Tatjana Berger BSc

Alterungsmodellierung von Lithium-Ionen-Zellen mit Gauß-Prozess-Regression

MASTERARBEIT

zur Erlangung des akademischen Grades
Diplom-Ingenieurin

Masterstudium Technische Mathematik: Operations Research und Statistik

eingereicht an der

Technischen Universität Graz

Betreuer:
Univ.-Prof.i.R. Dipl.-Ing. Dr. techn. Ernst Stadlober
Institut für Statistik

Dipl.-Ing. Matthias Scharrer
VIRTUAL VEHICLE Research Center (ViF)

Graz, April 2018

DANKSAGUNG

Diese Arbeit entstand am VIRTUAL VEHICLE Research Center in Graz, Österreich. Ich bedanke mich für die Förderung im Rahmen des COMET K2 - Competence Centers for Excellent Technologies Programms des Österreichischen Bundesministeriums für Verkehr, Innovation und Technologie (bmvit), des Österreichischen Bundesministeriums für Digitalisierung und Wirtschaftsstandort (bmdw), der Österreichischen Forschungsförderungsgesellschaft mbH (FFG), des Landes Steiermark sowie der Steirischen Wirtschaftsförderung (SFG).

Ebenfalls danke ich den unterstützenden Industriepartnern AVL List GmbH, Porsche AG und Volkswagen AG.

Nun möchte ich mich bei denjenigen bedanken, die durch ihre fachliche und persönliche Unterstützung zum Gelingen dieser Masterarbeit beigetragen haben.

Besonderer Dank gilt meinen Betreuern Univ.-Prof.i.R. Dipl.-Ing. Dr.techn. Ernst Stadlober vom Institut für Statistik an der Technischen Universität Graz und an Dipl.-Ing. Matthias Scharrer von VIRTUAL VEHICLE Research Center für die Unterstützung und die hilfreichen Anregungen, sowie für die konstruktive Kritik bei der Erstellung dieser Arbeit. Ich bedanke mich außerdem dafür, dass sie jederzeit geduldig und hilfsbereit meine Fragen beantwortet haben. Des Weiteren bedanke ich mich bei der Batteriefachgruppe des VIRTUAL VEHICLE Research Center, allen voran beim Gruppenleiter Dipl.-Ing. Dr. Alexander Thaler, für die liebevolle Aufnahme in der Gruppe während der Entstehung dieser Arbeit.

Mein größter Dank gilt meinen Eltern, die mir das Studium ermöglicht und mich in all meinen Lebensentscheidungen immer wieder unterstützt haben.

Herzlich bedanken möchte ich mich auch bei meinen Freunden und Studienkollegen, die mich auf diesem Weg begleitet haben. Allen voran möchte ich mich bei Fatima Jammoul, Florian Mussner und Emanuel Pichlbauer bedanken, die mich während meiner Studienzeit immer wieder ermutigt und mir nicht nur in mathematischen Angelegenheiten geholfen haben. Danke, dass ich euch als Freunde habe.

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz,am ....

(Unterschrift)

ABSTRACT

The main focus of this thesis is the aging modeling of lithium ion cells with a Gaussian process. This thesis is realized in cooperation with the VIRTUAL VEHICLE Research Center (ViF).

The first chapter gives a short overview of the structure and applications of lithium ion cells. Furthermore an overview of the battery aging experiment of the VIRTUAL VEHICLE Research Center (ViF) is given in this chapter. In the second chapter the basic knowledge, needed for the theory of the Gaussian process regression is given. Beside the basic ideas of the Gaussian process regression, the selection of plausible covariance functions and mean functions are given in chapter three. Methods to determine plausible hyperparameters, i.e. parameters of the covariance function, are explained in this chapter as well. Due to the fact that most of this methods need optimization procedures for nonlinear functions (e.g. Differential Evolution), an overview over such procedures is also given in chapter three. In the last chapter an implementation of the Gaussian process regression is done. Therefore the End-of-Life of lithium ion cells are simulated. To do so, we used the *ALICe 1* data, which was created during the aging experiment. This data consist of 41 load points, which are distinguished in the different settings of 7 factors: $T, CC, ADC, PDC, F, SoC, dSoC$. Beside the implementation of the Gaussian process regression, we also compare the results of the common linear regression models to the results of the Gaussian process regression.

KURZFASSUNG

In Zusammenarbeit mit dem VIRTUAL VEHICLE Research Center (ViF) wird im Rahmen dieser Arbeit die Alterung von Lithium-Ionen-Zellen mit Hilfe einer Gauß-Prozess-Regression modelliert.

In Kapitel 1 wird ein kurzer Überblick über den Aufbau und die Anwendungen von Lithium-Ionen-Zellen gegeben. Weiters wird das Alterungsexperiment vorgestellt aus dem die Daten erhoben wurden, die im Rahmen dieser Arbeit untersucht werden. In Kapitel 2 wird das Grundwissen, das für die Theorie der Gauß-Prozess-Regression benötigt wird, vorgestellt. In Kapitel 3 wird zusammen mit den Grundideen der Gauß-Prozess-Regression geklärt, wie wir eine geeignete Kovarianz-Funktion, sowie Erwartungswert-Funktion für ein gegebenes Problem auswählen. Weiters werden in diesem Kapitel Methoden zur Bestimmung der Hyperparameter, das sind Parameter der Kovarianz-Funktion, vorgestellt. Da die meisten Methoden zur Hyperparameterbestimmung Optimierungsverfahren für nichtlineare Probleme (z.B. Differential Evolution) benötigen, enthält Kapitel 3 einen Überblick über geeignete Verfahren. Kapitel 4 beschäftigt sich mit einer praktischen Anwendung der Gauß-Prozess-Regression. Dabei werden die Lebensdauern von Lithium-Ionen-Zellen simuliert. Als Daten werden die im Rahmen des Alterungsexperimentes erstellten *ALICe 1* Daten verwendet. Diese bestehen aus 41 Lastpunkten, die sich durch unterschiedliche Ausprägungen von 7 Faktoren: $T, CC, ADC, PDC, F, SoC, dSOC$ unterscheiden. Neben der praktischen Anwendung der Gauß-Prozess-Regression wird ein Vergleich der Ergebnisse zwischen einer gewöhnlichen linearen Regression und einer Gauß-Prozess-Regression angegeben.

Inhaltsverzeichnis

1 Einleitung **1**
1 Lithium-Ionen-Zellen . . . 2
1.1 Vor- und Nachteile von Lithium-Ionen-Zellen . . . 4
1.2 Anwendungen . . . 4
2 Alterungsexperiment . . . 5

2 Grundlagen **9**
1 Maschinelles Lernen . . . 9
2 Gauß-Prozess . . . 10
2.1 Gaußverteilung . . . 10
2.2 Bayes-Statistik . . . 11
2.3 Gauß-Prozess . . . 13

3 Gauß-Prozess-Regression **17**
1 Grundlagen der Gauß-Prozess-Regression . . . 17
2 Multivariate Gauß-Prozess-Regression . . . 22
2.1 Additiver Ansatz . . . 23
2.2 Abstandsnorm Ansatz . . . 25
3 Plausible Wahl der Kovarianz-Funktion . . . 26
3.1 Stationäre Kovarianz-Funktionen . . . 26
3.2 Kombinationen von Kovarianz-Funktionen . . . 29
3.3 Modellselektion . . . 29
4 Plausible Wahl der Hyperparameter . . . 30
4.1 Testen von verschiedenen Werten . . . 30
4.2 Optimierungsverfahren . . . 32
4.3 Maximierung der marginalen log-Likelihood-Funktion . . . 44
4.4 Minimierung Gesamtabstand mit Hilfe der L_2-Norm . . . 46
4.5 Minimierung Gesamtabstand mit Hilfe der L_1-Norm . . . 49
5 Plausible Wahl der Erwartungswert-Funktion . . . 50
5.1 Simple Kriging . . . 50
5.2 Universal Kriging . . . 52

4 Praktische Anwendung **55**
1 Wahl der Test- und Trainingsdaten . . . 55
1.1 ALICe 1 Daten . . . 55
1.2 Erstellung Test- und Trainingsdaten . . . 58

2 Plausible Wahl der Kovarianz-Funktion 66
3 Plausible Wahl der Hyperparameter 69
4 Plausible Wahl der Erwartungswert-Funktion 76
5 Vergleich lineare Regression und Gauß-Prozess-Regression 79
6 Zusammenfassung 81

A Line-Search-Methode **85**

Tabellenverzeichnis

3.1 Beispiele für stationärer Kovarianz-Funktionen. 27
3.2 Optimierung Rosenbrock Funktion mit Startwerten $(-1.2, 1)$. 42
3.3 Optimierung Rosenbrock Funktion mit Startwerten $(0.1, 0.1)$. 42
3.4 Optimierung Rosenbrock Funktion mit Startwerten $(2, 2)$. 42
3.5 Maximierung der log-Likelihood-Funktion mit Startwerten $(1, 0.880, 0.180)$. 45
3.6 Maximierung der log-Likelihood-Funktion mit Startwerten $(10, 10, 10)$. . . . 45
3.7 Minimierung Gesamtabstand mit L_2-Norm mit Startwerten $(1, 0.880, 0.180)$. 48
3.8 Minimierung Gesamtabstand mit L_2-Norm mit Startwerten $(10, 10, 10)$. . . 48
3.9 Minimierung Gesamtabstand mit L_1-Norm mit Startwerten $(1, 0.880, 0.180)$. 51
3.10 Minimierung Gesamtabstand mit L_1-Norm mit Startwerten $(10, 10, 10)$. . . 51

