol{x}))}{2\Psi(\gamma_{y_*}(\boldsymbol{x}))} - \log\left(\Psi(\gamma_{y_*}(\boldsymbol{x}))\right)\right) \\ &= \min_{\boldsymbol{x}} \gamma_{y_*}(\boldsymbol{x}).\end{aligned}$$

Das Minimum der resultierenden Formulierung entspricht darüber hinaus dem Minimum der AQ-Funktion Probability of Improvement (PI) (vgl. [WJ17]). Dabei ist die Wahl von y_* besonders bedeutsam. Ist der Funktionswert des tatsächlichen Minimums unbekannt und wird y_* deutlich höher als dieses gewählt, so konvergiert die BO schon für deutlich höhere Werte (in der Nähe von y_*). Wird y_* dagegen zu niedrig gewählt, konvergiert der Algorithmus nicht und erkundet den Eingangsraum nach anderen möglichen Stellen. Die Wahl von y_* entscheidet also zwischen Exploration und Exploitation. Ist der Wert des

[30] eine (stetige) spezielle Art Wahrscheinlichkeitsverteilung

realen Optimums unbekannt und kann nicht hinreichend approximiert werden, sollte wie oben beschrieben über mögliche y_* marginalisiert werden. Im Nachfolgenden soll bei der auf $K = 1$ sowie z.B. $y_* = 0$ reduzierten Version zur Vermeidung von Verwechslungen von *Max-Value Entropy Search mit Minimum 0* (MESmin0) gesprochen werden.

$$\alpha_{PI}(\boldsymbol{x}) = \Psi(\gamma_{y_*}(\boldsymbol{x})),$$

$$\alpha_{MESmin0}(\boldsymbol{x}) := \gamma_0(\boldsymbol{x}) = \frac{-0 - (-\mu(\boldsymbol{x}))}{\sigma(\boldsymbol{x})} = \frac{\mu(\boldsymbol{x})}{\sigma(\boldsymbol{x})}$$

Wie in Abbildung 3-3 zu erkennen ist, ist die Performance von MESmin0 der von LCB gerade bei der Konvergenz (Exploitation) überlegen. Daher soll bei allen folgenden Untersuchungen mit Minimum gleich 0 standardmäßig MESmin0 eingesetzt werden.

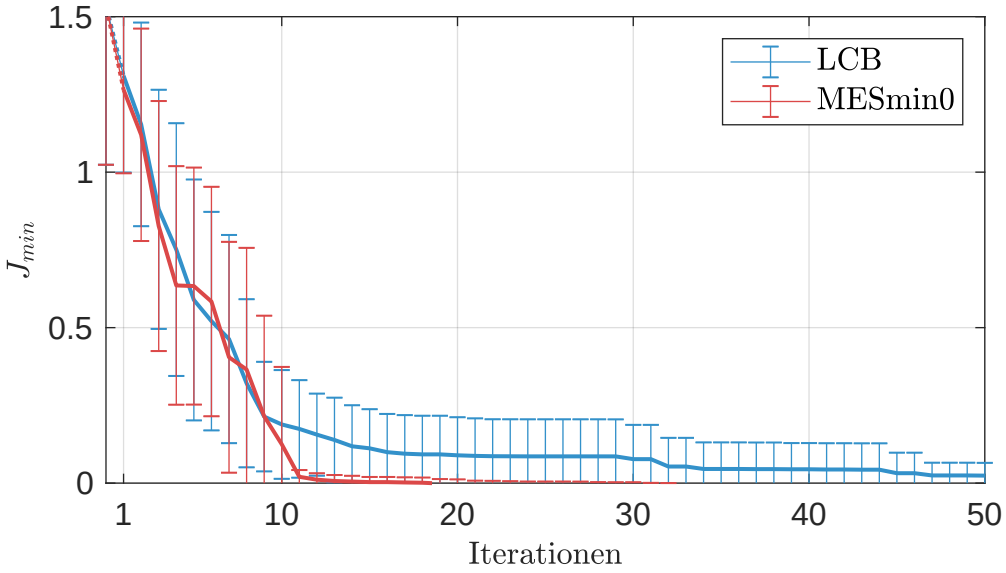


Bild 3-3: *Das Konvergenzverhalten von MESmin0 übertrifft die Performance von LCB deutlich. Für den Vergleich wurden Mittelwert und Standardabweichung von 20 Durchläufen ermittelt.*

4 Anwendungsbeispiel: Aufschwung eines starren Pendels

Um die Bedeutung von Vorwissen in Form eines physikalischen Dynamikmodells für die Bayessche Optimierung technischer Systeme zu evaluieren, soll ein akademisches Beispiel dienen, das bestimmte Anforderungen erfüllt. Ein geeignetes System ist nur unzulänglich in geschlossener Form durch ein physikalisches Dynamikmodell zu beschreiben, Gradienten bezüglich der Eingangsparameter sind nicht bekannt, die zu messenden Ausgangsgrößen sind verrauscht und oder aufwändig auszuwerten. Darüber hinaus soll im Rahmen dieser Arbeit lediglich eine Einzieloptimierung angestrebt werden und die Dimension des Eingangsraums soll mehrdimensional sein.

Um das Verhalten mit Prior-Wissen hinlänglich untersuchen zu können, soll als Beispielsystem ein starres Pendel betrachtet werden. Durch Variieren der Systemparameter kann eine Version des Systems mit Realsystem-Parametrierung mit einer Systemversion mit Prior-Parametrierung verglichen werden. Durch geeignete Wahl der Anregung (Bang-Bang-Steuerung) ist so ein mehrdimensionales Optimierungsproblem konstruierbar. Darüber hinaus wird davon ausgegangen, dass Messdaten mit einem Rauschen überdeckt sind und es gilt die Anzahl nötiger Systemauswertungen (Trainingsdaten) zu minimieren. Zunächst soll dafür ein Modell aufgestellt und eine geeignete Steuerung entworfen werden, die im Anschluss optimiert werden soll.

4.1 Physikalische Modellbildung

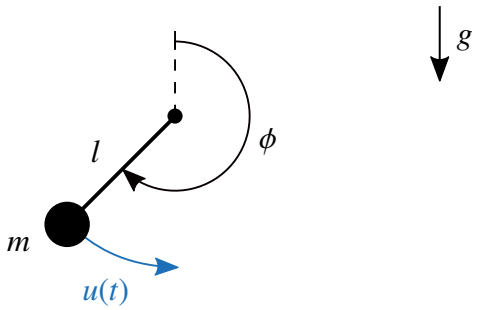


Bild 4-1: *Das starre Pendel wird durch Masse m und Länge l charakterisiert. Die Anregung u(t) erfolgt dabei als direkt angreifende Kraft im Fußpunkt.*

Als Beispiel für diese Arbeit soll, wie zuvor erwähnt, ein starres Pendel betrachtet werden (siehe Abb. 4-1). Dabei findet eine Fußpunktanregung $u(t)$ statt und Winkel $\phi(t)$ sowie Winkelgeschwindigkeit $\dot{\phi}(t)$ stellen sich ein. Bei Vernachlässigung von Dämpfungseffekten ergibt sich folgende nichtlineare Dynamikgleichung:

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

$$\Leftrightarrow \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\dot{\phi}(t)\\ \phi(t)\end{bmatrix} = \begin{bmatrix}\frac{g}{l}\sin\phi(t) + \frac{u(t)}{ml^2}\\ \dot{\phi}(t)\end{bmatrix}.$$

Es wird im Folgenden angenommen, dass die Parametrierung des realen Systems unbekannt ist und der Prior durch ein identisches Dynamikmodell mit davon abweichender Parametrierung verkörpert wird. Durch Gegenüberstellung der beiden Systeme kann der Einfluss verschiedener Prior-Konfigurationen untersucht werden. Da das starre Pendel sehr sensitiv auf Parameteränderungen reagiert, können so deutlich schwerwiegendere Abweichungen erzielt werden, als es durch die Vernachlässigung von Dynamiktermen wie der Dämpfung der Fall ist. Zunächst soll nun eine geeignete Steuerung zum Aufschwung des Pendels entworfen werden.

4.2 Steuerungsentwurf

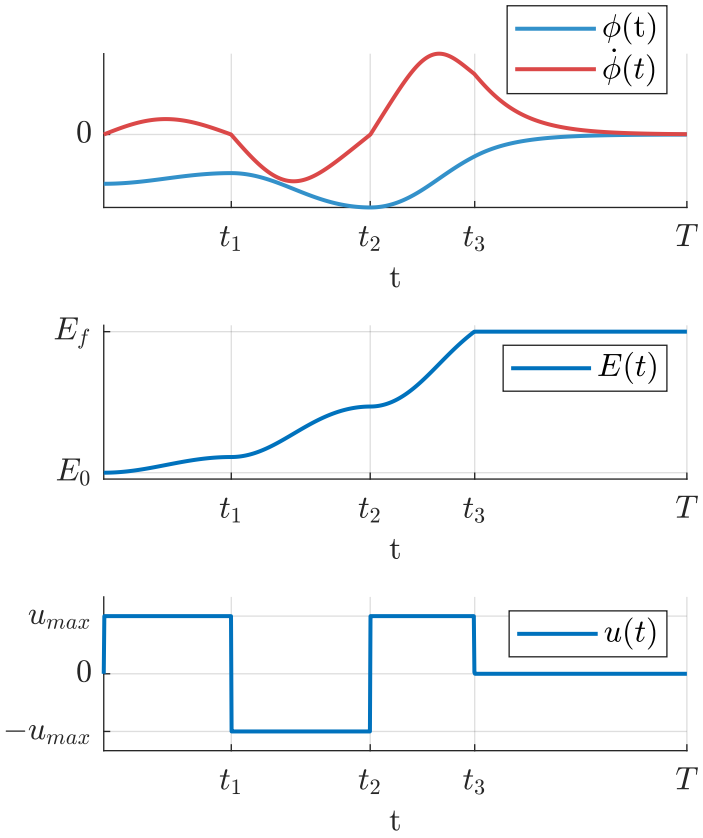


Bild 4-2: Mithilfe einer Energy Control kann das Pendel in drei Zügen aufgeschwungen werden. Eine mögliche Zustandstrajektorie (oben) ergibt sich durch die unten dargestellte Regelsequenz auf Basis der Systemenergie (Mitte).

Eine geeignete Regelung zum Aufschwung des Pendels von einem Anfangszustand x_0 in die obere Ruhelage x_f ist durch die in Kapitel 2.3 vorgestellte Energy Control gegeben. In der Endlage soll sowohl Winkel, -geschwindigkeit und Energie gleich null sein:

$$\phi(t_f) = 0, \quad \dot{\phi}(t_f) = 0, \quad E_f = 0.$$

Dabei ergibt sich die Energie zum Zeitpunkt t als

$$E(t) = mgl\left(\frac{1}{2}\left(\frac{\dot{\phi}(t)}{\omega_0}\right)^2 + \cos\phi(t) - 1\right).$$

Eine mögliche Regelung ergibt sich damit zu

$$\begin{aligned} u(t) &= \mathrm{sat}_{u_{max}}\left[k \cdot \left(E(t) - E_f\right) \cdot \mathrm{sign}\left(\dot{\phi}(t)\cos\phi(t)\right)\right] \\ &= \mathrm{sat}_{u_{max}}\left[kmgl\left(\frac{1}{2}\left(\frac{\dot{\phi}(t)}{\omega_0}\right)^2 + \cos\phi(t) - 1\right)\right] \cdot \mathrm{sign}(\dot{\phi}(t)\cos\phi(t)). \end{aligned}$$

Die Steuerbarkeit des Systems ist zu jedem Zeitpunkt gegeben, da u unabhängig vom Zustand des Pendels angreifen kann. Eine mögliche Trajektorie ist oben in Abbildung 4-2 abgebildet. Das Pendel schwingt drei mal hin und her und wird durch den Proportionalregler für kleine Regelfehler in der oberen Ruhelage stabilisiert. Die dafür notwendige Energie $\Delta E = E_f - E_0$ wird progressiv zugeführt (siehe Abb. 4-2 Mitte). Die zugehörige Steuertrajektorie (Abb. 4-2 unten) ähnelt in großen Teilen einer Rechteckfunktion und weicht nur für kleine Regelfehler davon ab. Im Folgenden soll daher eine Steuerung eingesetzt werden, eine sogenannte Triple-Switch-Control[31] (vgl. [ÅF96]). Da die optimale Einstellung der Steuerung durch Bayessche Optimierung erfolgen soll, wird so ferner die dafür notwendige gezielte Beeinflussung der Steuertrajektorie ermöglicht. Dafür wird die Steuertrajektorie in drei Intervalle unterteilt, in denen jeweils abwechselnd maximale Stellgrößen aufgebracht werden (vgl. Formel (4-1)). Da eine derartige Rechteckfunktion nicht stetig und somit nicht differenzierbar ist, wird sich erneut einer Approximation bedient, gegeben durch Gleichung (4-2). Es ergibt sich ein dreidimensionales Optimierungsproblem für die Optimierungsvariable $\boldsymbol{\theta} := [\theta_1, \theta_2, \theta_3]^T$ mit $\theta_1 := t_1$, $\theta_2 := t_1 + t_2$ und $\theta_3 := t_1 + t_2 + t_3$.

$$u_{Rechteck}(t) = \begin{cases} u_{max}, & \text{wenn } x \in \{[0, t_1), [t_2, t_3)\} \\ -u_{max}, & \text{wenn } x \in [t_1, t_2) \\ 0, & \text{sonst,} \end{cases} \tag{4-1}$$

$$u_{approx}(t, \boldsymbol{\theta}) = 2u_{max}\left(\mathcal{A}(t, 0) - \mathcal{A}(t, \theta_1) + \mathcal{A}(t, \theta_1 + \theta_2) - \frac{\mathcal{A}(t, \sum_i \theta_i)}{2} - \frac{1}{2}\right), \tag{4-2}$$

$$\text{mit } \mathcal{A}(t, \xi) = \frac{1}{1 + \exp\{50(\xi - t)\}}$$

In Abbildung 4-3 ist eine mögliche Steuertrajektorie bei Nutzung der approximierten Steuerung dargestellt. Der Verlauf der Rechteckfunktion wird hinreichend genau angenähert. Zur Bewertung der resultierenden Endlage des realen Systems $\boldsymbol{x}_{f,mess}(\boldsymbol{\theta})$ bzw. des Prior-Systems $\boldsymbol{x}_{f,sim}(\boldsymbol{\theta})$ muss eine geeignete Gütefunktion gewählt werden. Da lediglich die Abweichung der Endlage von Bedeutung ist, ergibt sich eine mögliche Gütefunktion als

$$J(\boldsymbol{\theta}) = \sum_2 (\boldsymbol{x}_f - \boldsymbol{x}_{f,mess/sim}(\boldsymbol{\theta}))^2.$$

Durch Wahl der Gütefunktion wurde das Optimierungsproblem festgelegt. Zur Optimierung muss nun die AQF betrachtet werden.

[31] dt. dreifach umschaltende Steuerung

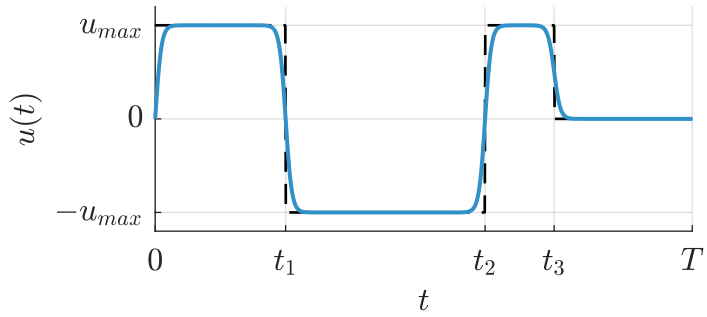


Bild 4-3: *Die verwendete Triple-Switch-Control in Form einer Rechteckfunktion (gestrichelt in grau) kann durch $u_{approx}(t, \boldsymbol{\theta})$ (in blau) hinreichend genau approximiert werden. Dadurch wird die Stetigkeit der Steuertrajektorie gewährleistet.*

4.3 Akquisitionsfunktion

In jedem Iterationsschritt muss das Extremum der AQF ermittelt werden. Dazu bieten sich unter anderem Grid-Search und gradientenbasierte Optimierungsverfahren an. Der dazu nötige Gradient der AQF bezüglich der Optimierungsvariablen $\boldsymbol{\theta}$ kann entweder durch die Finite-Gradienten-Methode oder analytisch berechnet werden. Im Nachfolgenden soll der analytische Gradient der AQF hergeleitet werden, wobei der Gradient des Priors bestimmt werden muss. Dieser ist immer vom betrachteten System abhängig und somit in diesem Fall nur für das starre Pendel gültig.

4.3.1 Gradient der Akquisitionsfunktion

Der Gradient der AQ-Funktionen LCB sowie MESmin0 ist direkt abhängig vom Gradienten des Posteriors, der sich bestimmen lässt durch

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mu^{+}(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \left[\mu^{-}(\boldsymbol{\theta}) + \boldsymbol{k}^{*}(\boldsymbol{\theta})^{T} \boldsymbol{K}^{-1} \left(\boldsymbol{J}(\boldsymbol{\Theta}) - \boldsymbol{\mu}^{-}(\boldsymbol{\Theta}) \right) \right] \\
&= \nabla_{\boldsymbol{\theta}} \mu^{-}(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} \boldsymbol{k}^{*}(\boldsymbol{\theta})^{T} \boldsymbol{K}^{-1} \left(\boldsymbol{J}(\boldsymbol{\Theta}) - \boldsymbol{\mu}^{-}(\boldsymbol{\Theta}) \right), \\
\nabla_{\boldsymbol{\theta}} \sigma^{+}(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \left[k^{**} - \boldsymbol{k}^{*}(\boldsymbol{\theta})^{T} \boldsymbol{K}^{-1} \boldsymbol{k}^{*}(\boldsymbol{\theta}) \right] \\
&= -\nabla_{\boldsymbol{\theta}} \boldsymbol{k}^{*}(\boldsymbol{\theta})^{T} (\boldsymbol{K}^{-1} + (\boldsymbol{K}^{-1})^{T}) \cdot \boldsymbol{k}^{*}(\boldsymbol{\theta}).
\end{aligned}$$

Um die Abhängigkeit der Matrix $\boldsymbol{k}^{*}$ von der Optimierungsvariablen $\boldsymbol{\theta}$ zu verdeutlichen, wird an dieser Stelle von der expliziten Schreibweise Gebrauch gemacht. Außerdem werden die Datenpunkte durch den Index „$\mathcal{D}$“ von der Optimierungsvariablen abgegrenzt. Die Trainingsdaten $\boldsymbol{\Theta}$, $\boldsymbol{J}(\boldsymbol{\Theta})$, sowie zugehörige Prior-Werte, sind hinsichtlich des Gradienten konstant. Es gilt also nur noch die Betrachtung des Gradienten von Prior und $\boldsymbol{k}^{*}$.

Zweiterer ist abhängig von der Wahl der Kernelfunktion. Alle in dieser Arbeit betrachteten Kernelfunktionen sind mindestens einmal stetig differenzierbar.

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\theta}_{\mathcal{D}1} \\ \boldsymbol{\theta}_{\mathcal{D}2} \\ \vdots \\ \boldsymbol{\theta}_{\mathcal{D}P} \end{bmatrix}, \quad \boldsymbol{J}(\boldsymbol{\theta}) = \begin{bmatrix} J(\boldsymbol{\theta}_{\mathcal{D}1}) \\ J(\boldsymbol{\theta}_{\mathcal{D}2}) \\ \vdots \\ J(\boldsymbol{\theta}_{\mathcal{D}P}) \end{bmatrix}, \quad \boldsymbol{\mu}^-(\boldsymbol{\theta}) = \begin{bmatrix} \mu^-(\boldsymbol{\theta}_{\mathcal{D}1}) \\ \mu^-(\boldsymbol{\theta}_{\mathcal{D}2}) \\ \vdots \\ \mu^-(\boldsymbol{\theta}_{\mathcal{D}P}) \end{bmatrix}, \quad \nabla_{\boldsymbol{\theta}} \boldsymbol{k}^*(\boldsymbol{\theta}) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}} \mathrm{k}(\boldsymbol{\theta}_{\mathcal{D}1}, \boldsymbol{\theta}) \\ \nabla_{\boldsymbol{\theta}} \mathrm{k}(\boldsymbol{\theta}_{\mathcal{D}2}, \boldsymbol{\theta}) \\ \vdots \\ \nabla_{\boldsymbol{\theta}} \mathrm{k}(\boldsymbol{\theta}_{\mathcal{D}P}, \boldsymbol{\theta}) \end{bmatrix}$$

Der Gradient des Priors ergibt sich als Gradient der Gütefunktion bezüglich des physikalischen Prior-Dynamikmodells zu

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mu^-(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \left[\sum_2 (\boldsymbol{x}_f - \boldsymbol{x}_{f,sim}(\boldsymbol{\theta}))^2 \right] \\ &= \nabla_{\boldsymbol{\theta}} (\phi_f - \phi_{f,sim}(\boldsymbol{\theta}))^2 + \nabla_{\boldsymbol{\theta}} (\dot{\phi}_f - \dot{\phi}_{f,sim}(\boldsymbol{\theta}))^2 \\ &= -2(\phi_f - \phi_{f,sim}(\boldsymbol{\theta})) \cdot \nabla_{\boldsymbol{\theta}} \phi_{f,sim}(\boldsymbol{\theta}) - 2(\dot{\phi}_f - \dot{\phi}_{f,sim}(\boldsymbol{\theta})) \cdot \nabla_{\boldsymbol{\theta}} \dot{\phi}_{f,sim}(\boldsymbol{\theta}) \\ &= -2(\boldsymbol{x}_f - \boldsymbol{x}_{f,sim}(\boldsymbol{\theta}))^T \cdot \nabla_{\boldsymbol{\theta}} \boldsymbol{x}_{f,sim}(\boldsymbol{\theta}). \end{aligned}$$

Es zeigt sich, dass der Gradient des Priors nur vom Gradienten der Endlage abhängt. Dieser kann seinerseits wiederum entweder durch Finite Gradienten oder analytisch evaluiert werden. Die analytische Bestimmung des Gradienten bedingt die Zustandsdiskretisierung für endliche Zeitschritte t_k, da der Gradient nicht in geschlossener Form formuliert werden kann.

4.3.2 Zustandsdiskretisierung

Zur Berechnung des Gradienten der Endlage bezüglich der Optimierungsparameter $\boldsymbol{\theta}$ wird die Zustandstrajektorie in N Schritte diskretisiert. Der Gradient kann dementsprechend als Ableitung des letzten Elements rückpropagierend der Trajektorie formuliert werden:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \boldsymbol{x}_{f,sim} &= \frac{\mathrm{d}\boldsymbol{x}_N(\boldsymbol{x}_{N-1}, u_{N-1})}{\mathrm{d}\boldsymbol{\theta}} \\ &= \frac{\partial \boldsymbol{x}_N(\boldsymbol{x}_{N-1}, u_{N-1})}{\partial \boldsymbol{x}_{N-1}} \cdot \frac{\mathrm{d}\boldsymbol{x}_{N-1}}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial \boldsymbol{x}_N(\boldsymbol{x}_{N-1}, u_{N-1})}{\partial u_{N-1}} \cdot \frac{\mathrm{d}u_{N-1}}{\mathrm{d}\boldsymbol{\theta}} \\ &= \cdots \\ &= \sum_{k=0}^{N-1} \left(\prod_{m=k+1}^{N-1} \frac{\partial \boldsymbol{x}_{m+1}}{\partial \boldsymbol{x}_m} \right) \frac{\partial \boldsymbol{x}_{k+1}}{\partial u_k} \cdot \frac{\mathrm{d}u_k}{\mathrm{d}\boldsymbol{\theta}}. \end{aligned}$$

Um die involvierten partiellen Ableitungen lösen zu können, muss ein Single-Step-Solver[32] gewählt werden. Dieser dient zur schrittweisen Lösung der Dynamikdifferentialgleichung $\dot{\boldsymbol{x}}_k = \boldsymbol{f}(t_k, \boldsymbol{x}_k, \boldsymbol{u}_k)$. In Betracht kommen dafür z.B. das explizite *Euler-Verfahren* oder das

[32] dt. Einzelschritt-Löser

Runge-Kutta-Verfahren, das an dieser Stelle nicht weitergehend beschrieben werden soll. Weitere Informationen dazu sind in [Tec] zu finden. Wird die Schrittweite $h := t_{k+1} - t_k$ klein genug gewählt, können mit dem expliziten Euler-Verfahren hinreichend genaue Lösungen erzeugt werden.

$$\text{explizites Euler-Verfahren:} \quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + h \cdot \boldsymbol{f}(t_k, \boldsymbol{x}_k, \boldsymbol{u}_k), \qquad k = 0, 1, \cdots, N$$

$$\text{Runge-Kutta-Verfahren:} \quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + h \sum_{j=1}^{N} \beta_j \boldsymbol{k}_j, \qquad k = 0, 1, \cdots, N,$$

$$\text{mit } \boldsymbol{k}_j = \left(t_k + \gamma_j h, \boldsymbol{x}_k + h \sum_{l=1}^{N} \alpha_{j,l} \boldsymbol{k}_l \right)$$

Mit dem expliziten Euler-Verfahren ergeben sich die partiellen Ableitungen der Zustände zu

$$\frac{\partial \boldsymbol{x}_{m+1}}{\partial \boldsymbol{x}_m} = \begin{bmatrix} 1 & \frac{gh}{l}\cos(\phi_m) \\ h & 1 \end{bmatrix},$$

$$\frac{\partial \boldsymbol{x}_{k+1}}{\partial u_k} = \begin{bmatrix} 0 & \frac{h}{ml^2} \end{bmatrix}.$$

Die Ableitung der Steuergröße u_k bezüglich $\boldsymbol{\theta}$ kann durch Einsatz der Approximationslösung u_{approx} aus Kapitel 4.2 einfach bestimmt werden:

$$\begin{aligned} \frac{\mathrm{d}u_k}{\mathrm{d}\boldsymbol{\theta}} &= \frac{\mathrm{d}u_{approx}(t_k, \boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} \\ &= u_{max} \cdot (2 \cdot \frac{\mathrm{d}\mathcal{A}(t, 0)}{\mathrm{d}\boldsymbol{\theta}} - 2 \cdot \frac{\mathrm{d}\mathcal{A}(t, \theta_1)}{\mathrm{d}\boldsymbol{\theta}} + 2 \cdot \frac{\mathrm{d}\mathcal{A}(t, \theta_1 + \theta_2)}{\mathrm{d}\boldsymbol{\theta}} - \frac{\mathrm{d}\mathcal{A}(t, \sum_i \theta_i)}{\mathrm{d}\boldsymbol{\theta}} - 1). \end{aligned}$$

4.4 Untersuchungen

In diesem Kapitel sollen die Ergebnisse der Untersuchungen präsentiert werden, die hinsichtlich der Effizienzsteigerung unter Berücksichtigung A-priori-Wissens angestellt wurden. Zunächst wird dafür das Optimierungsproblem auf ein eindimensionales Problem reduziert, um die dabei auftretenden Effekte besser darstellen und beurteilen zu können.

4.4.1 Eindimensionaler Fall

Die Reduktion des multidimensionalen Optimierungsproblems erfolgt durch Vorgabe der ersten beiden Parameter θ_1, θ_2. Die verbleibende Größe θ_3 soll nun durch Bayessche Optimierung iterativ bestimmt werden. Dazu werden folgende Fälle betrachtet - Optimierung ohne Vorwissen und Optimierung mit unterschiedlich gut ausgeprägtem Vorwissen.