4.1 Ausgangsversuchsplan für ALICe 1 Experiment, $n = 41$. 56
4.2 Auflistung der Faktoren und ihre Einstellungen. 57
4.3 Auflistung minimaler und maximaler Werte der Faktoren. 58
4.4 Skalierter Versuchsplan ALICe 1, $n = 41$. 59
4.5 ALICe 1 Datensatz Teil 1, $n = 3 \times 14 = 42$. 60
4.6 ALICe 1 Datensatz Teil 2, $n = 3 \times 14 = 42$. 61
4.7 ALICe 1 Datensatz Teil 3, $n = 3 \times 13 = 39$. 62
4.8 Gemittelte ALICe 1 Daten, $n = 41$. 64
4.9 Testdaten 1 65
4.10 Testdaten 2 65
4.11 Testdaten 3 65
4.12 Optimale Testdaten mit quadratischer Exponentialfunktion. 68
4.13 Optimale Testdaten mit rational quadratischer Exponentialfunktion. 68
4.14 Optimale Testdaten mit Gamma-Exponentialfunktion. 69
4.15 Optimale Testdaten mit Matern Funktion mit $\nu = p + \frac{1}{2}$. 69
4.16 Zufällige Testdaten mit quadratischer Exponentialfunktion. 70
4.17 Zufällige Testdaten mit rational quadratischer Exponentialfunktion. 71
4.18 Zufällige Testdaten mit Gamma-Exponentialfunktion. 71
4.19 Zufällige Testdaten mit Matern Funktion mit $\nu = p + \frac{1}{2}$. 71
4.20 Vergleich Gesamtabstände für verschiedene Funktionen und Testdaten. . . 72
4.21 Gauß-Prozess-Regression mit getesteten Hyperparameter. 73
4.22 Vergleich Gauß-Prozess-Regression mit unterschiedlichen Hyperparameter. . 73
4.23 Hyperparameterbestimmung mit verschiedenen Optimierungsverfahren. . . 73

4.24 Vergleich Kovarianz-Funktionen mit optimierten Hyperparametern. 75
4.25 Gauß-Prozess-Regression mit Hyperparameterbestimmung durch DE. 76
4.26 Vergleich Gesamtabstände für verschiedene Erwartungswert-Funktionen. . 77
4.27 Gauß-Prozess-Regression mit lm_6 als Erwartungswert-Funktion. 78
4.28 Gauß-Prozess-Regression und multiple lineare Regression. 80
4.29 Berechnungsdauer: Gauß-Prozess-Regression vs. lineare Regression. 83

Abbildungsverzeichnis

1.1 Alterung mit linearer Regression [Cifrain, 2015]. 2
1.2 Entladung einer Zelle [Brain, 2006]. 3
1.3 Ladung einer Zelle [Brain, 2006]. 4
1.4 Visualisierung Entladungs- und Ladungszyklen [M.Cifrain, 2013]. 6

3.1 Drei Realisierungen eines zufälligen Gauß-Prozesses. 19
3.2 Bedingter Gauß-Prozess mit rauschfreien A-priori Beobachtungen. 21
3.3 Bedingter Gauß-Prozess mit A-priori Beobachtungen die ein Rauschen beinhalten. 23
3.4 Quadratische Exponentialfunktion. 27
3.5 Weitere Beispiele für stationäre Kovarianz-Funktionen. 28
3.6 Kombination mit Hilfe von Addition und Multiplikation. 28
3.7 Rosenbrock-Funktion mit globalen Minimum bei $(1, 1)$. 41
3.8 Rosenbrock-Funktion mit den eingezeichneten lokalen Minima, die wir durch das CG-Verfahren erhalten. 43
3.9 Maximierung der log-Likelihood-Funktion mit Startwerten $(1, 0.880, 0.180)$. 45
3.10 Maximierung der log-Likelihood-Funktion mit Startwerten $(10, 10, 10)$. . . . 45
3.11 Minimierung Gesamtabstand mit L_2-Norm mit Startwerten $(1, 0.880, 0.180)$. 48
3.12 Minimierung Gesamtabstand mit L_2-Norm mit Startwerten $(10, 10, 10)$. . . 48
3.13 Minimierung Gesamtabstand mit L_1-Norm mit Startwerten $(1, 0.880, 0.180)$. 51
3.14 Minimierung Gesamtabstand mit L_1-Norm mit Startwerten $(10, 10, 10)$. . . 51

4.1 Visualisierung Ergebnisse mit zufälligen Testdaten. 63
4.2 Visualisierung Lage der berechneten Testdaten. 67
4.3 Vergleich Kovarianz-Funktionen ohne Optimierung der Hyperparameter. . . 70
4.4 Vergleich Kovarianz-Funktionen ohne Optimierung der Hyperparameter. . . 72
4.5 Hyperparameterbestimmung mit verschiedenen Optimierungsverfahren. . . 74
4.6 Vergleich Kovarianz-Funktionen mit optimierten Hyperparametern. 75
4.7 Bestimmung Erwartungswert-Funktion. 78
4.8 Vergleich Gauß-Prozess-Regression und lineare Regression. 80

Kapitel 1

Einleitung

Fossile Brennstoffe gehören heutzutage immer noch zu den wichtigsten Formen der Energiequellen. Da diese Brennstoffe nicht nur für uns und unsere Umwelt negative Einflüsse haben, sondern auch ihre Vorräte immer knapper werden, ist es wichtig alternative Energiequellen zu entwickeln und weiter auszubauen.

Vor allem in der Automobilindustrie erhalten deshalb elektrisch betriebene Fahrzeuge immer mehr Aufmerksamkeit, und man ist interessiert diese Art von Fahrzeugen weiterzuentwickeln.

Als Energiequelle dienen dabei Batterien. Vor allem Lithium-Ionen-Batterien wird dabei besondere Beachtung geschenkt, da sie einige Vorteile gegenüber anderen wiederaufladbaren Batterien haben. Diese Vorteile werden im Abschnitt 1 aufgelistet.

In dieser Arbeit interessieren wir uns für die Alterung der Lithium-Ionen-Zellen, aus denen eine Lithium-Ionen-Batterie besteht. Diese Alterung ist von zahlreichen Faktoren abhängig. Des Weiteren existieren Interaktionen zwischen den Faktoren, aber auch mit anderen Faktoren, die bisher nicht beobachtet werden können. Wegen dieser Komplexität der Alterung wurden im Rahmen des *K2 Projektes E3T3 (ALICe)* [Cifrain, 2015] des *VIRTUAL VEHICLE Research Center (ViF)* Alterungsexperimente durchgeführt. Die so entstandenen Daten wurden im Rahmen der Arbeit von *G. Gößler* [Gößler, 2015] statistisch mit Hilfe von linearen Regressionen ausgewertet und modelliert, um so die Alterung von Lithium-Ionen-Zellen zu untersuchen. Diese Arbeit lieferte schon eine gute Möglichkeit die Alterung von Lithium-Ionen-Zellen zu modellieren.

In Abb. 1.1 wird das Ergebnis der Schätzung der Lebensdauer, die mit Hilfe einer linearen Regression von *G. Gößler* [Gößler, 2015] erstellt wurde, grafisch dargestellt. Das Ergebnis ist dabei sortiert nach den beobachteten Lebensdauern in Jahren. Die blaue Linie gibt die gemessenen Lebensdauern der jeweiligen Zellen an. Die geschätzten Lebensdauern werden mit Hilfe der roten Linie dargestellt. Neben den 6 Lastpunkten (C series), die während des Alterungsexperimentes nur gelagert wurden - ohne geladen oder entladen zu werden - und den 35 Lastpunkten (L series), die entladen und geladen wurden, werden bei dieser linearen Regression auch 3 Validierungspunkte (V series) betrachtet. Diese Validierungspunkte werden in unserer Arbeit allerdings vernachlässigt.

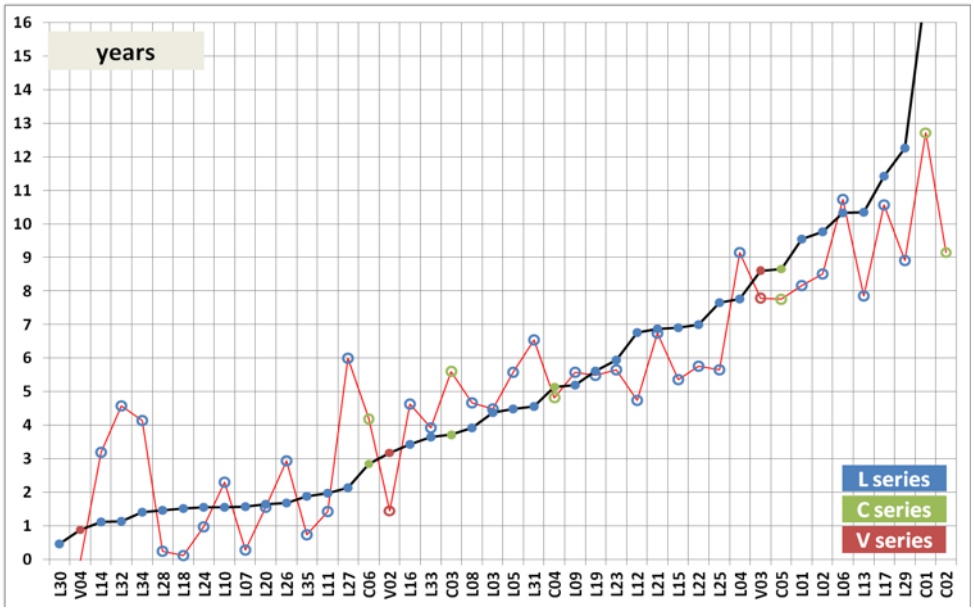

Abbildung 1.1: Alterung mit linearer Regression [Cifrain, 2015].