Es zeigt sich, dass die Gütefunktion des Pendels einen besonders ungünstigen Verlauf hat (siehe Abb. 4-4). So handelt es sich um ein für Optimierungsverfahren besonders schlecht

konditioniertes Optimierungsproblem, da das Minimum im Funktionsverlauf „versteckt“ ist und die Kurve erst ausgiebig exploriert werden muss. Da die Güte des gesuchten Minimums bekannt ist ($J^* = 0$), findet, sobald ein hinreichendes Minimum gefunden wird, Konvergenz des Algorithmus statt. Wichtig bleibt dennoch die möglichst enge Eingrenzung des Suchbereichs, um das Verhältnis von Exploration zu Exploitation zu kippen. Dies ist in jedem Fall hinsichtlich höherdimensionaler Optimierungsprobleme zu beachten. Ist der Algorithmus zu lange mit der Abtastung des Suchraums beschäftigt, lohnt es sich, diesen weiter zu beschränken.

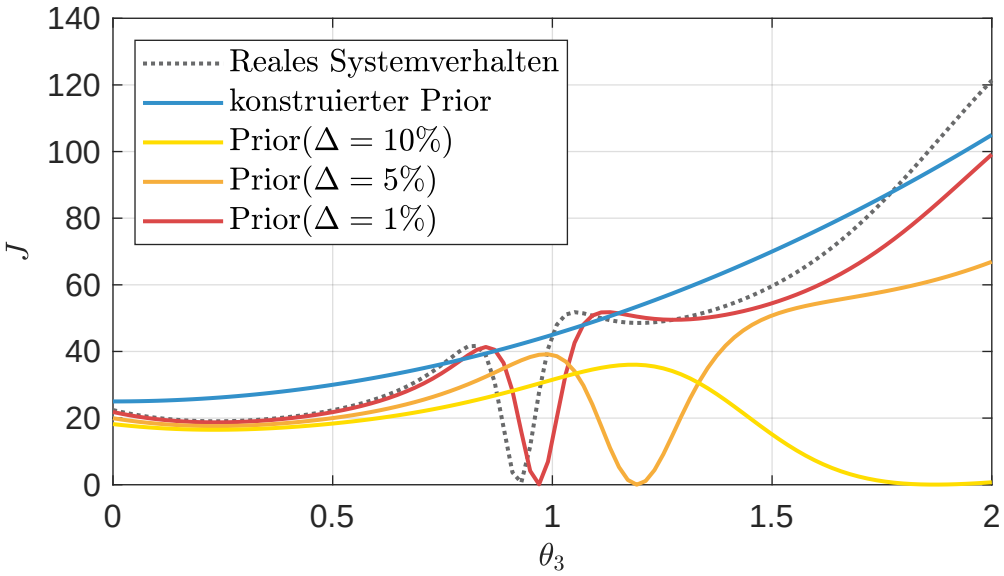
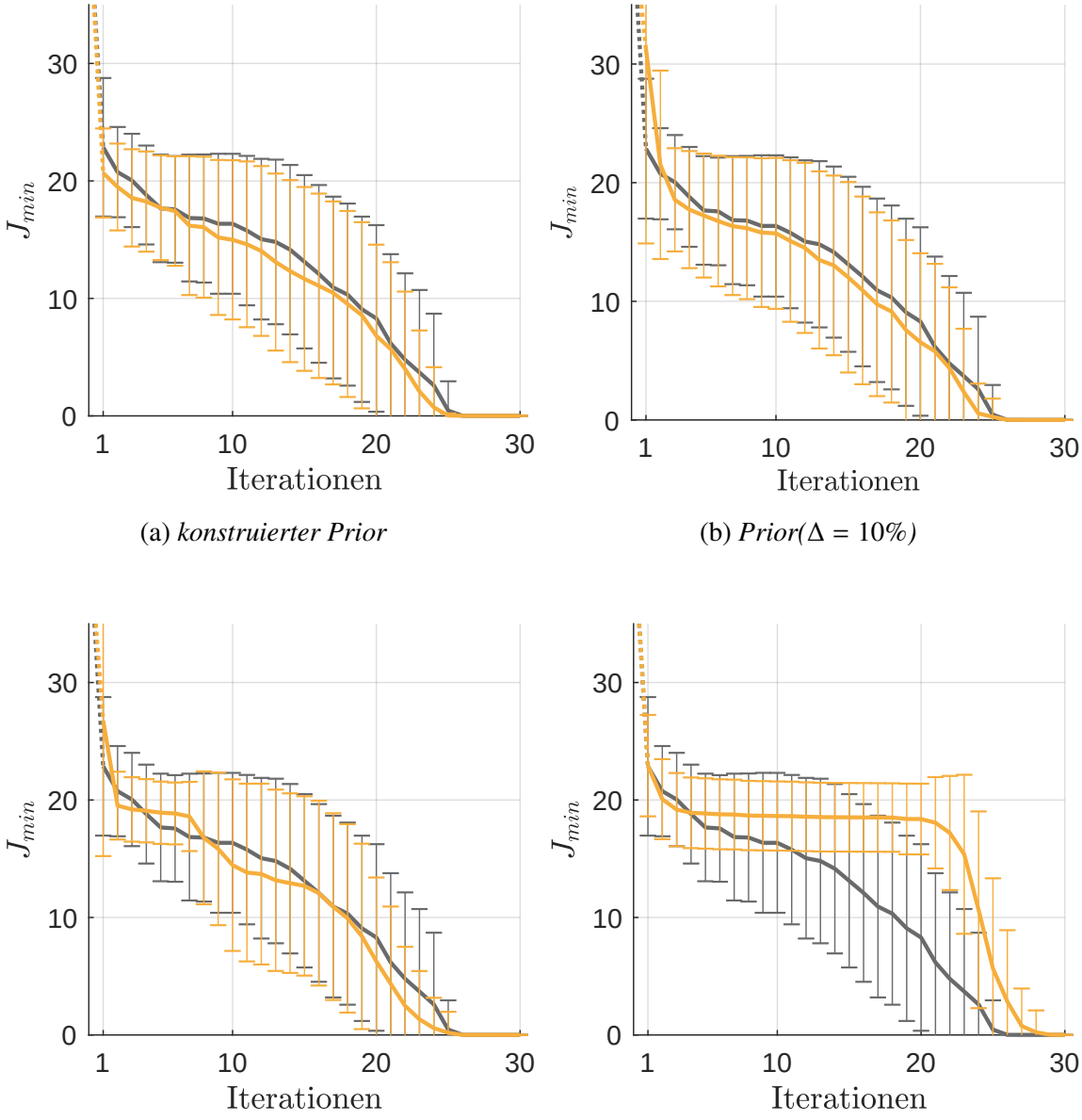


Bild 4-4: *Abgebildet ist der Verlauf verschiedener Prior-Konfigurationen mit entsprechender Abweichung ggü. dem realem System. Zudem wurde ein Prior konstruiert, der den Verlauf des realen Systems grob annähert.*

Der Einsatz von Vorwissen erzielt im eindimensionalen Fall durchwachsene Ergebnisse. Das Systemverhalten reagiert sehr sensitiv auf Änderung der Parameter. Das ist besonders anschaulich anhand der Prior-Konfigurationen in Abbildung 4-4 zu sehen. Dabei wurden jeweils die Parameter Länge und Masse des Prior-Systems um Δ-Prozent erhöht. Schon bei einer minimalen Abweichung von 1% deckt sich das reale Minimum nicht mehr mit dem Prior-Minimum. Für höhere Abweichungen ergeben sich deutlich verschiedenartige Verläufe. Es zeigt sich, dass besonders die Prior-Konfigurationen mit Minimum in der direkten Nähe, aber nicht hinreichend nah, des wahren Minimums zu einer deutlich verzögerten Konvergenz des Optimierungsverfahrens führt (siehe Abb. 4-5d). Die Prior-Varianten mit höherer Parameterabweichung vom realen System zeigen dagegen sogar bessere Konvergenzeigenschaften. Sie bieten zwar kaum einen Mehrwert zur No-Prior-Version (siehe Abb. 4-6), bringen aber auch keine fehlleitenden Informationen in direkter Nähe des realen Minimums ein. Neben den Prior-Konfigurationen, die durch Parametervariierung erzeugt wurden, wurde auch ein konstruierter Prior betrachtet, der lediglich den groben Verlauf des realen Systems abbilden soll. Auch mit diesem Prior zeigt sich kein bedeutender Vorteil zur No-Prior-Konfiguration. Von allen betrachteten Prior-Varianten ist dieser dennoch der Effektivste.



(a) *konstruierter Prior* (b) *Prior*($\Delta = 10\%$)

(c) *Prior*($\Delta = 5\%$) (d) *Prior*($\Delta = 1\%$)

Bild 4-5: *Vergleich des Konvergenzverhaltens der Prior-Konfigurationen (in gelb) mit variierender Parameterabweichung vom realen System ggü. No-Prior-Konfiguration (in grau). Dargestellt sind Mittelwert und Standardabweichung (68%-Bereich) für 100 Durchläufe. Dabei wurden $\theta_1 = konst.$, $\theta_2 = konst.$ vorgegeben.*

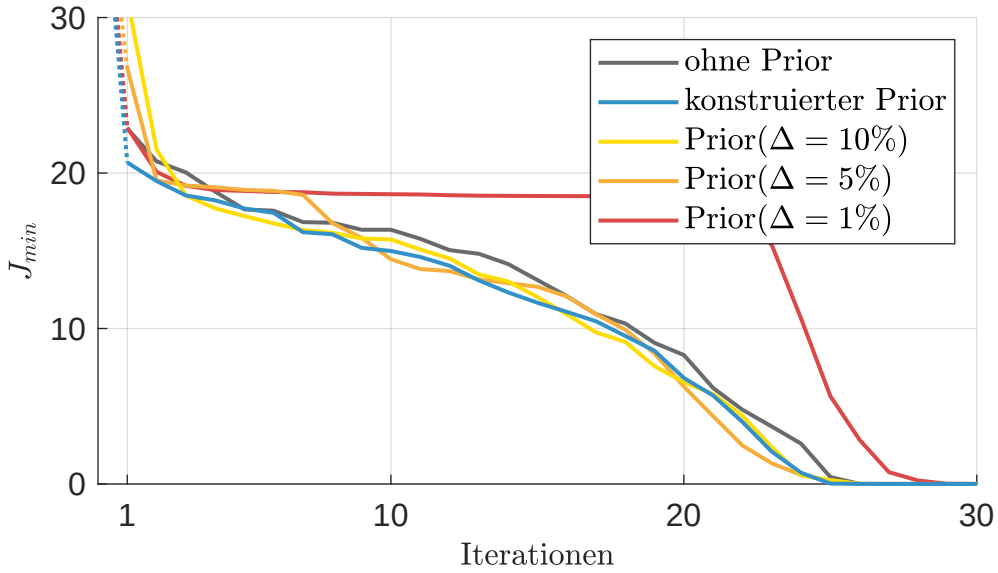


Bild 4-6: *Das Konvergenzverhalten der BO bei Verwendung unterschiedlicher Prior-Varianten ist im Eindimensionalen ähnlich. Die Variante mit einer Abweichung von nur einem Prozent fällt dabei aus der Reihe und zeigt ein besonders schlechtes Konvergenzverhalten.*

Erklären lässt sich dieser Effekt anhand einer genaueren Betrachtung des GPs. In Abbildung 4-7 ist der GP mit zufällig gewähltem Anfangswert (Punkt mit Index 0) in der zweiten Iteration dargestellt. Es ist erneut zu erkennen, dass das Minimum des Priors (in gelb) in direkter Nähe des Minimums des realen Systemverhaltens (in grau) liegt. Die Auswertungen (rote Punkte) werden iterativ durch Maximierung der AQF (in rot) gewählt und erzeugen den GP (in blau). Dieser wird mit Mittelwert und einfacher Standardabweichung dargestellt. Die zweite Auswertung (Punkt mit Index 1) wird an der Stelle des Prior-Minimums durchgeführt. Obwohl der entsprechende Wert des realen Systems an dieser Stelle deutlich höher als die Prädiktion des Priors ausfällt, versucht der GP den Rest des Verlaufes vom Prior zu übernehmen. Dies führt dazu, dass an ebendieser Stelle weiterhin ein lokales Minimum angenommen wird, nur mit deutlich höherem Wert. Erst in späteren Iterationen wird dieser Fehler korrigiert, wie anhand von Abbildung 4-5d ersichtlich ist (der deutliche Abfall von J_{min} findet erst in späten Iterationen statt). Dadurch werden links und rechts neben dem lokalen Minimum höhere Werte prädiziert. Da der Algorithmus vor allem tiefe Funktionswerte auswertet, steht somit der Bereich des lokalen Minimums nicht mehr im Fokus der Auswertung und das reale Minimum wird „übersehen“.