Wie wir nun anhand Abb. 1.1 erkennen, gibt es allerdings immer noch Verbesserungspotenzial, da die geschätzten Lebensdauern oft von den gemessenen Lebensdauern abweichen. Diese Verbesserung erhoffen wir zu erhalten, indem wir die *ALICe 1* Daten, die wir durch das Alterungsexperiment im Rahmen des *K2 Projektes E3T3 (ALICe)* des *VIRTUAL VEHICLE Research Center (ViF)* erhalten haben, mit Hilfe einer Gauß-Prozess-Regression modellieren. Deshalb beschäftigt sich diese Arbeit im Rahmen des *K2 Projektes E3T6 (ALICe 2)* des *VIRTUAL VEHICLE Research Center (ViF)* mit der Theorie (Kapitel 2 und Kapitel 3) und der anschließenden Auswertung der Daten (Kapitel 4) mit Hilfe der Gauß-Prozess-Regression.

1 Lithium-Ionen-Zellen

In diesem Abschnitt wollen wir einen kurzen Überblick über die Funktionsweise und Vor- und Nachteile von Lithium-Ionen-Zellen geben. Die folgende Zusammenfassung basiert vor allem auf der Arbeit von *G. L. Plett* [Plett, 2015].

Funktionsweise von Lithium-Ionen-Zellen
Die wesentlichen Komponenten einer Zelle sind:

- negative Elektrode
- positive Elektrode
- Elektrolyt
- Separator

Während der Entladung gibt die negative Elektrode Elektronen ab. Deshalb wird die negative Elektrode auch als Anode bezeichnet. Diese freigegebenen Elektronen fließen über einen äußeren Kreis zur positiven Elektrode. Die positive Elektrode wird deshalb auch als Kathode bezeichnet, da sie Elektronen dazugewinnt. Um den Kreis zu schließen wandern während der Entladung positiv geladene Ionen von der negativen Elektrode zur positiven Elektrode. Dieser Entladungsvorgang wird grafisch in Abb. 1.2 dargestellt.

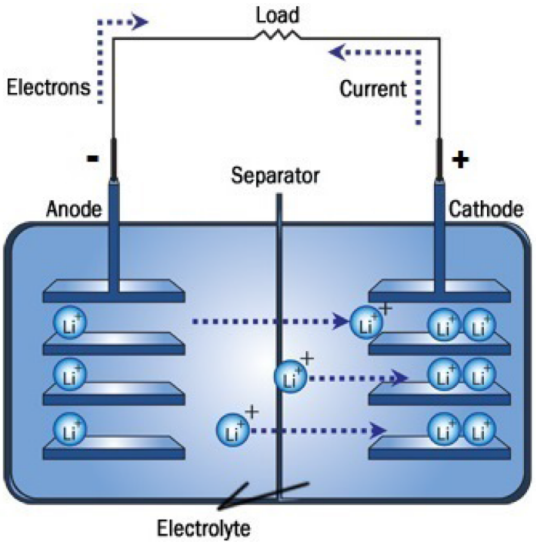

Abbildung 1.2: Entladung einer Zelle [Brain, 2006].

Während der Ladung geschieht genau das Umgekehrte. Die positive Elektrode gibt Elektronen ab und wird dadurch zur Anode. Diese freigegebenen Elektronen fließen durch den äußeren Kreis zur negativen Elektrode. Dadurch bekommt die negative Elektrode Elektronen dazu und wird zur Kathode. Weiters wandern nun die positiv geladenen Ionen von der positiven Elektrode durch den Elektrolyt zur negativen Elektrode. Die Ladung ist in Abb. 1.3 dargestellt.

Der Strom fließt während der Ladung und Entladung immer in die andere Richtung der Elektronen. Das bedeutet, dass bei der Entladung der Strom von der positiven Elektrode in Richtung der negativen Elektrode fließt und umgekehrt.

Der Separator trennt die positiven und negativen Elektroden voneinander. Er ist ein ionischer Leiter, da die positiv geladenen Ionen von einer Elektrode über den Elektrolyt durch den Separator zur anderen Elektrode wandern können. Allerdings ist der Separator ein elektronischer Isolator, da die Elektronen nicht durch den Separator hindurchkommen. Der Separator schützt also vor einem internen Kurzschluss. Dadurch könnte sich die Zelle selbst entladen und daher nutzlos werden.

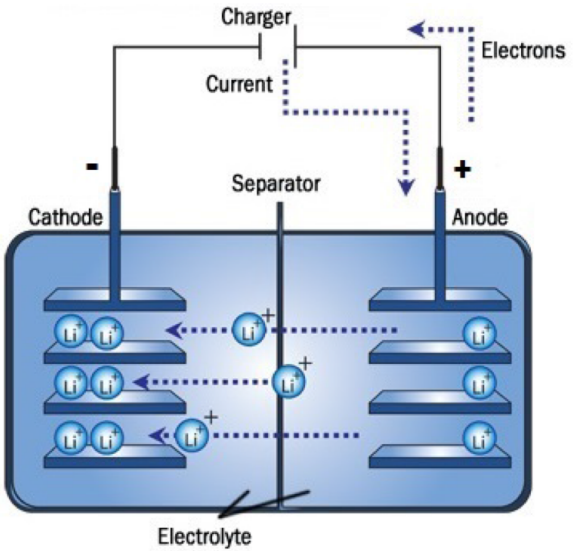

Abbildung 1.3: Ladung einer Zelle [Brain, 2006].

1.1 Vor- und Nachteile von Lithium-Ionen-Zellen

Die Funktionsweise, die wir im vorherigen Abschnitt beschrieben haben, gilt auch für andere Typen von wiederaufladbaren Zellen wie z.B. Bleisäure- oder Nickel-Metallhydrid-Akkumulatoren. Allerdings haben Lithium-Ionen-Zellen einige Vorteile gegenüber diesen anderen Zellen.

- Lithium-Ionen-Zellen funktionieren mit höherer Spannung als andere wiederaufladbare Zellen. Typischerweise liegt die Spannung bei Lithium-Ionen-Zellen bei $3.7V$. Die Spannung anderer Zellen liegt nur zwischen $1.2V$ und $2.1V$.
- Die höhere Spannung der Lithium-Ionen-Zelle bedeutet, dass eine einzelne Zelle bei gleichem Strom wesentlich mehr Leistung abgeben kann als andere Zelltypen.
- Weiters haben Lithium-Ionen-Zellen eine niedrige Selbstentladung.

Doch obwohl die Lithium-Ionen-Zellen sehr viele Vorteile haben, besitzen sie auch Nachteile:

- Lithium-Ionen-Zellen sind teurer als Zellen mit vergleichbarer Kapazität.
- Des Weiteren sind Lithium-Ionen-Zellen anfälliger für Überlastungen.

1.2 Anwendungen

Doch trotz dieser Nachteile werden Lithium-Ionen-Zellen für eine große Anzahl an Anwendungen genutzt [Warner, 2015]. Vor allem im Automobilbereich spielen Lithium-Ionen-Batterien eine wichtige Rolle. Dabei wird die Nutzung der Batterien in vier Kategorien eingeteilt:

- Micro Hybride (uHEV)

- Hybrid Elektrofahrzeuge (HEV)
- Plug-In Hybrid Elektrofahrezeuge (PHEV)
- Batterie-Elektrofahrzeuge (BEV)

Genauere Details zu den einzelnen Kategorien findet man im Buch [Warner, 2015].

Aber nicht nur im Automobilbereich haben sich Lithium-Ionen-Batterien etabliert, sondern auch für tragbare Geräte, die wir alle nutzen - wie z.B. Laptops oder Smartphones - werden Lithium-Ionen-Batterien verwendet.

Für das Alterungsexperiment des *K2 Projektes E3T3 (ALICe)* [Cifrain, 2015] des *VIRTUAL VEHICLE Research Center (ViF)* wurden zwei verschiedene Zelltypen betrachtet. Zum einen wurden kleine Zellen des Typs 18650 betrachtet, die z.B. für Laptops, aber auch als Batteriezellen in BEVs verwendet werden. Zum anderen werden große Zellen des PHEV2-Typs benutzt. Diese finden in der Automobilindustrie bei PHEVs ihre Verwendung.

2 Alterungsexperiment

In diesem Abschnitt beschreiben wir kurz die Vorgehensweise des Alterungsexperimentes des *K2 Projektes E3T3 (ALICe)* und welche Faktoren bei diesem Experiment berücksichtigt worden sind. Der folgende Abschnitt basiert vor allem auf der Arbeit von *G.Gößler* [Gößler, 2015] und dem Abschlussbericht des *K2 Projekts E3T3 (ALICe)* [Cifrain, 2015].

Die Grundidee des Experiments besteht darin, die Zellen unter Einfluss verschiedener, vorher festgelegter Faktoren so lange zu laden und entladen, bis die Zelle nicht mehr einsatzfähig ist. Eine Zelle hat dann ihr Lebensende (End-of-Life, EoL) erreicht, falls einer der folgenden Punkte eintritt.

- Die Kapazität der Zelle fällt unter 70% der ursprünglichen Kapazität.
- Der Innenwiderstand liegt 300% über dem ursprünglichen Innenwiderstand.

Um zu überprüfen, ob die Zelle noch funktionsfähig ist, wird nach einigen Wochen des Ladens und Entladens, die Zelle in einem sogenannten *Referenz-Test-Procedure (RTP)*-Schritt getestet. Ist die Zelle noch einsatzfähig, wird sie wieder geladen und entladen, bis zur nächsten Überprüfung. Dies sollte solange geschehen, bis die Zelle ihr Lebensende erreicht hat. Da das Experiment aber nicht so lange gelaufen ist, bis alle Zellen einsatzunfähig geworden sind, wurde eine Extrapolation der Lebensdauer durchgeführt. Genauere Details werden in Kapitel 4, Abschnitt 1 beschrieben.

Wie zuvor erwähnt, werden die Zellen während des Experiments immer wieder geladen und entladen. Eine vollständige Ladung und Entladung der Zelle wird Zyklus genannt. Dieser Zyklus wird für jede Zelle unter Einfluss von bestimmten Faktoren durchgeführt. In Abb. 1.4 werden solche Zyklen grafisch dargestellt.

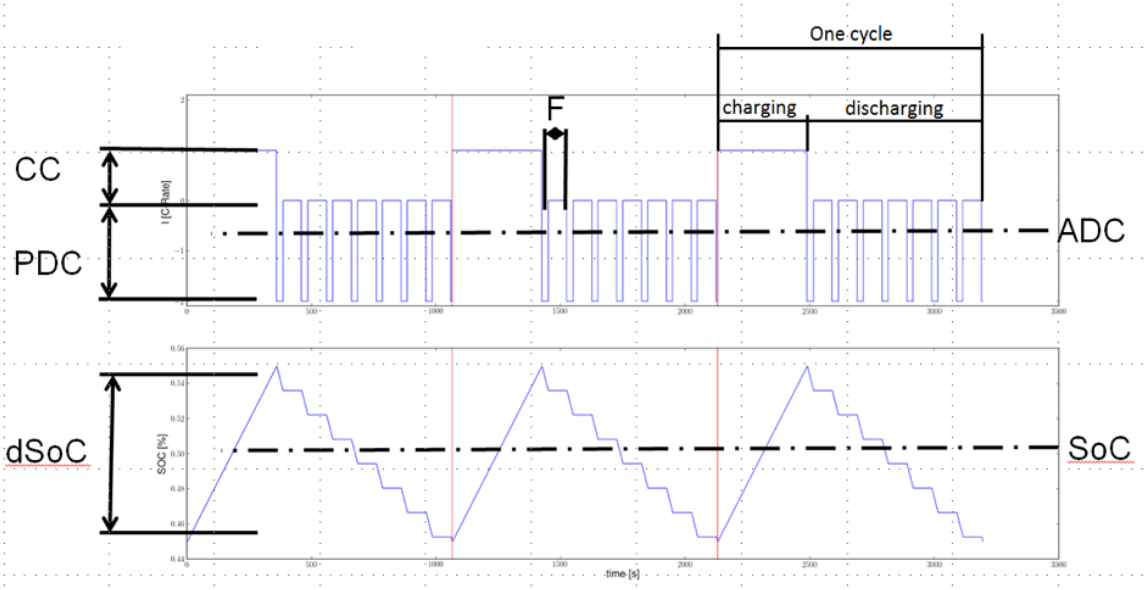

Abbildung 1.4: Visualisierung Entladungs- und Ladungszyklen [M.Cifrain, 2013].

Im Alterungsprojekt *K2 E3T3 (ALICe)* [Cifrain, 2015] wurden folgende Faktoren festgelegt:

- **Temperatur (T)**
  Da die Außentemperatur bei der die Zelle entladen und geladen wird einen Einfluss auf die Lebensdauer der Zelle hat, werden die Zyklen unter verschiedenen Temperaturen durchgeführt.

- **Ladestrom (*Charge Current*, CC)**
  Jede Zelle wird mit Gleichstrom geladen. Der Faktor *CC* gibt dabei an mit welcher Stromstärke die Zelle geladen wird.

- **Durchschnittlicher Ladestrom (*Average discharge current*, ADC)**
  Das Entladen der Zelle geschieht mit Hilfe von Stromimpulsen. Diese Impulse sind von mehreren Faktoren abhängig. Einer davon ist der durchschnittliche Entladestrom (*ADC*). Wie der Name schon sagt, gibt dieser Faktor die durchschnittliche Stromstärke an mit der die Zelle entladen wird.

- **Entladestromspitze (*Peak discharge current*, PDC)**
  Auch der *PDC*-Faktor hat einen Einfluss auf den Stromimpuls der zum Entladen der Zelle verwendet wird. Er gibt die maximale Stromstärke an, die bei einem Entladungsimpuls verwendet wird.

- **Frequenz (F)**
  Die Frequenz gibt an, wie lange ein jeweiliger Entladungsimpuls dauert.

- **Ladezustand (*State of Charge*, SoC)**
  Der *SoC*-Faktor gibt die aktuelle Kapazität der Zelle an. Dabei wird dieser als arithmetisches Mittel des minimalen und maximalen *SoC*-Wertes während eines Zykluses

angesehen:

$$SoC = \frac{min_{SoC} + max_{SoC}}{2}.$$

- **Delta-Ladezustand (*delta State of Charge*, dSoC)**
  Der $dSoC$-Faktor gibt die Differenz zwischen minimalem und maximalem SoC- Wert während eines Zyklus an.

$$dSoC = max_{SoC} - min_{SoC}.$$

Natürlich gibt es noch andere Faktoren, die einen Einfluss auf die Alterung von Zellen haben, wie z.B. Unterschiede im Zellaufbau, Messfehler, Messabweichungen, etc. Diese Faktoren können allerdings nicht einfach kontrolliert werden. Deshalb wurden diese Faktoren nicht im Experiment berücksichtigt. Welche Einstellungen die gewählten Faktoren besitzen, wurde in der Arbeit von *G.Pregartner* [Pregartner, 2012] festgelegt. Genauere Details zu den Einstellungen der Faktoren findet man in Kapitel 4, Abschnitt 1.

Kapitel 2

Grundlagen

In diesem Abschnitt wollen wir einige wichtige Grundlagen erklären, die wir für die spätere Betrachtung der Gauß-Prozess-Regression in Kapitel 3 benötigen.

Wir wollen zuerst einen Einblick in den Bereich des maschinellen Lernens (Abschnitt 1) geben. Des Weiteren führen wir den Gauß-Prozess (Abschnitt 2) ein und klären die dafür benötigten Grundlagen der multivariaten Gaußverteilung und Bayes-Statistik. Außerdem geben wir einen Überblick über die Vor- und Nachteile des Gauß-Prozesses an.

1 Maschinelles Lernen

Das Wort „Lernen“ ist für uns ein selbstverständlicher Begrff. Wir benutzen dieses Wort um die Fähigkeit von Menschen und Tieren zu beschreiben sich Wissen und verschiedene Fertigkeiten anzueignen und diese auf neue Situationen anzuwenden. Doch nicht nur Menschen und Tiere können lernen, sondern auch Computerprogramme und Maschinen.

T. M. Mitchell [Mitchell, 1997] gibt folgende Definition für das maschinelle Lernen eines Computerprogramms an:

Definition 2.1 (**Maschinelles Lernen**)**.**
Ein Computerprogramm lernt von einer Erfahrung E unter Berücksichtigung von verschiedenen Aufgaben T und einem Leistungsmaß P, wenn es eine Aufgabe T mit Hilfe der Erfahrung E gemessen am Leistungsmaß P besser erfüllt als zuvor.

Dies bedeutet nichts anderes, als dass beim maschinellen Lernen Algorithmen eingesetzt werden, welche aus bekannten Daten, den sogenannten Trainingsdaten, lernen und diese gewonnenen Informationen nutzen, um Aussagen über neue Daten, den Testdaten, zu treffen. Somit steht hier der Begriff Lernen für den Gewinn von Informationen aus bekannten Daten.

Wenn bestimmte Trainingsdaten gegeben sind, kann somit maschinelles Lernen genutzt werden um Schätzer für Werte einer unbekannten Funktion f an neuen Datenpunkten zu erhalten. Dabei kann zwischen zwei Arten von maschinellem Lernen unterschieden werden. [Nilsson, 1998]:

- **Überwachtes Lernen (*supervised learning*)**
  Hierbei geht man davon aus, dass die Werte der unbekannten Funktion f für alle Trainingspunkte bekannt sind. Mit diesem Wissen kann eine Funktion h gefunden werden, die ähnliche Werte für die Trainingsdaten liefert wie die unbekannte Funktion f. Funktion h wird dann genutzt um die Werte der neuen Datenpunkte zu schätzen.

- **Unüberwachtes Lernen (*unsupervised learning*)**
  In diesem Fall sind die Werte der unbekannten Funktion f für die Trainingspunkte nicht bekannt. Bei dieser Art von Lernen will man die Daten auf Gemeinsamkeiten untersuchen und sie in verschiedene Kategorien anordnen.

Der Gauß-Prozess (Abschnitt 2) bzw. die Gauß-Prozess-Regression (Kapitel 3), mit der wir uns in dieser Arbeit genauer beschäftigen, zählt zum überwachten Lernen, da in unserem Fall die Werte der Trainingsdaten bekannt sind. Ebenso gehört die lineare Regression zu den Verfahren des überwachten Lernens. Auf diese Methode wird allerdings in dieser Arbeit nicht mehr genauer eingegangen, da die Auswertungen der *ALICe 1* Daten (Kapitel 4, Abschnitt 1) schon in der Masterarbeit von *G. Gößler* [Gößler, 2015] erfolgten.