Es lässt sich folgender Zusammenhang formulieren. Der Prior kann in zweierlei Form auf den Posterior wirken. Zum einen kann der Prior als „Schablone“ dienen. Der Verlauf wird direkt in den Posterior übernommen und nur die Werte, sprich die Höhe der Funktionswerte $J(\boldsymbol{x}_i)$ werden angepasst. Die zweite Wirkungsweise zeigt sich erst in späten Iterationen. Sind genügend Auswertungen in einem Bereich des Eingangsraums erfolgt, wird der Verlauf des GPs dem realen Verlauf angepasst. Der Prior spielt dabei so gut wie keine Rolle mehr. Es zeigt sich also, dass Prior-Informationen besonders für frühe Iterationen von Bedeutung sind und fehlleitende Prior-Bestandteile zu einer deutlich ver-

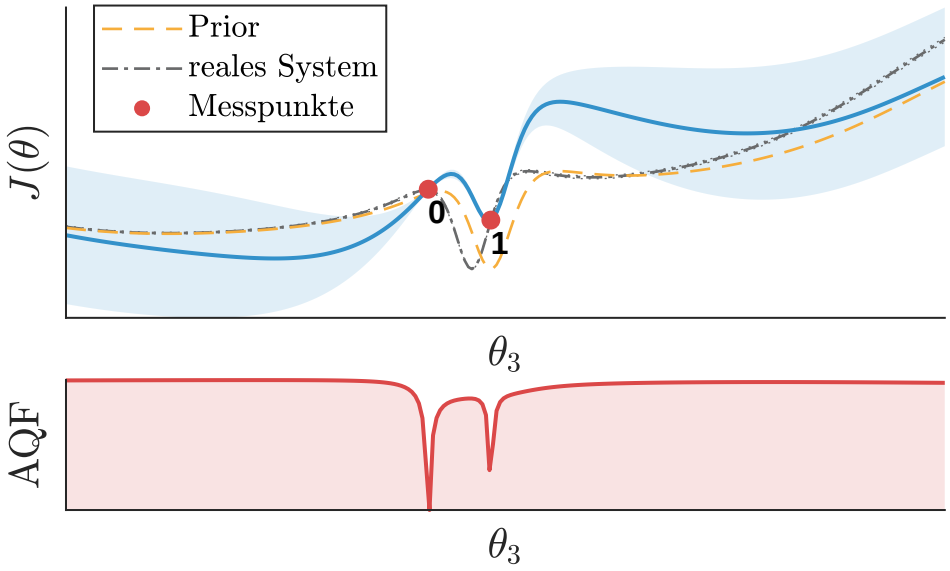


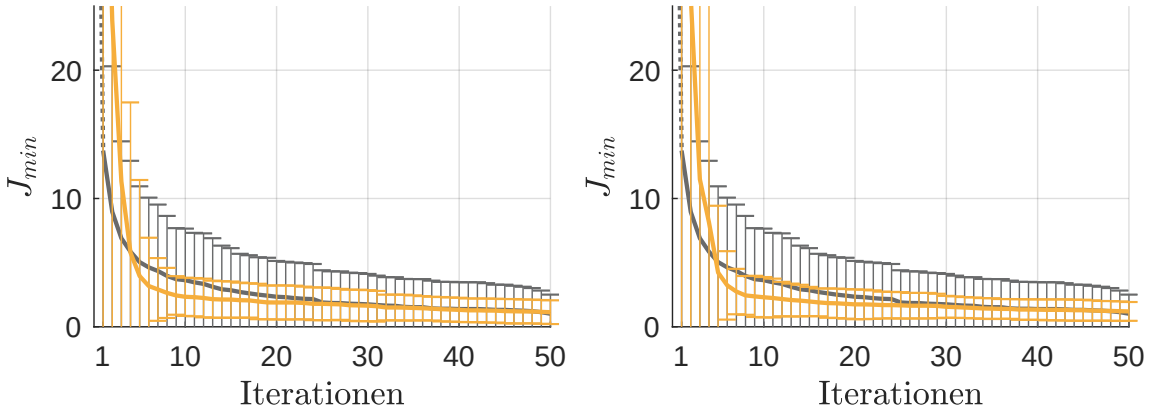
Bild 4-7: *Dargestellt ist die zweite Iteration eines beispielhaften Durchlaufs der BO. Trotz der zweiten Auswertung behält der GP (in blau) den Verlauf des Priors bei.*

zögerten Konvergenz führen. Es sollte demnach besser nur ein grober Verlauf in Form des Priors übergeben werden, als falsche Minima vorzugeben. In der Realität ist das allerdings oft nur eingeschränkt umsetzbar. Dennoch können oft Aussagen über den groben Verlauf des realen Systems gemacht werden. Z.B. sind Randwerte bekannt und ein grober Verlauf kann interpoliert werden. Darüber hinaus wird per Definition ohne Prior-Vorgabe ein Prior gleich null angenommen. Dies führt dazu, dass der Algorithmus, der nach einem Minimum mit Wert null sucht, direkt versucht zu konvergieren.

4.4.2 Dreidimensionaler Fall

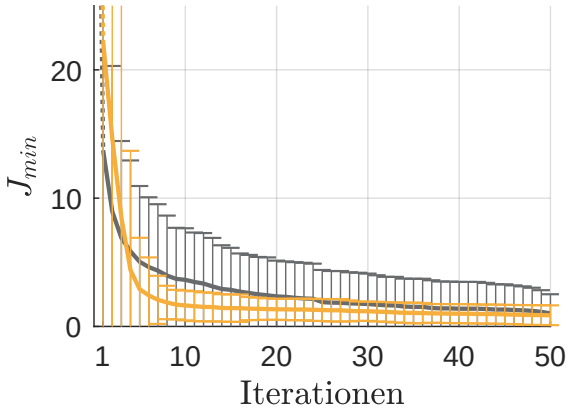
In diesem Kapitel soll der Aufschwung des Pendels in vollem Umfang durch Bayessche Optimierung erfolgen. Dabei soll $J(\boldsymbol{\theta})$ für $\boldsymbol{\theta} \in \Omega \subseteq \mathbb{R}^3$ global minimiert werden. Dazu wird erneut die BO ohne Vorwissen mit der BO mit Prior variierender Güte verglichen. Die Prior-Konfigurationen ergeben sich wie in Kapitel 4.4.1 durch Variation der Systemparameter Länge l und Masse m um Δ-Prozent von der realen Parametrierung.

Es zeigt sich, dass im mehrdimensionalen Fall im Kontrast zum eindimensionalen Fall die Wirksamkeit des Priors proportional zur Abweichung vom realen System ist. Prior-Konfigurationen mit besonders kleinen Abweichungen von 0,5-2% weisen eine deutlich reduzierte Zahl von Optimierungsiterationen ggü. der No-Prior-Konfiguration auf (siehe Abb. 4-8). Ab einer Abweichung von ca. 10% von der realen Systemparametrierung hat der Prior nur noch marginale Vorteile ggü. No-Prior, steigende Abweichungen scheinen zudem keine weitere Verschlechterung des Konvergenzverhaltens zu induzieren. Es scheint sich um eine Art Deckelung der den Algorithmus hemmenden Wirksamkeit des Priors zu handeln. Dies kann jedoch nicht für zunehmend genauere Prior-Konfigurationen gesagt werden. So scheint es, dass zunehmend bessere Prior-Daten zu einer immer besseren Konvergenz führen. Hinsichtlich des betrachteten Pendels sollte eine hinreichen-

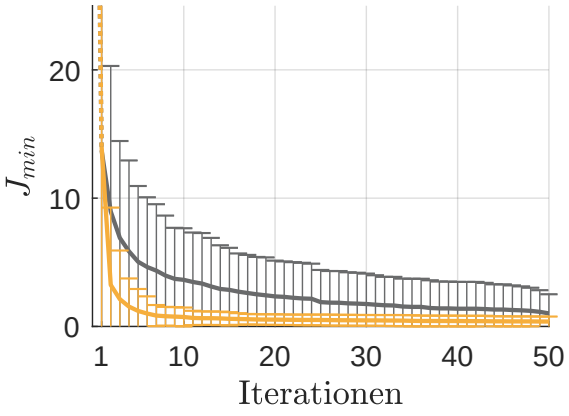


(a) *Prior*($\Delta = 20\%$)

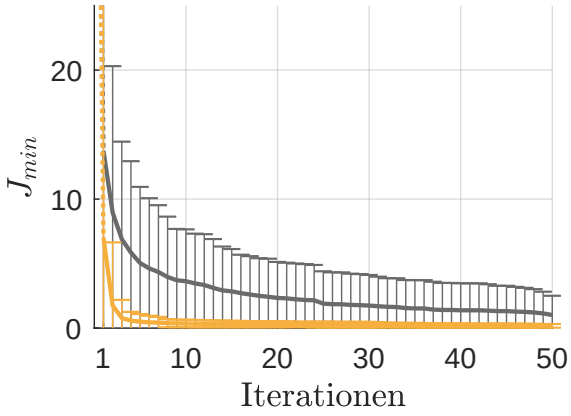
(b) *Prior*($\Delta = 10\%$)



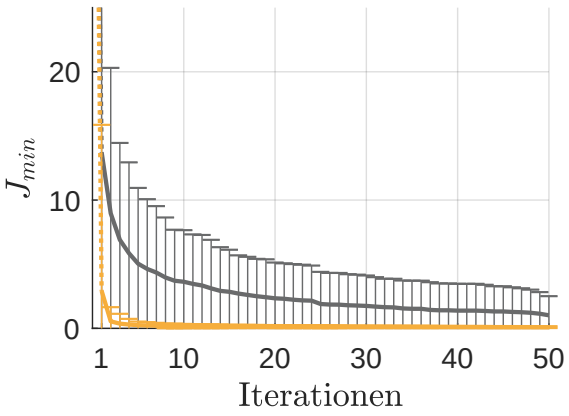
(c) *Prior*($\Delta = 5\%$)



(d) *Prior*($\Delta = 2\%$)



(e) *Prior*($\Delta = 1\%$)



(f) *Prior*($\Delta = 0,5\%$)

Bild 4-8: *Die Konvergenz des Algorithmus wurde für 50 Durchläufe untersucht und in Form von Mittelwert und 2-facher Standardabweichung (95%-Bereich) abgebildet. Dabei zeigt sich, dass die Prior-Konfiguration (in gelb) ggü. der No-Prior-Konfiguration (in grau) deutlich performanter sind.*

de Prior-Parametrierung jedoch als gewährleistet angesehen werden. Darüber hinaus ist zu erkennen, dass für $\Delta \gtrapprox 2\%$ innerhalb der ersten Iterationen eine Verschlechterung des Konvergenzverhaltens ggü. No-Prior zu verzeichnen ist. Die beinhalteten Minima-Informationen decken sich nominell nicht mehr hinreichend mit den realen Minima. Dieser negative Effekt wird jedoch in allen untersuchten Fällen frühzeitig ausgeglichen und es kommt sogar zu einer Überkompensation. Wie bereits im eindimensionalen Fall beobachtet, spielt das Prior-Wissen für spätere Iterationen kaum mehr eine Rolle. Hinsichtlich der Varianz der Ergebnisse für zufällig gewählte Startpunkte $\boldsymbol{\theta}_0$ lässt sich zudem ein Muster erkennen. Die Hinzunahme von Prior-Wissen reduziert die auftretende Varianz der Ergebnisse deutlich. Es kommt zu einer Reduktion der „Willkür“ der BO und somit zu einer zielgerichteteren Konvergenz des Optimierers. Dieser Effekt kann jedoch auch nur im Hinblick auf frühe Iterationen verzeichnet werden. Danach bleibt die Varianz nahezu konstant. Es wird erneut deutlich, dass die Nutzung A-priori-Wissens nur zu einer Effizienzsteigerung hinsichtlich der Exploration und weniger hinsichtlich der Exploitation führt, solange es sich nicht um eine sehr genaue Abbildung des realen Systems handelt.

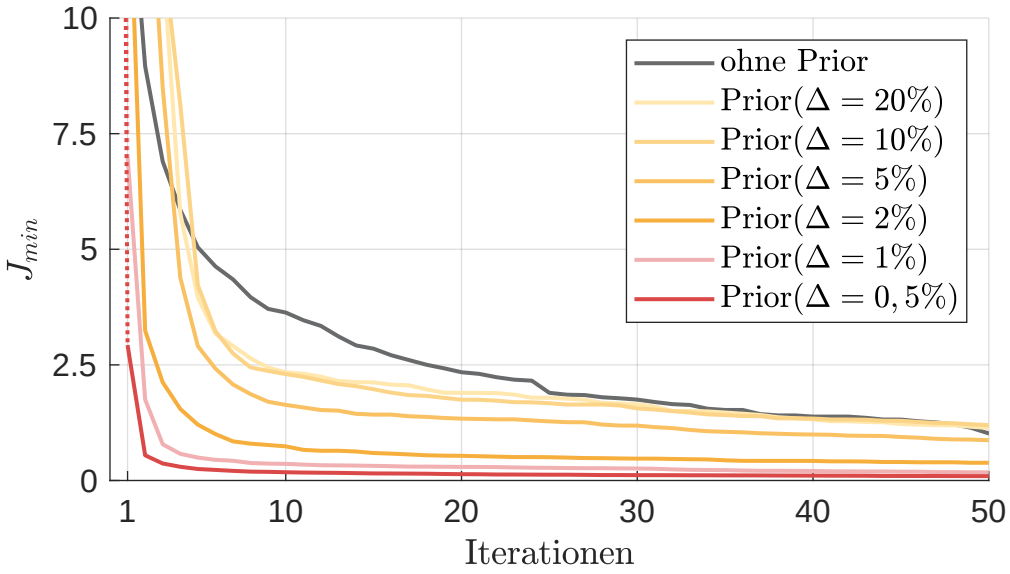


Bild 4-9: *Dargestellt ist eine Übersicht des Konvergenzverhaltens des Algorithmus für unterschiedliche Prior-Konfigurationen. Es ist ein deutlicher Zusammenhang zwischen Abweichung Δ und Performance zu erkennen.*

Erklären lässt sich der verbleibende benevolente Einfluss der Prior-Konfigurationen mit hohem Δ anhand Abbildung 4-9. Die vorzeitige Verschlechterung der Ergebnisse konnte bereits auf die Einbringung falscher Minima-Informationen zurückgeführt werden. Den Effekt der Überkompensation gilt es jedoch noch ferner zu betrachten. Erklären ließe sich dieser wie folgt. Da der Prior auch für hohe Δ eine grobe Verkörperung des realen Verlaufs bildet, können grundlegende Informationen über Wertebereich und Plateaus in den Posterior kommuniziert werden. Für No-Prior kommt es lokal zu einem Abfallen des GPs auf null und somit zu vermehrter Exploitation durch irrtümliche Annahme lokaler Minima. Es lässt sich also aussagen, dass ein Prior immer bestmöglich gewählt werden

sollte, es bei angemessener Verfehlung aber zu keiner langwährenden Fehlleitung des Algorithmus kommt. Es sollte daher für multidimensionale Optimierungsprobleme, sofern verfügbar, stets mindestens der grobe Zusammenhang von Eingangs- zu Ausgangsraum in Form eines Priors übergeben werden.