2 Gauß-Prozess

Der Gauß-Prozess gehört, wie schon zuvor erwähnt, zum überwachten maschinellen Lernen. Da dieser Prozess vor allem auf der multivariaten Gauß- oder Normalverteilung und den Grundlagen der Bayes-Statistik beruht, wollen wir diese Grundlagen einführen, bevor wir uns mit der eigentlichen Definition des Gauß-Prozesses beschäftigen.

2.1 Gaußverteilung

Ist ein Zufallsvektor $\boldsymbol{X}$ Gaußverteilt, bedeutet das nichts anderes, als dass er einer multivariaten Normalverteilung entspricht. Dazu betrachten wir nun folgende Definitionen [Vogel, 2013]:

Definition 2.2 (**univariate Normalverteilung**)**.**
Sei X eine univariat normalverteilte Zufallsvariable:

$$X \sim N(\mu, \sigma^2).$$

Dann besitzt X folgende Dichte:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),\ x \in \mathbb{R},\ \mu \in \mathbb{R},\ \sigma^2 > 0.$$

Der Parameter μ entspricht dem Erwartungswert und σ^2 der Varianz von X.

Definition 2.3 (**multivariate Normalverteilung**)**.**
Im Gegensatz dazu ist ein p-dimensionaler Zufallsvektor $\boldsymbol{X}$ genau dann p-variat normalverteilt oder auch Gaußverteilt

$$\boldsymbol{X} \sim N(\boldsymbol{\mu}, \Sigma),$$

falls er folgende Dichte besitzt:

$$f(\boldsymbol{x}) = f(x_1, \ldots, x_p) = \frac{1}{\sqrt{(2\pi)^p det(\Sigma)}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right),\ \boldsymbol{x} \in \mathbb{R}.$$

Hier bezeichnet der Parameter $\boldsymbol{\mu} \in \mathbb{R}^p$ den Erwartungswertvektor und $\Sigma \in \mathbb{R}^p \times \mathbb{R}^p$ die Varianz-Kovarianzmatrix von $\boldsymbol{X}$.

Definition 2.4 (Varianz-Kovarianzmatrix).
Die Varianz-Kovarianzmatrix $\Sigma_{p\times p}$ ist folgendermaßen definiert:

$$\Sigma = Cov(\boldsymbol{X}) = E((\boldsymbol{X}-\boldsymbol{\mu})(\boldsymbol{X}-\boldsymbol{\mu})^T) = E(\boldsymbol{X}\boldsymbol{X}^T) - \boldsymbol{\mu}\boldsymbol{\mu}^T =$$
$$= \begin{pmatrix} Var(X_1) & Cov(X_1, X_2) & \cdots & Cov(X_1, X_p) \\ Cov(X_2, X_1) & Var(X_2) & \cdots & Cov(X_2, X_p) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(X_p, X_1) & Cov(X_p, X_2) & \cdots & Var(X_p) \end{pmatrix}.$$

Multivariat normalverteilte Zufallsvektoren besitzen nun folgende wichtige Eigenschaften [Vogel, 2013]:

- Ein p-variat normalverteilter Zufallsvektor besteht aus p univariat normalverteilten Zufallsvariablen.
- Seien $\boldsymbol{X_1}$ und $\boldsymbol{X_2}$ k- bzw. $(d-k)$- verteilte Zufallsvektoren mit:

  $$\begin{pmatrix} \boldsymbol{X_1} \\ \boldsymbol{X_2} \end{pmatrix} \sim N\left(\begin{pmatrix} \boldsymbol{\mu_1} \\ \boldsymbol{\mu_2} \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}\right).$$

  Dann gilt :

  $$\boldsymbol{X_2} \sim N(\boldsymbol{\mu_2}, \Sigma_{22}),\ \boldsymbol{\mu_2} \in \mathbb{R}^{d-k},\ \Sigma_{22} \in \mathbb{R}^{d-k} \times \mathbb{R}^{d-k},$$

  und für die bedingte Verteilung von $\boldsymbol{X_2}$ gegeben $\boldsymbol{X_1} = \boldsymbol{x_1}$ gilt:

  $$\boldsymbol{X_2}|(\boldsymbol{X_1} = \boldsymbol{x_1}) \sim N\left(\boldsymbol{\mu_2} + \Sigma_{21}\Sigma_{11}^{-1}(\boldsymbol{x_1} - \boldsymbol{\mu}_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}\right). \tag{2.1}$$

2.2 Bayes-Statistik

Die Bayes-Statistik [Hoff, 2009] kann für vielfältige Aufgaben verwendet werden. z.B:

- Parameterschätzungen
- Beschreibung von beobachteten Daten
- Vorhersage von fehlenden Daten
- Vorhersage zukünftiger Daten

- Bildung von Strukturen für Modellschätzungen, Modellauswahl und Modellvalidierungen.

Die Grundlagen der Bayes-Statistik liegen in der Wahrscheinlichkeitstheorie und dem Satz von Bayes [Bronstein et al., 2001a]:

Satz 2.1 (**Satz von Bayes**).
Seien A und B zwei Ereignisse mit A-priori Wahrscheinlichkeiten $P(A)$ und $P(B) > 0$. Dann kann die bedingte Wahrscheinlichkeit, dass das Ereignis A eintritt, gegeben, dass das Ereignis B bereits geschehen ist, folgendermaßen berechnet werden:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

Da in der Bayes-Statistik allerdings hauptsächlich mit Dichten gearbeitet wird, gibt es eine analoge Formulierung des Satzes von Bayes für Dichten [Böcker, 2004]:

Satz 2.2 (**Satz von Bayes für Dichten**).
Seien $\boldsymbol{X}$ und $\boldsymbol{Y}$ zwei stetige Zufallsvektoren mit gemeinsamer Dichte $f(\boldsymbol{x}, \boldsymbol{y})$. Weiters seien

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \int_{\mathbb{R}} f(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\boldsymbol{y},$$
$$f_{\boldsymbol{Y}}(\boldsymbol{y}) = \int_{\mathbb{R}} f(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\boldsymbol{x}$$

die Randdichten von $\boldsymbol{X}$ und $\boldsymbol{Y}$. Dann kann die bedingte Dichte

$$f(\boldsymbol{y}|\boldsymbol{x}) = \frac{f(\boldsymbol{x}, \boldsymbol{y})}{f_{\boldsymbol{X}}(\boldsymbol{x})}$$

folgendermaßen berechnet werden:

$$f(\boldsymbol{y}|\boldsymbol{x}) = \frac{f(\boldsymbol{x}|\boldsymbol{y}) f_{\boldsymbol{Y}}(\boldsymbol{y})}{\int_{\mathbb{R}} f(\boldsymbol{x}|\boldsymbol{y}) f_{\boldsymbol{Y}}(\boldsymbol{y}) \mathrm{d}\boldsymbol{y}} = \frac{f(\boldsymbol{x}|\boldsymbol{y}) f_{\boldsymbol{Y}}(\boldsymbol{y})}{f_{\boldsymbol{X}}(\boldsymbol{x})}. \tag{2.2}$$

In der Bayes-Statistik wird für den Begriff der Dichte $f(\cdot)$ auch häufig der Begriff Verteilung verwendet. Es wird also nicht wie in anderen Bereichen üblich zwischen Dichten und Verteilungen unterschieden. Weiters wird in der Bayes-Statistik die Notation $p(\cdot)$ für Dichten bzw. Verteilungen verwendet.

Für die weitere Betrachtung der Bayes-Statistik führen wir nun folgende Parameter ein:

- Sei $\boldsymbol{Y}$ der Stichprobenraum, der die Menge aller möglichen beobachtbaren Datensätze aus der Population beinhaltet. Dann bezeichnet man mit $\boldsymbol{y} \in \boldsymbol{Y}$ den Datensatz der die numerischen Werte der beobachteten Teilmenge einer Population beschreibt.

- Weiters sei $\boldsymbol{\Theta}$ der Parameterraum, der die Menge aller möglichen Parameterwerte beinhaltet, die die wahre Populationscharakteristik beschreiben könnte. Dann bezeichnen wir mit $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ den numerischen Wert der unbekannten Populationscharakteristik.

Die Werte von $\boldsymbol{y}$ und $\boldsymbol{\theta}$ werden vor der Durchführung des Experimentes als unbekannt angenommen. Erst nach dem Experiment ist $\boldsymbol{y}$ bekannt. Diese Information über $\boldsymbol{y}$ kann in die Schätzung von $\boldsymbol{\theta}$ miteinfließen.

Ziel der Bayes-Statistik ist es, die unbekannte Charakteristik einer Population $\boldsymbol{\theta}$ mit Hilfe einer beobachteten Teilmenge $\boldsymbol{y}$ dieser Population zu beschreiben.