4.4.3 Effekt von Ausreißern

Bei der Verarbeitung von Daten treten für gewöhnlich ab und zu Ausreißer auf. Als *Ausreißer* werden zufällig gezogene Punkte einer Wahrscheinlichkeitsverteilung bezeichnet, die stark von ihrem Mittelwert abweichen. Da ein GP für jede Stelle $\boldsymbol{x} \in X$ eine Normalverteilung mit Mittelwert μ und Varianz σ^2 aufspannt, können auch derartige Ausreißer modelliert werden. Ob ein Ausreißer jedoch als solcher behandelt wird, hängt von den Hyperparametern ab, insbesondere der Noise Variance (siehe Seite 25). Wird σ_n^2 ausreichend groß gewählt, kann die Abweichung des Ausreißers zum entsprechenden Mittelwert als Rauschen abgebildet werden. In der Regel sind dafür jedoch zahlreiche Auswertungen in der direkten Nähe um den Ausreißer notwendig. Eine weitere Möglichkeit ist die Annahme einer hohen Noise Variance vorab in Form einer geeigneten Noise-Prior-Verteilung. Dabei sind jedoch in der Regel Kompromisse bezüglich der Abbildungsschärfe des GPs unvermeidbar, es seid denn, es erfolgt eine lokale Betrachtung des Rauschens mit Noise Variance $\sigma_n^2(\boldsymbol{x})$. Es empfiehlt sich daher, im Vorhinein die Daten einer Vorfilterung zu unterziehen, um eine Störung des GPs zu vermeiden.

5 Zusammenfassung und Ausblick

Die Bayessche Optimierung hat sich als mächtiges Werkzeug zur Optimierung komplexer Systeme erwiesen. Gerade für den Einsatz im Kontext von Black-Box-Optimierungen gibt es nur wenige Alternativen. Daher wird das Verfahren auch vermehrt zur Einrichtung von Prozessen genutzt, die nicht hinreichend beschrieben werden können. Durch BO kann das Ein-Ausgangsverhalten durch stichprobenartige Auswertung des realen Systems bestimmt werden. Obwohl gerade im Rahmen technischer Systeme oft gewisse Informationen über das Verhalten des Systems vorliegen, z.B. in Form eines physikalischen Dynamikmodells, werden diese im Hinblick auf die Optimierung meist nicht berücksichtigt.

Bei der Betrachtung eines eindimensionalen Optimierungsproblems konnte nachgewiesen werden, dass ein hinreichend genaues Dynamikmodell zu einer schnelleren Konvergenz des Algorithmus führt. Da es sich beim betrachteten Anwendungsbeispiel jedoch um ein besonders schlecht konditioniertes Optimierungsproblem handelt, führen gerade die Prior-Konfigurationen zu einer Verschlechterung des Konvergenzverhaltens, die die Extrema des realen Systems nur knapp verfehlen. Durch die resultierende Überdeckung der realen Minima kommt es zu einer verspäteten Auswertung der genannten Bereiche. Es zeigt sich, dass in diesen Fällen die Vorgabe eines groben Wertebereichs in Form des Priors zu deutlich besseren Ergebnissen führt. Die Effizienz des Verfahrens konnte im eindimensionalen Fall durch Einsatz eines Priors jedoch nur marginal erhöht werden.

Im dreidimensionalen Fall hingegen kommt dem Prior eine völlig andere Relevanz zu. So konnte durch Einsatz von Prior-Wissen mit deutlichen Abweichungen vom realen System (bis zu 20% Abweichung der Parameter) eine erhebliche Steigerung der Effizienz beobachtet werden. Die Effizienzsteigerung wirkt dabei vor allem in frühen Iterationen durch Vorgabe des groben Wertebereichs sowie in Form von Krümmungsinformationen. Da für technische Systeme eine Verfehlung der Parameterwerte um mehr als 20% als unwahrscheinlich anzunehmen ist und sich für hochdimensionale Systeme kein negativer Effekt des Priors gezeigt hat, sollte bei der Optimierung technischer Systeme mittels Bayesscher Optimierung in jedem Fall ein Prior eingesetzt werden. Dieser sollte so gut wie möglich gewählt werden, da sich ein proportionaler Zusammenhang zwischen Genauigkeit des Priors und Beschleunigung der Konvergenz gezeigt hat.

Des weiteren gilt es besonders bei der Optimierung technischer Systeme mit schlecht konditionierter Gütefunktion sowie unstetigem Verhalten die Abbildungsfähigkeit des GPs zu erhöhen (siehe Kapitel 5.1). Eine mögliche Ursache für die schlechte Konditionierung des Verlaufs einer Gütefunktion ist eine hohe Sensitivität des Systems hinsichtlich Änderung der Parameter. Für die Modellierung von Systemen mit besonders hohem Messrauschen und damit einhergehendem vermehrten Auftreten von Ausreißern empfiehlt sich der Einsatz von Student-t-Prozessen[33] oder Additiven Prozessen[34], die im Vergleich zu Gaußprozessen mit hoher Noise Variance eine weitestgehende Beschränkung von Kompromissen hinsichtlich der Modellierung der evaluierten Daten ermöglichen (siehe dazu [VRH+]). Darüber hinaus könnte im Rahmen der Optimierung des Systems eine Verifizierung der

[33] engl. student-t processes
[34] engl. additive processes

Prior-Informationen durchgeführt werden. So könnte bei nur geringer Übereinstimmung von Prior und realem System der Einfluss des Priors ggf. verkleinert werden. Eine derartige Implementierung des Priors wird in Kapitel 5.3 näher beschrieben. Auch hinsichtlich der Reduktion des mit BO verbundenen Rechenaufwands gibt es bekannte Ansätze wie Sparse Gaussian Process Regression (siehe dazu [Bij16]). Der Einsatz dieser Verfahren führt jedoch erst bei Verarbeitung größerer Datenmengen zu einer messbaren Steigerung der Performance.

Im Rahmen dieser Arbeit konnte nachgewiesen werden, dass es bei Berücksichtigung hinreichend genauen Prior-Wissens zu einer deutlichen Effizienzsteigerung der Bayesschen Optimierung kommt. Inwiefern es sich dabei um eine exakte Abbildung des realen Systems handeln muss, hängt von der Parametersensitivität des betrachteten Anwendungssystems sowie der Dimension des Optimierungsproblems ab. Allgemein empfiehlt sich selbst der Einsatz groben Vorwissens in Form eines Priors.

Im Laufe der Arbeit haben sich eine Reihe erfolgversprechender Ansätze zur weiterführenden Effizienzsteigerung der Einbindung des Prior-Wissens ergeben. Zuvor soll aber ein Konzept zur Steigerung der Abbildungsfähigkeit von GPs vorgestellt werden.

5.1 Stückweise Glättung der Kovarianz

Gaußprozesse folgen in der Regel einem glatten Verlauf. Das ist eine Eigenschaft der Kovarianzfunktion $\mathrm{k}(\boldsymbol{x}, \boldsymbol{x}')$. Soll jedoch unstetiges Prozessverhalten durch einen GP abgebildet werden, gereicht diese Eigenschaft zum Nachteil. In diesen Fällen ist es sinnvoll, die durch Unstetigkeiten voneinander abgegrenzten Bereiche des Eingangsraums unabhängig voneinander zu betrachten. Dazu kann eine stückweise glatte Kovarianzfunktion[35] $\mathrm{k}_{ps}(\boldsymbol{x}, \boldsymbol{x}')$ eingesetzt werden. Der Eingangsraum X wird dafür in Regionen $\Re_i$ unterteilt. Die Regionen werden in der Kovarianzfunktion voneinander entkoppelt durch (vgl. [Bij16])

$$\mathrm{k}_{ps}(\boldsymbol{x}, \boldsymbol{x}') := \begin{cases} \mathrm{k}(\boldsymbol{x}, \boldsymbol{x}'), & \boldsymbol{x} \in \Re_i \text{ und } \boldsymbol{x}' \in \Re_i \text{ für ein/mehrere } \Re_i \\ 0, & \text{sonst.} \end{cases}$$

Im eindimensionalen Fall können derartige Unstetigkeiten beispielsweise durch zweifache Differenzierung von $\mathrm{k}(\boldsymbol{x}, \boldsymbol{x}')$ für alle $\boldsymbol{x}, \boldsymbol{x}' \in \{\boldsymbol{x} \in X | \boldsymbol{x} \notin \mathfrak{X}\}$ identifiziert werden. Die Menge aller unstetigen Stellen $\mathfrak{X}$ muss dafür abzählbar sein. Durch Interpolation ergeben sich Peaks an den unstetigen Stellen. Im mehrdimensionalen Fall dagegen ist die Identifikation von Regionen $\Re_i$ nicht trivial. Hier können z.B. *Selbstorganisierende Karten*[36] genutzt werden, um Muster im Eingangsraum zu erkennen.

5.2 Adaptive Akquisitionsfunktionen

Im Rahmen der Untersuchungen hat sich gezeigt, dass die AQF MESmin0 in vielen Fällen gute Ergebnisse liefert. Jedoch gibt es noch Verbesserungspotential bezüglich der Optimierung parametersensitiver Systeme, wie dem starren Pendel. In Kapitel 4.4.1 wurde der

[35] engl. piecewise smooth covariance function
[36] engl. self-organizing maps

Effekt beschrieben, wenn Prior-Konfigurationen unzureichend genaue Abbildungen der realen Minima enthalten und so den GP „in die Irre führen“. Dieses Phänomen wird in dem Moment zum Problem, wenn dadurch die Auswertung von Stellen schlechterer Güte, jedoch mit deutlich höherer Standardabweichung, bevorzugt werden und somit nicht weiter in der Nähe des aktuellen GP-Minimums gesucht wird. Es bleibt zu zeigen, ob ein AQF-Ansatz mit Berücksichtigung des GP-Minimums y_{min} zu einer gewünschten Effizienzsteigerung führt. Ein möglicher Ansatz ergibt sich aus einer Erweiterung von MES-min0:

$$\alpha(\boldsymbol{x}) := \frac{1}{2}\left[\frac{\gamma_{y_*}(\boldsymbol{x})\cdot\Phi(\gamma_{y_*}(\boldsymbol{x}))}{2\Psi(\gamma_{y_*}(\boldsymbol{x}))} - \log\left(\Psi(\gamma_{y_*}(\boldsymbol{x}))\right) + \frac{\gamma_{y_{min}}(\boldsymbol{x})\cdot\Phi(\gamma_{y_{min}}(\boldsymbol{x}))}{2\Psi(\gamma_{y_{min}}(\boldsymbol{x}))} - \log\left(\Psi(\gamma_{y_{min}}(\boldsymbol{x}))\right)\right].$$

Dieser Ansatz kann als eine adaptive AQF gesehen werden, da y_{min} sich abhängig vom GP ändert. Die Existenz ungünstiger Nebeneffekte gilt es hierbei jedoch noch auszuschließen.

5.3 Gewichteter Prior

Als Resultat dieser Arbeit hat sich gezeigt, dass schlechte Prior-Informationen, besonders für parametersensitive Systeme, durch lokale Unzulänglichkeiten zu deutlich verzögerter Konvergenz führen können. Um derartige Unzulänglichkeiten geeignet identifizieren und ausbessern zu können, kann eine lokale Gewichtung des Priors vorgenommen werden. Die Gewichtung erfolgt durch die Funktion $\Xi(\boldsymbol{x}) \in (0, 1]$. Damit ergibt sich der Mittelwert der Posterior-Verteilung für Stellen $\boldsymbol{X}_*$ zu

$$\boldsymbol{\mu}^+(\boldsymbol{X}_*) = \Xi(\boldsymbol{X}_*)\cdot\boldsymbol{\mu}^-(\boldsymbol{X}_*) + \boldsymbol{K}_*^T\cdot(\boldsymbol{K}+\sigma_n^2\boldsymbol{E})^{-1}(\boldsymbol{f}(\boldsymbol{X}) - \Xi(\boldsymbol{X})\cdot\boldsymbol{\mu}^-(\boldsymbol{X})).$$

Die Wahl einer Funktion $\Xi(\boldsymbol{x})$ beinhaltet die geeignete Identifikation der Unzulänglichkeiten in Form von Abweichung zwischen Prior und realem System. Dabei ist die Fehlerfläche zwischen Prior-Funktion $\boldsymbol{\mu}^-(\boldsymbol{x})$ und Systemfunktion $\boldsymbol{f}(\boldsymbol{x})$ das geeignete Maß. Eine mögliche Prior-Gewichtungsfunktion ergibt sich somit zu

$$\Xi(\boldsymbol{x}) := \left(1 + \int_{\boldsymbol{x}\in X} \|\boldsymbol{f}(\boldsymbol{\xi}) - \boldsymbol{\mu}^-(\boldsymbol{\xi})\|\,\mathrm{d}\boldsymbol{\xi}\right)^{-1} \approx \left(1 + \sum_{\boldsymbol{x}_i\in X} \|\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{\mu}^-(\boldsymbol{x}_i)\|\right)^{-1}.$$

Die gegebene Funktion kann auch als Vertrauensfunktion interpretiert werden. Geht der Wert von Ξ an einer Stelle $\boldsymbol{x}$ gegen Null, wird dem Prior lokal nur wenig vertraut. Dies ist besonders für Stellen hoher Sensitivität der Fall.

5.4 Parameteridentifikation

Ein weiterer Ansatz zur Erhöhung der Effizienz des Priors ist das Konzept der Parameteridentifikation. Dabei erfolgt eine unterlagerte Optimierung der Parameter des Prior-Systems. Mittels Maximum Likelihood wird die wahrscheinlichste Parametrierung bestimmt. So können die besonders sensitiven Parameterabweichungen hinreichend ausgeglichen werden. Zur effizienten Maximierung der Likelihood bezüglich der Parameter ist

auch hier wieder der Einsatz eines gradientenbasierten Verfahrens sinnvoll. Der Gradient der Gütefunktion bezüglich der Prior-Parametrierung $\boldsymbol{p} := [l, m]^T$ kann, ähnlich in Kapitel 4.3, hergeleitet werden als

$$\begin{aligned}\nabla_{\boldsymbol{p}} J(\boldsymbol{\theta}, \boldsymbol{p}) &= \nabla_{\boldsymbol{p}} \left[\sum_{2} (\boldsymbol{x}_f - \boldsymbol{x}_{f,sim}(\boldsymbol{\theta}, \boldsymbol{p}))^2 \right] \\ &= -2(\boldsymbol{x}_f - \boldsymbol{x}_{f,sim}(\boldsymbol{\theta}, \boldsymbol{p}))^T \cdot \nabla_{\boldsymbol{p}} \boldsymbol{x}_{f,sim}(\boldsymbol{\theta}, \boldsymbol{p}).\end{aligned}$$

Der Gradient der Endlage ergibt sich ferner zu

$$\begin{aligned}\nabla_{\boldsymbol{p}} \boldsymbol{x}_{f,sim}(\boldsymbol{\theta}, \boldsymbol{p}) &= \frac{\mathrm{d}\boldsymbol{x}_N(\boldsymbol{x}_{N-1}, u_{N-1}, \boldsymbol{p})}{\mathrm{d}\boldsymbol{p}} \\ &= \frac{\partial \boldsymbol{x}_N}{\partial \boldsymbol{x}_{N-1}} \cdot \frac{\mathrm{d}\boldsymbol{x}_{N-1}}{\mathrm{d}\boldsymbol{p}} + \frac{\partial \boldsymbol{x}_N}{\partial u_{N-1}} \cdot \cancel{\frac{\mathrm{d}u_{N-1}}{\mathrm{d}\boldsymbol{p}}} + \frac{\partial \boldsymbol{x}_N}{\partial \boldsymbol{p}} \cdot \underbrace{\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}\boldsymbol{p}}}_{=\boldsymbol{E}_{2\times 2}} \\ &= \cdots \\ &= \sum_{k=1}^{N} \left(\prod_{m=k}^{N-1} \frac{\partial \boldsymbol{x}_{m+1}}{\partial \boldsymbol{x}_m} \right) \frac{\mathrm{d}\boldsymbol{x}_k}{\mathrm{d}\boldsymbol{p}}.\end{aligned}$$

Auf diese Weise ist es dem Algorithmus möglich, die Unstimmigkeit der Parameter zu erkennen und diese zu variieren. Die Anpassung erfolgt entweder zu Anfang jeder Iteration vor Bestimmung der Hyperparameter oder in definierten Iterationsintervallen.

Literaturverzeichnis

[ÅF96] Åström, K. J.; Furuta, K.: *Swinging up a pendulum by Energy Control*. Hrsg. von Lund Institute of Technology, Tokyo Institute of Technology. San Francisco, California, 1996

[Aue02] Auer, P.: Using confidence bounds for exploitation-exploration tradeoffs. *Journal of Machine Learning Research* (2002), Nr. 3, S. 397–422

[Bij16] Bijl, H.: *Gaussian process regression techniques: With applications to wind turbines: student version*. 2016. http://www.hildobijl.com/Research.php (besucht am 26. 11. 2019)

[Bis06] Bishop, C. M.: *Pattern recognition and machine learning*. Information science and statistics. New York: Springer, 2006

[Cul16] Culbertson, M. J.: *MCMC*. 25.10.2016. https://www.youtube.com/watch?v=kV_UfhjLlHI (besucht am 26. 11. 2019)

[DLvht] Deisenroth, M. P.; Luo, Y.; van der Wilk, M.: *A Practical Guide to Gaussian Processes*. nicht veröffentlicht. https://drafts.distill.pub/gp/ (besucht am 26. 11. 2019)

[FGB15] Fischer, L.; Gao, S.; Bernstein, A.: Machines Tuning Machines: Configuring Distributed Stream Processors with Bayesian Optimization. *Proceedings, 2015 IEEE International Conference on Cluster Computing*. Los Alamitos, California, 2015, S. 22–31

[Gab04] Gabašová, E.: *Covariance functions: Squared exponential kernel*. 2004. http://evelinag.com/Ariadne/covarianceFunctions.html (besucht am 26. 11. 2019)

[GSM+17] Golovin, D.; Solnik, B.; Moitra, S.; Kochanski, G.; Karro, J.; Sculley, D.: Google Vizier. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17*. Hrsg. von Matwin, S.; Yu, S.; Farooq, F. New York, New York, USA, 2017, S. 1487–1495

[Hea17] Head, T.: *Bayesian Optimization - Can you do better than randomly guessing parameters?* Hrsg. von Chaos Computer Club e.V. 2017. https://doi.org/10.5446/38177 (besucht am 26. 11. 2019)

[HHG14] Hern´andez-Lobato, J. M.; Hoffman, M. W.; Ghahramani, Z.: Predictive entropy search for efficient global optimization of black-box functions. *Advances in Neural Information Processing Systems (NIPS)* (2014)

[HS12] Hennig, P.; Schuler, C. J.: Entropy search for information-efficient global optimization. *Journal of Machine Learning Research* (2012), Nr. 13, S. 1809–1837

[Kat19] Kathan, M.: *Markov Chain Monte Carlo: Lifting your understanding of MCMC to an intermediate level*. 28.02.2019. https://towardsdatascience.com/markov-chain-monte-carlo-291d8a5975ae (besucht am 26. 11. 2019)

[Moc74] Moˇckus, J.: On Bayesian methods for seeking the extremum. *Optimization Techniques IFIP Technical Conference* (1974)

[Ras06] Rasmussen Carl Edward; Williams, C. K. I.: *Gaussian processes for machine learning*. Adaptive computation and machine learning. Cambridge, Mass.: MIT Press, 2006

[Six19] Six Sigma TC, Hrsg.: *Grundlagen: Die Normalverteilung*. 2019. https://www.6sigma-tc.de/de/six-sigma/grundlagen/six-sigma-normalverteilung/ (besucht am 26. 11. 2019)

[SKKS10] Srinivas, N.; Krause, A.; Kakade, S. M.; Seeger, M.: Gaussian process optimization in the bandit setting: no regret and experimental design. *International Conference on Machine Learning (ICML)* (2010)

[SLA12] Snoek, J.; Larochelle, H.; Adams, R. P.: *Practical Bayesian Optimization of Machine Learning Algorithms*. Lake Tahoe, Nevada, 2012

[SSW+16] Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R. P.; Freitas, N. de: Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE* 104 (2016), Nr. 1, S. 148–175

[Tec] Technische Universität Bergakademie Freiberg, Hrsg.: *Numerik gewöhnlicher Differentialgleichungen: 4 Runge-Kutta-Verfahren*. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&uact=8&ved=2ahUKEwi6_czhmP7lAhUN3qQKHW9oB3AQFjACegQIAhAC&url=http%3A%2F%2Fwww.mathe.tu-freiberg.de%2F~eiermann%2FVorlesungen%2FODE%2FFolien%2Fode_4_alt.pdf&usg=AOvVaw1lKpITfRlRnJ0WH8gf8AIq (besucht am 26. 11. 2019)

[Tim19] Timmermann, J.: *Optimale Steuerungen und Regelungen*. Paderborn, 2019

[TRL11] Titsias, M. K.; Rattray, M.; Lawrence, N. D.: Markov chain Monte Carlo algorithms for Gaussian processes. *Bayesian time series models*. Hrsg. von Cemgil, A. T.; Barber, D.; Chiappa, S. Cambridge: Cambridge University Press, 2011

[VRH+] Vanhatalo, J.; Riihimäki, J.; Hartikainen, J.; Jylänki, P.; Tolvanen, V.; Vehtari, A.: Bayesian Modeling with Gaussian Processes using the GPstuff Toolbox ()

[VRH+12] Vanhatalo, J.; Riihimäki, J.; Hartikainen, J.; Jylänki, P.; Tolvanen, V.; Vehtari, A.: *Bayesian Modeling with Gaussian Processes using the GPstuff Toolbox*. 2012

[Wan17] Wang, Z.: *Max-value Entropy Search for Efficient Bayesian Optimization*. 2017. https://github.com/zi-w/Max-value-Entropy-Search (besucht am 26. 11. 2019)

[WJ17] Wang, Z.; Jegelka, S.: *Max-value Entropy Search for Efficient Bayesian Optimization*. 2017

Anhang

Inhaltsverzeichnis

A1 Systemverhalten des starren Pendels **59**

A2 Code **63**
A2.1 Main 63
A2.2 Prozedur 64
A2.3 Bestimmung der Hyperparameter 66
A2.4 Akquisitionsfunktion 68
A2.5 Prior 69

A1 Systemverhalten des starren Pendels

Die in diesem Kapitel aufgeführten Abbildungen zeigen die Gütefunktion des starren Pendels, die das reale Systemverhalten darstellt. Dafür wurden stichprobenartig Ausschnitte aus dem Verlauf der Funktion gewählt und in zwei Darstellungsformen abgebildet. Jeweils links ist der Ausschnitt der Gütefunktion als dreidimensionale Oberfläche und rechts daneben als Heatmap dargestellt. Dabei wurde für die Heatmap zur Herausstellung markanter Linien eine logarithmische Skalierung der Güte gewählt. Die Bereiche gleicher Güte sind durch entsprechende Bereiche gleicher Färbung dargestellt.

Da für die Darstellung der Ausschnitte jeweils ein $\theta_i \in \{\theta_1, \theta_2, \theta_3\}$ einen konstanten Wert besitzt, ist der zugehörige Wert unter den Abbildungen angegeben. Die Einheit der θ_i entspricht Sekunden.

Es zeigt sich, dass sich besonders für die Wahl von θ_1 völlig verschiedene Funktionsverläufe für θ_2 und θ_3 ergeben. Auch bzgl. der Wahl von θ_2 besteht eine deutliche Sensitivität, die sich in Form verschiedenartiger Verläufe widerspiegelt. Hinsichtlich der Wahl von θ_3 ist lediglich eine unterschiedlich starke Ausprägung der lokalen Extrema zu verzeichnen. Die zugehörigen Stellen lokaler Extrempunkte verschieben sich in diesem Fall allerdings nicht.

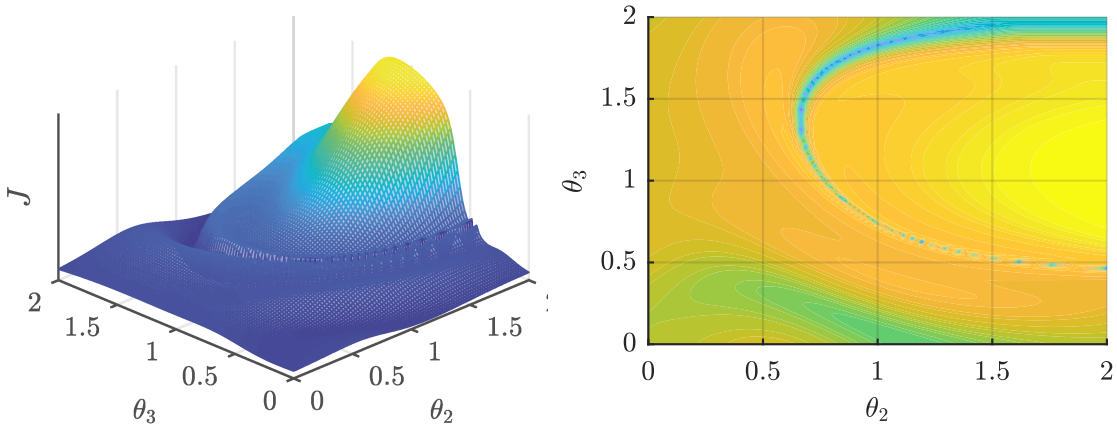


Bild A1-1: *Dargestellt ist die Gütefunktion des realen Systems für ein konstantes* $\theta_1 = 0\text{s}$ *(rechts logarithmische Skalierung der Güte).*

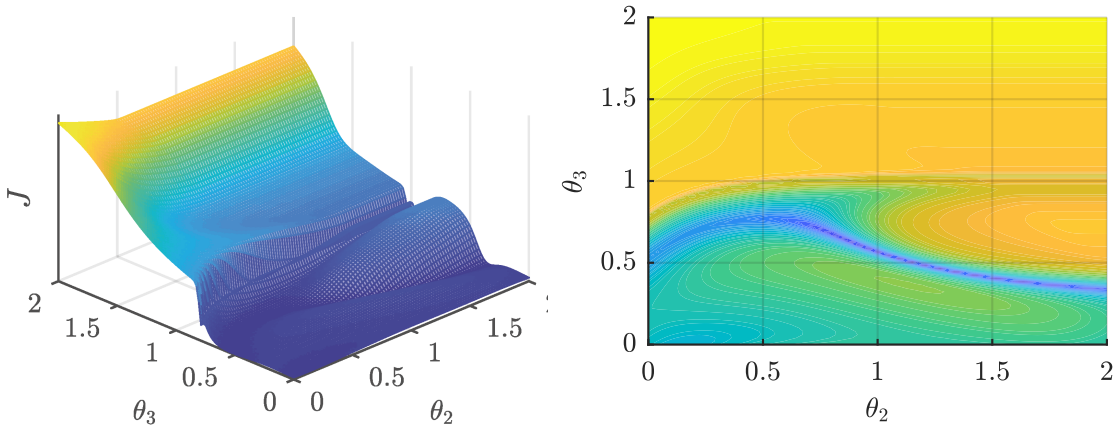


Bild A1-2: *Dargestellt ist die Gütefunktion des realen Systems für ein konstantes* $\theta_1 = 1\text{s}$ *(rechts logarithmische Skalierung der Güte).*