Dazu kennen wir für jeden numerischen Wert $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ die A-priori Verteilung $p(\boldsymbol{\theta})$. Diese Verteilung beschreibt die Wahrscheinlichkeit unter der Annahme, dass $\boldsymbol{\theta}$ der wahre Wert unserer Populationscharakteristik ist, bevor $\boldsymbol{y} \in \boldsymbol{Y}$ beobachtet wurde. Weiters kennen wir für jedes $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ und $\boldsymbol{y} \in \boldsymbol{Y}$ ein Strichprobenmodell $p(\boldsymbol{y}|\boldsymbol{\theta})$. Dieses Stichprobenmodell beschreibt das Verhalten einer Stichprobe $\boldsymbol{y}$ unter dem Parameter $\boldsymbol{\theta}$. Aus dem Integral des Produkts dieser zwei Verteilungen ergibt sich die sogenannte marginale Likelihood der beobachteten Daten $\boldsymbol{y} \in \boldsymbol{Y}$:

$$p(\boldsymbol{y}) = \int_{\boldsymbol{\Theta}} p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}.$$

Nachdem die Werte von $\boldsymbol{y}$ beobachtet wurden, können wir die Verteilung von $\boldsymbol{\theta}$, gegeben die beobachteten Daten $\boldsymbol{y}$, mit Hilfe des Satz von Bayes (2.2) berechnen:

$$p(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{y})} \propto p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta}).$$

Wichtig zu erwähnen ist, dass der Satz von Bayes nur beschreibt wie sich die Verteilung des Parameters $\boldsymbol{\theta}$ verändert, wenn wir die Information von den beobachtbaren Daten $\boldsymbol{y}$ hinzufügen. Der Satz sagt nichts über die Berechnung der A-priori Verteilung von $\boldsymbol{\theta}$ aus, sondern nur wie man den A-posteriori Wert $p(\boldsymbol{\theta}|\boldsymbol{y})$ berechnet, wenn ein geeigneter A-priori Wert für $p(\boldsymbol{\theta})$ bekannt ist.

Da der Gauß-Prozess, mit dem wir uns im Folgenden beschäftigen, zu der Klasse der konjungierten A-priori Verteilungen zählt, wollen wir diese definieren.

Definition 2.5 (konjungierte A-priori Verteilung).
Eine Klasse P von A-priori Verteilungen für den Parameter $\boldsymbol{\theta}$ heißt konjungiert für das Modell $p(\boldsymbol{y}|\boldsymbol{\theta})$ falls

$$p(\boldsymbol{\theta}) \in P \Rightarrow p(\boldsymbol{\theta}|\boldsymbol{y}) \in P.$$

Das bedeutet, dass die A-posteriori Verteilung $p(\boldsymbol{\theta}|\boldsymbol{y})$ in der selben Verteilungsklasse P liegt wie die A-priori Verteilung $p(\boldsymbol{\theta})$.

2.3 Gauß-Prozess

Nun haben wir alle Grundlagen definiert um den Gauß-Prozess und seine Vor- und Nachteile näher zu betrachten. Laut *C. E. Rasmussen & C. K. I. Williams* ist ein Gauß-Prozess [Rasmussen and Williams, 2006] folgendermaßen definiert:

Definition 2.6 (Gauß-Prozess).
Ein Gauß-Prozess ist eine Kollektion von Zufallsvektoren $\{f(\boldsymbol{x})|\boldsymbol{x} \in \boldsymbol{X}\}$*, die eine gemeinsame Gaußverteilung besitzen. D.h. für alle* $\boldsymbol{x} = (x_1, \ldots, x_n) \in \boldsymbol{X}$ *sind* $f(x_1), \ldots, f(x_n)$ *normalverteilt.*

Somit ist der Gauß-Prozess wie die Gaußverteilung eindeutig durch seine ersten beiden Momente bestimmt. Der einzige Unterschied zur Gaußverteilung ist, dass beim Gauß-Prozess nicht der Erwartungswertvektor $\boldsymbol{\mu}$ und die Varianz-Kovarianzmatrix Σ die ersten beiden Momente bestimmen, sondern die Erwartungswert-Funktion $\boldsymbol{m}(\boldsymbol{x})$ und die Kovarianz-Funktion $k(\boldsymbol{x}, \boldsymbol{x_*})$ diese Aufgabe übernehmen. Außerdem besitzt der Gauß-Prozess die selben Eigenschaften wie eine Gaußverteilung.

Wir führen folgende Notation für einen Gauß-Prozess ein:

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x}) &\sim GP(\boldsymbol{m}(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x_*})), \\ \text{wobei } \boldsymbol{m}(\boldsymbol{x}) = E(\boldsymbol{f}(\boldsymbol{x})),\ & k(\boldsymbol{x}, \boldsymbol{x_*}) = E[(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{m}(\boldsymbol{x}))(\boldsymbol{f}(x_*) - \boldsymbol{m}(\boldsymbol{x_*})^T)]. \end{aligned} \tag{2.3}$$

Mit Hilfe des Gauß-Prozesses können wir nun auch Regressionsprobleme lösen. Dabei nehmen wir an, dass unser Modell A-priori normalverteilt ist und durch die Definition des Gauß-Prozesses eindeutig durch seine Erwartungswert- und Kovarianz-Funktion bestimmt ist. Sind nun die Erwartungswert- und Kovarianz-Funktion, sowie die Trainingsdaten bekannt, können wir mit Hilfe dieser Information den Erwartungswertvektor und die Varianz-Kovarianzmatrix der Testdaten berechnen. Wie diese Berechnungen verlaufen werden im Kapitel 3 genauer beschrieben.

Vor- und Nachteile des Gauß-Prozesses

In der Arbeit von *D. K. Duvenaud* [Duvenaud, 2014] werden folgende nützliche Eigenschaften des Gauß-Prozesses beschrieben:

- **Analytische Inferenz**
  Die geschätzte A-posteriori Verteilung kann exakt berechnet werden, falls die geeignete Kovarianz-Funktion und einige Trainingsdaten bekannt sind.

- **Ausdrucksstärke**
  Je nach Wahl der Kovarianz-Funktion können andere Annahmen für das Modell getroffen werden.

- **Integration über Hypothesen**
  Mit einer fixen Kovarianz-Funktion kann der A-posteriori Erwartungswertvektor und die Varianz-Kovarianzmatrix des Gauß-Prozesses exakt über eine große Anzahl an Hypothesen integriert werden. Diese Eigenschaft hat zur Folge, dass die Wahrscheinlichkeit, dass das Modell eines Gauß-Prozesses überangepasst ist, geringer ist, als bei vergleichbaren Modellklassen. Ein Modell ist dann überangepasst, falls das Modell zusätzliche, irrelevante Faktoren enthält [Backhaus et al., 2006]. Man nennt diese Eigenschaft auch „Overfitting“.

- **Modellselektion**
  Ein Nebeneffekt der Eigenschaft, dass wir über alle Hypothesen integrieren können, ist, dass wir die marginale Likelihood von Daten berechnen können, wenn wir ein Modell gegeben haben. Diese Eigenschaft macht es uns leichter verschiedene Modelle zu vergleichen.

- **Geschlossene Vorhersageverteilung**
  Die Vorhersageverteilung des Gauß-Prozesses für Testdaten entspricht einer Gaußverteilung. Deshalb können Gauß-Prozesse relativ einfach mit anderen Modellen oder Entscheidungsprozeduren kombiniert werden.

- **Einfache Analyse**
  Simple Modelle wie z.B. lineare Modelle sind einfach zu analysieren. Diese einfachen Modelle können genutzt werden um interessantere und kompliziertere Modelle zu konstruieren.

Neben den Vorteilen von Gauß-Prozessen beschreibt *D. K. Duvenaud* [Duvenaud, 2014] in seiner Arbeit auch einige Nachteile des Gauß-Prozesses:

- **Langsame Inferenz**
  Wie wir später sehen werden wird - für jede Berechnung des A-posteriori Erwartungsvektors und der Varianz-Kovarianzmatrix - die Inverse der Kovarianz-Funktion benötigt. Dadurch wird die Berechnung komplexer als bei anderen Methoden.

- **Geringe Abweichungen**
  Da die Vorhersageverteilung Gaußverteilt ist, gibt es nur geringe Abweichungen der Vorhersageverteilung an den Rändern.

- **Wahl einer Kovarianz-Funktion**
  Da der Gauß-Prozess sehr flexibel ist, stellt sich die Frage welche Kovarianz-Funktion für ein gegebenes Problem gewählt werden soll. Diese Frage ist nicht immer einfach zu beantworten. Es braucht oft einiges an Zeit, Erfahrung und Versuche um eine geeignete Kovarianz-Funktion zu finden (vgl. Kapitel 3, Abschnitt 3).

Kapitel 3

Gauß-Prozess-Regression

Im folgenden Kapitel werden jene Prinzipien der Gauß-Prozess-Regression erläutert, mit denen wir im Kapitel 4 die vorliegenden *ALICe* 1 Daten untersuchen wollen. Neben den Grundlagen einer ein- und multidimensionalen Gauß-Prozess-Regression (Abschnitt 1 & Abschnitt 2) wollen wir auch klären wie eine passende Kovarianz-Funktion (Abschnitt 3) mit Hyperparametern (Abschnitt 4), sowie eine geeignete Erwartungswert-Funktion (Abschnitt 5) gefunden werden kann. Die theoretische Grundlage dieses Kapitels bilden vor allem die Arbeiten von *C. E. Rasmussen & C. K. I. Williams* [Rasmussen and Williams, 2006] und *D. K. Duvenaud* [Duvenaud, 2014].

1 Grundlagen der Gauß-Prozess-Regression

Wie schon in Kapitel 2 erwähnt, können mit Hilfe eines Gauß-Prozesses auch Regressionsprobleme gelöst werden.