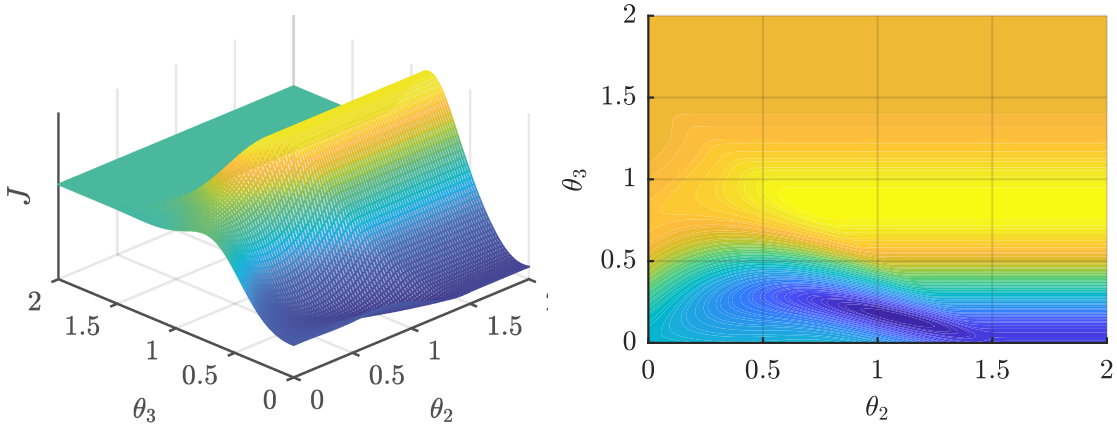


Bild A1-3: *Dargestellt ist die Gütefunktion des realen Systems für ein konstantes* $\theta_1 = 2\text{s}$ *(rechts logarithmische Skalierung der Güte).*

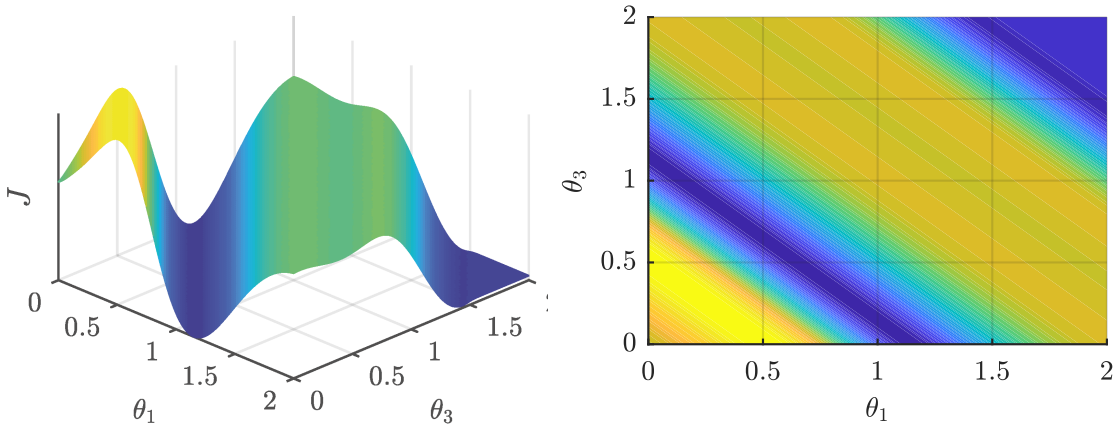


Bild A1-4: *Dargestellt ist die Gütefunktion des realen Systems für ein konstantes* $\theta_2 = 0\text{s}$ *(rechts logarithmische Skalierung der Güte).*

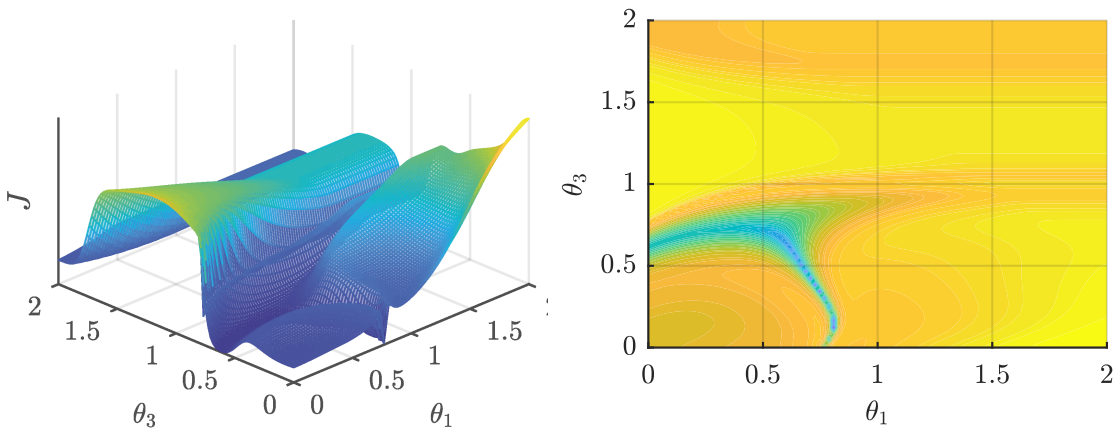


Bild A1-5: *Dargestellt ist die Gütefunktion des realen Systems für ein konstantes* $\theta_2 = 1\text{s}$ *(rechts logarithmische Skalierung der Güte).*

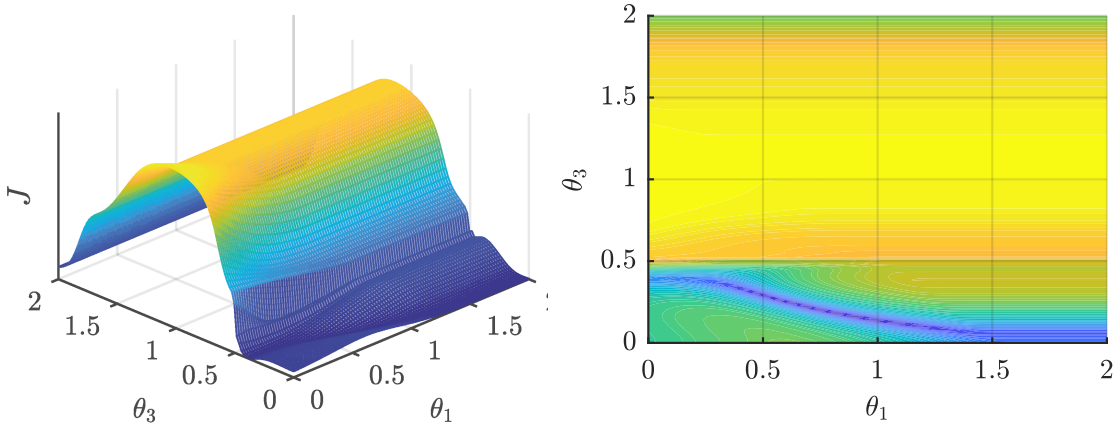


Bild A1-6: *Dargestellt ist die Gütefunktion des realen Systems für ein konstantes* $\theta_2 = 2\text{s}$ *(rechts logarithmische Skalierung der Güte).*

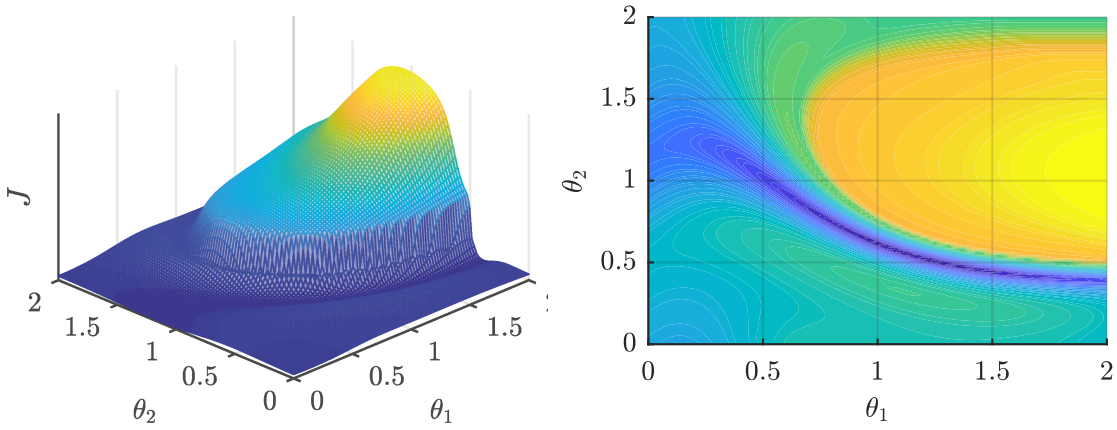


Bild A1-7: *Dargestellt ist die Gütefunktion des realen Systems für ein konstantes* $\theta_3 = 0\text{s}$ *(rechts logarithmische Skalierung der Güte).*

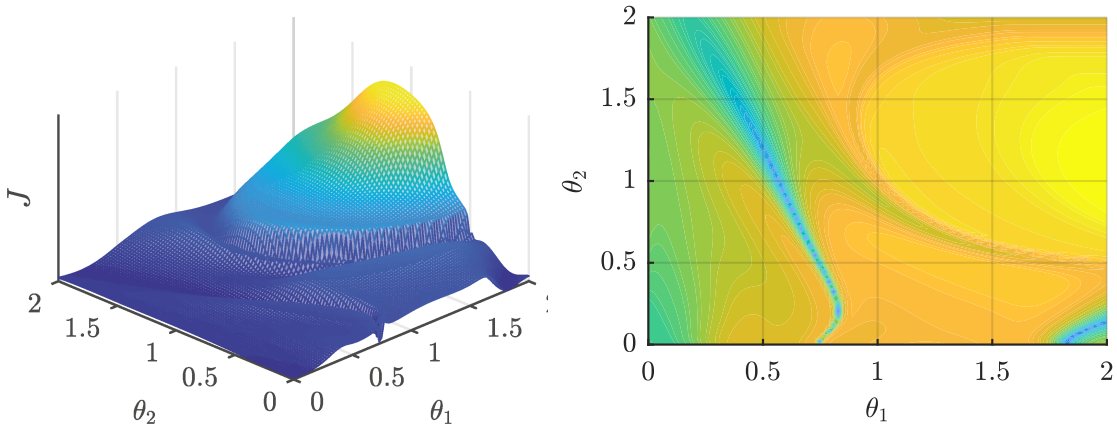


Bild A1-8: *Dargestellt ist die Gütefunktion des realen Systems für ein konstantes* $\theta_3 = 1\text{s}$ *(rechts logarithmische Skalierung der Güte).*

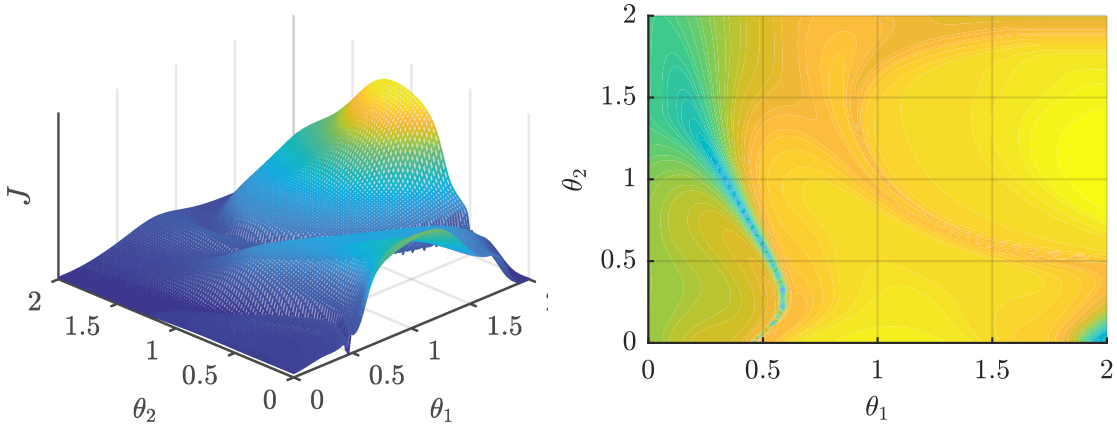


Bild A1-9: *Dargestellt ist die Gütefunktion des realen Systems für ein konstantes* $\theta_3 = 2\text{s}$ *(rechts logarithmische Skalierung der Güte).*

A2 Code

A2.1 Main

Zum Aufruf des Bayesschen Optimierers wird ‚`MAIN.m`' gestartet. Diese Datei enthält neben sämtlichen Parametern und Einstellungen die Schleife zur Durchführung von Batch-Berechnungen[37]. Die Ergebnisse werden gesammelt in ‚`batchBO.mat`' abgespeichert.

```
% Copyright (c) 2019 Oliver Schoen
clear; close all; clc; rng(254);

%General Options
param.T = [0,3.5];
param.umax = 5;
param.s = 50;
param.x0 = [-pi;0];
param.xf = [0;0];
param.nIter = 50; %number of iterations
param.zmin = [0 0 0];
param.zmax = [2 2 2];
param.nHypers = 10; %10
param.fixhyp = struct(); %presetting HPs possible
param.aqopt = optimoptions('fmincon','Algorithm','sqp','
   SpecifyObjectiveGradient',true,'Display','none','
   OptimalityTolerance',1e-12,'MaxIterations',100000,'
   MaxFunctionEvaluations',100000,'StepTolerance',1e-8);
param.prezz_k = []; %predefining points
param.preJJ_k = []; %predefining points

%MES-G
param.guesses = [];
param.nM = 1;
param.nK = 1;
param.maxes = 0; %define minimum value

%Real Model
param.realModel.g = 9.81;
param.realModel.l = 1.10;
param.realModel.m = 0.85;
param.realModel.opts = odeset('abstol', 1e-6, 'reltol', 1e
   -6, 'Stats', 'off');
param.realModel.noise = 0.02;

%Prior Simulation Model
```