Gegeben sei also ein einfaches lineares Regressionsmodell der Form:

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\boldsymbol{y}} = \underbrace{\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}}_{(\mathbb{1}\boldsymbol{x})^T} \underbrace{\begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}}_{\boldsymbol{\beta}} + \underbrace{\begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}}_{\boldsymbol{\epsilon}}.$$

Dieses Regressionsmodell kann auch folgendermaßen geschrieben werden:

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon} \qquad f(\boldsymbol{x}) = (\mathbb{1}\boldsymbol{x})^T\boldsymbol{\beta}.$$

Wir betrachten also ein Regressionsmodell, dass *n beobachtete Werte* (*Response*) $\boldsymbol{y}$ enthält. Diese beobachteten Werte können mit Hilfe einer *Zielfunktion* $f(\boldsymbol{x})$, bestehend aus einer *erklärenden Variable* (*Inputvektor, Regressor, Faktor*) mit *Intercept* $(\mathbb{1}\boldsymbol{x})^T$ und einem *unbekannten Gewicht* $\boldsymbol{\beta}$, und mit Hilfe von *unbeobachteten Fehlern* $\boldsymbol{\epsilon}$ beschrieben werden. Es wird angenommen, dass für die Fehler folgende Normalverteilung gilt:

$$\boldsymbol{\epsilon}_i \overset{iid}{\sim} N(0, \sigma^2).$$

Bei einer typischen linearen Regression würden wir nun passende Modelle auswählen, die unsere Response $\boldsymbol{y}$ mit Hilfe von geschätzten $\hat{\boldsymbol{\beta}}$ und dem Faktor $\boldsymbol{x}$ passend beschreibt. Details zur Durchführung von linearen Regressionen findet man in der Masterarbeit von *G. Gößler* [Gößler, 2015].

Anders geht man bei der sogenannten Gauß-Prozess-Regression vor. Hier nehmen wir an, dass die A-priori Funktionswerte $f(\boldsymbol{x})$ einem Gauß-Prozess (2.3) mit bekannter Erwartungswert-Funktion $\boldsymbol{m}(\boldsymbol{x})$ und Kovarianz-Funktion $k(\boldsymbol{x}, \boldsymbol{x_*})$ entspricht

$$f(\boldsymbol{x}) \sim N(\boldsymbol{m}(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x_*})).$$

Nehmen wir an, dass unsere Erwartungswert-Funktion und Kovarianz-Funktion folgendermaßen definiert sind:

$$\begin{aligned} \boldsymbol{m}(\boldsymbol{x}) &= 0 \\ k(\boldsymbol{x}, \boldsymbol{x_*}) &= \exp\left(-\frac{1}{2}\left|\boldsymbol{x} - \boldsymbol{x_*}\right|^2\right). \end{aligned} \tag{3.1}$$

Diese Kovarianz-Funktion, die wir in (3.1) verwenden, wird auch *quadratische Exponentialfunktion* genannt. In den späteren Abschnitten 3 und 4 werden wir die Wahl der Kovarianz-Funktion und Erwartungswert-Funktion genauer diskutieren.

Nun können wir mit diesem Wissen und dem Inputvektor $\boldsymbol{x}$ zufällige Funktionen aus unserem A-Priori Gauß-Prozess erzeugen.

Beispiel 3.1 (Zufällige Gauß-Prozesse)**.**
Die Idee zu dem folgenden Code, zur Erstellung des Plots (Abb. 3.1) stammt von *J. Keirstead* [Keirstead, 2012].

Wir erstellen zunächst eine beliebige Sequenz $\boldsymbol{x_*}$ und berechnen mit Hilfe der Kovarianz-Funktion (3.1) $k(\boldsymbol{x_*}, \boldsymbol{x_*})$ die A-posteriori Varianz-Kovarianzmatrix an der Stelle $\boldsymbol{x_*}$.

```
> x.star <- seq(-5,5,len=30)
```

```
> cov_funct_matrix_algo <- function(X1,X2,l=1, sigma_f=1){
+ cov_funct_matrix <- matrix(rep(0,length(X1)*length(X2)),nrow=length(X1))
+ for (i in 1:nrow(cov_funct_matrix)) {
+ for (j in 1:ncol(cov_funct_matrix)) {
+ cov_funct_matrix[i,j] <- sigma_f^2*exp(-0.5*(abs(X1[i]-X2[j])/l)^2)}}
+ return(cov_funct_matrix)}
```

```
>cov_funct_matrix<- cov_funct_matrix_algo(x.star,x.star)
```

Anschließend werden die Funktionwerte $f(\boldsymbol{x_*})$ mit Hilfe des `R`-Pakets `mvtnorm` und der Funktion `rmvnorm` simuliert und geplottet. Für den Plot wird die im `R`-Paket `ggplot2` enthaltene Funktion `ggplot` verwendet.

```
> n.samples <- 3

> values <- matrix(rep(0,length(x.star)*n.samples), ncol=n.samples)
> for (i in 1:n.samples) {
+ values[,i] <- rmvnorm(1, mean = rep(0, nrow(cov_funct_matrix)),
+ sigma = cov_funct_matrix) }

> ggplot(values,aes(x=x,y=value)) +
+ geom_rect(xmin=-Inf, xmax=Inf, ymin=-2, ymax=2, fill="grey80") +
+ geom_line(aes(group=variable)) +
+ theme_bw() +
+ scale_y_continuous(lim=c(-2.5,2.5), name="output, f(x)") +
+ xlab("input, x")
```

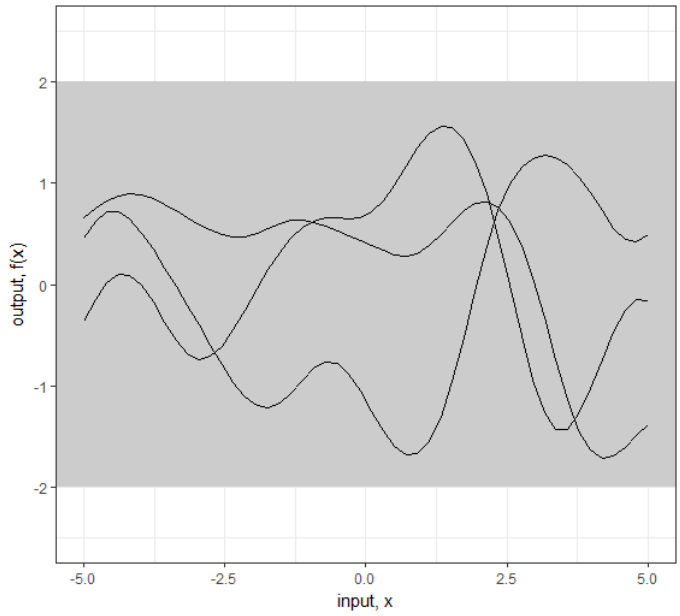

Abbildung 3.1: Drei Realisierungen eines zufälligen Gauß-Prozesses.

Wir wollen nicht nur Plots von zufälligen Funktionen erstellen, sondern auch die A-posteriori Werte unserer Funktion $f(\boldsymbol{x})$ schätzen.

Dazu gehen wir davon aus, dass unsere Inputvariablen in zwei Datensätze getrennt werden können. Für einen Datensatz, den sogenannten Trainingsdaten $\boldsymbol{x} = (x_1, \ldots, x_n)$, kennen wir neben den Werten der erklärenden Variablen auch die dazugehörigen Funktionswerte $\boldsymbol{f} = (f(x_1), \ldots, f(x_n))$. Von dem zweiten Datensatz, den Testdaten $\boldsymbol{x_*} = (x_1^*, \ldots, x_{n^*}^*)$, kennen wir nur die erklärenden Variablen.

Weiters gehen wir davon aus, dass die Beobachtungen zunächst rauschfrei sind. Das bedeutet, dass $f(\boldsymbol{x}) = \boldsymbol{y}$ entspricht. Außerdem nehmen wir an, dass unsere Erwartungswert-

Funktion $\boldsymbol{m}(\boldsymbol{x})$ auf 0 gesetzt wird. Ziel ist es, die A-posteriori Funktionswerte der Testdaten $\boldsymbol{f_*} = (f(x_1^*), \ldots, f(x_{n^*}^*))$ zu schätzen.

Die gemeinsame A-priori Verteilung der Funktionswerte der Trainingsdaten $\boldsymbol{f}$ und der Funktionswerte der Testdaten $\boldsymbol{f_*}$ kann nun folgendermaßen angenommen werden:

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f_*} \end{bmatrix} \sim N\left(0, \begin{bmatrix} k(\boldsymbol{x},\boldsymbol{x}) & k(\boldsymbol{x},\boldsymbol{x_*}) \\ k(\boldsymbol{x_*},\boldsymbol{x}) & k(\boldsymbol{x_*},\boldsymbol{x_*}) \end{bmatrix}\right), \tag{3.2}$$

wobei $k(\boldsymbol{x},\boldsymbol{x_*})$ der $n \times n^*$-Matrix entspricht, die entsteht, falls wir die Kovarianz-Funktion bezüglich aller Paare der Trainings- und Testdaten berechnen. Analog dazu sind die Matrizen $k(\boldsymbol{x},\boldsymbol{x})$, $k(\boldsymbol{x_*},\boldsymbol{x})$ und $k(\boldsymbol{x_*},\boldsymbol{x_*})$ definiert.

Wir wissen aus der Eigenschaft (2.1), dass die bedingte A-posteriori Funktion für die Testdaten ebenfalls einem Gauß-Prozess entspricht. Also erhalten wir:

$$\boldsymbol{f_*}|\,(\boldsymbol{x_*},\boldsymbol{x},\boldsymbol{f}) \sim N\left(\left(k(\boldsymbol{x_*},\boldsymbol{x})k(\boldsymbol{x},\boldsymbol{x})^{-1}\boldsymbol{f}\right), \left(k(\boldsymbol{x_*},\boldsymbol{x_*}) - k(\boldsymbol{x_*},\boldsymbol{x})k(\boldsymbol{x},\boldsymbol{x})^{-1}k(\boldsymbol{x},\boldsymbol{x_*})\right)\right). \tag{3.3}$$

Die geschätzten Funktionswerte der Testdaten $\boldsymbol{f_*}$ erhalten wir, indem wir Stichproben mit Hilfe des A-posteriori Erwartungswertvektors und der A-posteriori Varianz-Kovarianzmatrix aus der obigen Gleichung (3.3) erzeugen.