[37] dt. stapelweise Durchführung aufeinanderfolgender Durchläufe

```
33 param.usePrior = true;
34 Delta = 2; %deviation of parameters in percentage
35 param.simModel.g = 9.81;
36 param.simModel.l = 1.10*(100+Delta)/100;
37 param.simModel.m = 0.85*(100+Delta)/100;
38 param.simModel.opts = odeset('abstol', 1e-6, 'reltol', 1e
      -6, 'Stats', 'off');
39 param.simModel.nDisc = 10000;
40
41 %Batch Mode
42 param.batch.nRuns = 50; %number of runs
43
44 %% CODE
45 param.z_0_archive = [];
46 Batch = cell(param.batch.nRuns, 1);
47 Seed = randi(1000, param.batch.nRuns, 1);
48 batchtime = tic;
49 for R = 1:param.batch.nRuns
50     param.Run = R;
51     Batch{R}.seed = Seed(R);
52     rng(Seed(R));
53
54     %% BAYESIAN OPTIMIZATION
55     fprintf(['\nRun ',num2str(R),'\n']);
56     [z_star, J_star, sim] = BO_Pend(param); %run BO
57     Batch{R}.z_star = z_star;
58     Batch{R}.J_star = J_star;
59     Batch{R}.sim = sim;
60 end
61 toc(batchtime)
62 fprintf(['\n Finished Batch in ', num2str(batchtime), 's,
      saving results.\n']);
63 save('batchBO.mat','Batch','param');
```

A2.2 Prozedur

Der Algorithmus zur Bayesschen Optimierung des dreidimensionalen Pendels ist durch ‚BO_Pend.m' gegeben. Der Startpunkt wird aus der Datei ‚z_0ref.mat' geladen und initialisiert. Danach wird iterativ das Minimum der Gütefunktion ‚objective.m' bestimmt. Dies beinhaltet jeweils die Optimierung der Hyperparameter, Bestimmung der Kernelmatrix, Auswertung der AQF und Simulation des realen Systems. Nach Durchlauf der Iterationen wird die beste evaluierte Parametrierung gewählt und simuliert. Die beste Parametrierung sowie die zugehörigen Trajektorien werden übergeben. Dabei wird Code aus der Toolbox zu Max-value Entropy Search for Efficient Bayesian Optimization [Wan17] und der GPstuff-Toolbox [VRH+12] verwendet.

```
% Copyright (c) 2019 Oliver Schoen
function [z_star, J_star, sim] = BO_Pend(param)
%% INITIALISATION
fprintf('BAYESIAN OPTIMIZATION\n');
zz_k = [param.prezz_k];
JJ_k = [param.preJJ_k];

%% CHOOSE STARTING POINT
load('z_0ref.mat','z_0'); %load random starting points
z_0 = z_0(param.Run,:);
disp(['Starting Point: [',num2str(z_0(1,1)), ',', num2str(
   z_0(1,2)), ',', num2str(z_0(1,3)), ']']);
[~,~,xfsim] = realModel(z_0, param); %simulate real model
xfsim = xfsim + param.realModel.noise*randn(1,1); %add
   noise
J_k = objective(param.xf, xfsim); %get value of objective
disp(['Value of Objective: ',num2str(J_k)]);
zz_k = [zz_k; z_0];
JJ_k = [JJ_k; J_k];

%% BAYESIAN OPIMIZATION
for I = 1:param.nIter
    fprintf('\nIteration %d\n',I);

    %% HYPERPARAMETER OPTIMIZATION
    [l, sigma, sigma_0] = sampleHypers(zz_k, JJ_k, param.
       nHypers, param.fixhyp);

    %% KERNEL MATRIX
    KInv = cell(param.nHypers, 1);
    for j = 1:param.nHypers
        K = computeKmmMatern52(zz_k, l(j,:)', sigma(j),
           sigma_0(j)); %get K
        KInv{j} = inv(K); %invert K
    end

    %% ACQUISITION FUNCTION OPTIMIZATION
    z_kplus1 = mesg_choosePRIORfixMaxMatern52(param.nM,
       param.nK, zz_k, JJ_k, KInv, param.guesses, sigma_0,
       sigma, l, param.zmin', param.zmax', param); %
       evaluate acquisition function
    disp(['Next to AQ-Point: [', num2str(z_kplus1(1,1)), ',
       ', num2str(z_kplus1(1,2)), ',', num2str(z_kplus1
       (1,3)), ']']);

    %% SIMULATION
```

```
38      [~,~,xfsim] = realModel(z_kplus1, param); %simulate
           real model
39      xfsim = xfsim + param.realModel.noise*randn(1,1); %add
           noise
40      J_kplus1 = objective(param.xf, xfsim); %get value of
           objective
41      disp(['Value of Objective: ',num2str(J_kplus1)]);
42
43      %% LOOPING
44      z_k = z_kplus1;
45      J_k = J_kplus1;
46      zz_k = [zz_k; z_k];
47      JJ_k = [JJ_k; J_k];
48  end
49  fprintf('End of Bayesian Optimization\n\n');
50
51  %% CHOOSE Z_STAR
52  [J_star, IndexStar] = min(JJ_k); %best value of objective
53  z_star = zz_k(IndexStar, :); %corresponding z_star
54  disp(['Best Point: [', num2str(z_star(1,1)), ',', num2str(
       z_star(1,2)), ',', num2str(z_star(1,3)), '], J = ',
       num2str(J_star)]);
55
56  %% SIMULATE BEST SYSTEM
57  [sim.t, sim.x_star, ~] = realModel(z_star, param); %
       simulate z_star
58  sim.u_star = ctrlfcnApprox(sim.t', z_star, param); %get
       control trajectory
59  end
```

A2.3 Bestimmung der Hyperparameter

Zur Bestimmung der Hyperparameter dient die Funktion ‚`sampleHypers.m`‘ von [HHG14]. Es wurden nur minimale Anpassungen vorgenommen. So wurde die Noise Variance deutlich reduziert und der Kernel gewechselt. Auf Basis der übergebenen Messdaten werden mehrere Hyperparametersätze gesampled und zurückgegeben.

```
1  % Copyright (c) 2014, J.M. Hernandez-Lobato, M.W. Hoffman,
       Z. Ghahramani
2  % This function is adapted from the code for the paper
3  % Hernandez-Lobato J. M., Hoffman M. W. and Ghahramani Z.
4  % Predictive Entropy Search for Efficient Global
       Optimization of Black-box
5  % Functions, In NIPS, 2014.
6  % https://bitbucket.org/jmh233/codepesnips2014
```

```
7  function [ l, sigma, sigma0 ] = sampleHypers(Xsamples,
       Ysamples, nSamples, fixhyp)
8  % Check if the hyper parameters are fixed
9  if isfield(fixhyp, 'l') && isfield(fixhyp, 'sigma') &&
       isfield(fixhyp, 'sigma0')
10     % Fix the hyper parameters
11     l = repmat(fixhyp.l, [nSamples, 1]);
12     sigma = ones(nSamples,1)*fixhyp.sigma;
13     sigma0 = ones(nSamples,1)*fixhyp.sigma0;
14 else
15     % We specify a gamma prior for the noise
16 %     pNoise = prior_logunif();
17     pNoise = prior_gamma('sh', 0.00000001, 'is', 1000);
18
19     % We specify the likelihood
20     lik = lik_gaussian('sigma2_prior', pNoise);
21
22     % We specify a gamma prior for the lengthscales
23     meanG = 0.5;
24     varianceG = 0.1;
25     alpha = meanG^2 / varianceG;
26     beta = meanG / varianceG;
27     pLengthscale = prior_gamma('sh', alpha, 'is', beta);
28
29     % We specify a gamma prior for the amplitud
30     pAmplitud = prior_logunif();
31
32     % We specify the covariance function
33 %     gpcf = gpcf_sexp('lengthScale', 0.3 * ones(1, size(
       Xsamples, 2)),
34 %     'lengthScale_prior', pLengthscale, 'magnSigma2_prior
       ', pAmplitud);
35     gpcf = gpcf_matern52('lengthScale', 0.3 * ones(1, size(
           Xsamples, 2)), 'lengthScale_prior', pLengthscale, '
           magnSigma2_prior', pAmplitud);
36
37     % We infer the hyper-parameters
38     gp = gp_set('lik', lik, 'cf', gpcf, 'jitterSigma2', 1e
           -10);
39
40     % We sample from the hyper-parameters
41     multiplier = 3;
42     burnin = 50;
43     [gp_rec, g, opt] = gp_mc(gp, Xsamples, Ysamples, '
           nsamples', (nSamples + burnin) * multiplier, '
           display', 0);
```

```
44      gp_rec = thin(gp_rec, multiplier * burnin + 1,
            multiplier);
45
46      % We get the samples
47      sigma0 = gp_rec.lik.sigma2;
48      l = 1 ./ gp_rec.cf{ 1 }.lengthScale.^2;
49      sigma = gp_rec.cf{ 1 }.magnSigma2;
50  end
```

A2.4 Akquisitionsfunktion

Als Akquisitionsfunktion dient MESmin0, Adaption der MES von [WJ17]. Neben der Einbringung von Prior-Informationen wurde der Kernel gewechselt und die aufwendige Bestimmung der Maximaverteilung sowie gradientenbasierten Optimierung entfernt. In ‚mesg_choosePRIORfixMaxMatern52.m‘ wurden Funktionen aus der Toolbox zu Max-value Entropy Search for Efficient Bayesian Optimization [Wan17] und der GPstuff-Toolbox [VRH+12] verwendet. Es wird zunächst ein Gitter zufällig verteilter Punkte gesampled und mithilfe von Mittelwert, Varianz und Prior werden daraufhin die zugehörigen AQF-Werte bestimmt. Zuletzt erfolgt die Evaluierung des optimalen Punktes.

```
1  % Copyright (c) 2019 Oliver Schoen
2  % Copyright (c) 2017 Zi Wang
3  function optimum = mesg_choosePRIORfixMaxMatern52(nM, nK,
       xx, yy, KernelMatrixInv, guesses, sigma0, sigma, l, xmin
       , xmax, param)
4      yvals = 0;
5
6      % Sample a set of random points in the search space.
7      gridSize = 1000;
8      d = size(xmin, 1);
9      Xgrid = repmat(xmin', gridSize, 1) + repmat((xmax -
           xmin)', gridSize, 1) .* rand(gridSize, d);
10     Xgrid = [ Xgrid ; guesses; xx ];
11     sx = size(Xgrid, 1);
12     maxes = ones(nM, nK)*param.maxes;
13
14     %Prior
15     Pre = prior(xx,param);
16     pre = prior(Xgrid,param);
17
18     for i = 1:nM
19         % Compute the posterior mean/variance predictions
               for Xgrid.
20         [meanVector, varVector] = mean_varMatern52(Xgrid,
               xx, yy-Pre, KernelMatrixInv{i}, l(i,:), sigma(i)
               , sigma0(i),param);
```

```
21
22          % Avoid numerical errors by enforcing variance to
               be positive.
23          varVector(varVector<=sigma0(i)+eps) = sigma0(i)+eps
               ;
24
25          % Obtain the posterior standard deviation.
26          sigVector = sqrt(varVector);
27
28          %Prior
29          meanVector = meanVector + pre;
30
31          %Negate for maximization problem
32          meanVector = -meanVector;
33          yy = -yy;
34          maxes = -maxes;
35
36          % Compute Gamma
37          gamma = (repmat(maxes(i,:),[sx 1]) - repmat(
               meanVector, [1, nK])) ./ repmat(sigVector, [1,
               nK]);
38          yvals = yvals + gamma;
39      end
40  yvals = yvals / nM / nK;
41
42  %Pick Minimum
43  [~, minIdx] = min(yvals);
44  optimum = Xgrid(minIdx,:);
45  end
```

A2.5 Prior

Das Prior-Wissen wird durch Simulation des Systems mit Prior-Parametrierung verkörpert. Dazu wird bei Aufruf von ‚`prior.m`‘ der Prior-Wert, sowie optional der Gradient des Priors, ausgewertet. Zur Bestimmung des Gradienten werden Finite Gradienten genutzt.

```
1  % Copyright (c) 2019 Oliver Schoen
2  function [F,GR] = prior(z_k, param)
3  %Parameters
4  xf = param.xf;
5  g = param.simModel.g;
6  l = param.simModel.l;
7  m = param.simModel.m;
8  param.opts = param.simModel.opts;
9
```

```
%% PRIOR MEAN
%Prior Model System Functions
odefcn = @(x, u) [x(2, :); g*sin(x(1, :))/l+u./(m*l^2)];
ctrlfcn = @(t,z_k,umax) ctrlfcnApprox(t, z_k, param);

for jj = 1: size(z_k,1)
    if param.usePrior
        %Simulation
        [~,~,xfsim] = simulation(odefcn,ctrlfcn,z_k(jj,:),
           param);
        %Objective
        f = objective(xf,xfsim);
        if isnan(f)
            keyboard;
        end
        F(jj,:) = f; %normal Prior
%         F(jj,:) = 20*z_k(jj,end)^2+25; %use other Prior-
   function
    else
        F(jj,:) = 0; %for no-Prior
    end

    if nargout>1
        %% GRADIENT OF PRIOR MEAN
        if param.usePrior
            %Finite Gradients
            epsilon = 0.0001;
            gr = zeros(size(z_k(jj,:),2),1);
            for i = 1:(size(z_k(jj,:),2))
                z_kUP = z_k(jj,:);
                z_kDOWN = z_k(jj,:);

                z_kUP(i) = z_k(jj,i)+epsilon;
                [~,~,xfsimUP] = simulation(odefcn,ctrlfcn,
                   z_kUP,param);
                fUP = objective(xf,xfsimUP);
                z_kDOWN(i) = z_k(jj,i)-epsilon;
                [~,~,xfsimDOWN] = simulation(odefcn,ctrlfcn
                   ,z_kDOWN,param);
                fDOWN = objective(xf,xfsimDOWN);

                gr(i,1) = (fUP-fDOWN)/(2*epsilon);
            end
            GR(:,jj) = gr;  %with Prior
        else
          GR(:,jj) = [0,0,0]'; %for non-Prior
        end
```

```
53      end
54  end
55  end
```