Beispiel 3.2 (Gauß-Prozess-Regression mit rauschfreien A-priori Beobachtungen)**.**
Wir betrachten nun das Beispiel aus Kapitel 2 der Arbeit von *C. E. Rasmussen & C. K. I. Williams* [Rasmussen and Williams, 2006]. Die Idee zum folgenden `R`-Code stammt erneut von *J. Keirstead* [Keirstead, 2012].

Folgende A-priori Beobachtungen sind gegeben:

$$\begin{aligned} \boldsymbol{x} &= (-4,-3,-1,0,2) \\ \boldsymbol{y} &= (-2,-1,0,1,2). \end{aligned} \tag{3.4}$$

Als Testdaten wählen wir eine beliebige Sequenz $\boldsymbol{x_*}$.

```
> x.star <- seq(-5,5,len=30)
```

```
[1] -5.0000000 -4.6551724 -4.3103448 -3.9655172 -3.6206897 -3.2758621
[7] -2.9310345 -2.5862069 -2.2413793 -1.8965517 -1.5517241 -1.2068966
[13]-0.8620690 -0.5172414 -0.1724138  0.1724138  0.5172414  0.8620690
[19] 1.2068966  1.5517241  1.8965517  2.2413793  2.5862069  2.9310345
[25] 3.2758621  3.6206897  3.9655172  4.3103448  4.6551724  5.0000000
```

Weiters nehmen wir an, dass die Kovarianz-Funktion der quadratischen Exponentialfunktion (3.1) entspricht. Um die geschätzten Funktionswerte von $\boldsymbol{x_*}$ zu erhalten, erzeugen wir Stichproben mit Hilfe des Erwartungswertvektors und der Varianz-Kovarianzmatrix aus Gleichung (3.3). Die Ergebnisse werden anschließend mit der `R`-Funktion `ggplot` geplottet. (Abb. 3.2)

```
> k.xx <- cov_funct_matrix_algo(x,x)
> k.xsx <- cov_funct_matrix_algo(x.star,x)
> k.xsxs <- cov_funct_matrix_algo(x.star,x.star)

> f.star.bar <- k.xsx%*%solve(k.xx)%*%f$y
> cov.f.star <- k.xsxs - k.xsx%*%solve(k.xx)%*% t(k.xsx)

> n.samples <- 50
> values <- matrix(rep(0,length(x.star)*n.samples), ncol=n.samples)
> for (i in 1:n.samples) {
+ values[,i] <- mvrnorm(1, f.star.bar, cov.f.star)}
```

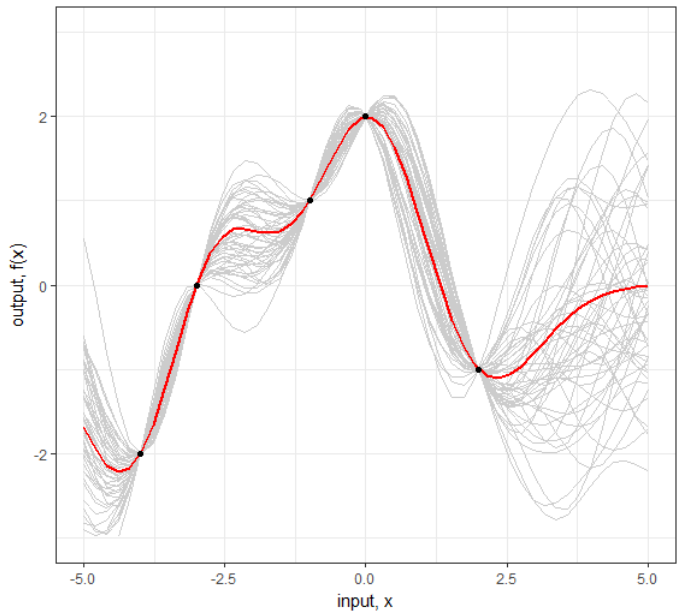

Abbildung 3.2: Bedingter Gauß-Prozess mit rauschfreien A-priori Beobachtungen.

Rauschfreie Beobachtungen sind in der Praxis nicht häufig vorzufinden. Deshalb betrachten wir nun den Fall, dass unsere Beobachtungen ein Rauschen beinhalten:

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon} \qquad \boldsymbol{\epsilon}_i \overset{iid}{\sim} N(0, \sigma_n^2).$$

Dadurch verändert sich die Berechnung der A-priori Kovarianz-Funktion folgendermaßen:

$$cov(\boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 I, \ I \in \mathbb{R}^{nxn}.$$

Da wir annehmen, dass unser Rauschen $\boldsymbol{\epsilon}$ aus unabhängigen Komponenten ϵ_i besteht, wird eine Diagonalmatrix zu unserem rauschfreien Fall (3.2) hinzugefügt.

Die gemeinsame A-priori Verteilung ändert sich dementsprechend:

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 I & k(\boldsymbol{x}, \boldsymbol{x}_*) \\ k(\boldsymbol{x}_*, \boldsymbol{x}) & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right). \tag{3.5}$$

Daraus erhalten wir folgende bedingte A-posteriori Verteilung für $\boldsymbol{f_*}$

$$\boldsymbol{f_*}|\left(\boldsymbol{x_*}, \boldsymbol{x}, \boldsymbol{y}\right) \sim N\left(\overline{\boldsymbol{f_*}}, cov(\boldsymbol{f_*})\right), \tag{3.6}$$

wobei

$$\overline{\boldsymbol{f_*}} = E[\boldsymbol{f_*}|\boldsymbol{x_*}, \boldsymbol{x}, \boldsymbol{y}] = k(\boldsymbol{x_*}, \boldsymbol{x})[k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 I]^{-1}\boldsymbol{y} \tag{3.7}$$

$$cov(\boldsymbol{f_*}) = k(\boldsymbol{x_*}, \boldsymbol{x_*}) - k(\boldsymbol{x_*}, \boldsymbol{x})[k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 I]^{-1}k(\boldsymbol{x}, \boldsymbol{x_*}). \tag{3.8}$$

Die geschätzten Funktionswerte $\boldsymbol{f_*}$ erhalten wir wieder indem wir Stichproben mit dem Erwartungswertvektor $\overline{\boldsymbol{f_*}}$ und der Varianz-Kovarianzmatrix $cov(\boldsymbol{f_*})$ erzeugen.

Zu erwähnen ist hier, dass die Varianz-Kovarianzmatrix $cov(\boldsymbol{f_*})$ unabhängig von den beobachteten Zielwerten der Trainingsdaten ist. Die Varianz-Kovarianzmatrix hängt nur von den Inputvariablen ab. Diese Eigenschaft der Gaußverteilung berechnet die Varianz-Kovarianzmatrix als Differenz zwischen zwei Termen. Der erste Term $k(\boldsymbol{x_*}, \boldsymbol{x_*})$ beschreibt die A-priori Kovarianz. Der zweite positive Term, der abgezogen wird, beschreibt die Information, die uns die Trainingsdaten über die Funktion liefern. Durch diese Eigenschaft können wir die Vorhersage der Response für die Testdaten $\boldsymbol{y}_*$ berechnen, indem wir den Term $\sigma_n^2 I$ zur Varianz-Kovarianzmatrix $cov(\boldsymbol{f_*})$ hinzufügen und mit Hilfe dieser Matrix und dem unveränderten Erwartungswertvektor $\overline{\boldsymbol{f_*}}$ Stichproben erstellen.

Beispiel 3.3 (Gauß-Prozess-Regression mit A-priori Beobachtungen, die ein Rauschen beinhalten)**.**
Gegeben seien dieselben Beobachtungen und die Testdaten wie in Beispiel 3.2.

Dieses Mal besitzen die A-priori Beobachtungen allerdings ein Rauschen mit $\sigma_n = 0.1$. Weiters wird die quadratische Exponentialfunktion (3.1) als Kovarianz-Funktion gewählt. Als Ergebnis erhalten wir Abb. 3.3.

```
> sigma.n <- 0.1

> f.bar.star <- k.xsx%*%solve(k.xx + sigma.n^2*diag(1, ncol(k.xx)))%*%f$y
> cov.f.star <- k.xsxs - k.xsx%*%solve(k.xx + sigma.n^2*diag(1, ncol(k.xx)))
+ %*%t(k.xsx)

> n.samples <- 50
> values <- matrix(rep(0,length(x.star)*n.samples), ncol=n.samples)
> for (i in 1:n.samples) {
+ values[,i] <- mvrnorm(1, f.bar.star, cov.f.star)}
```

2 Multivariate Gauß-Prozess-Regression

Bis jetzt nahmen wir immer an, dass unsere Response $\boldsymbol{y} = (y_1, \ldots, y_n)^T$ nur von einem Faktor $\boldsymbol{x}$ abhängt. In der Praxis ist dies nur selten der Fall. Deshalb wollen wir in diesem Abschnitt die Gauß-Prozess-Regression für m Faktoren $\boldsymbol{x_j}$ betrachten. $\boldsymbol{x_j} = (x_{1j}, \ldots, x_{nj})^T$

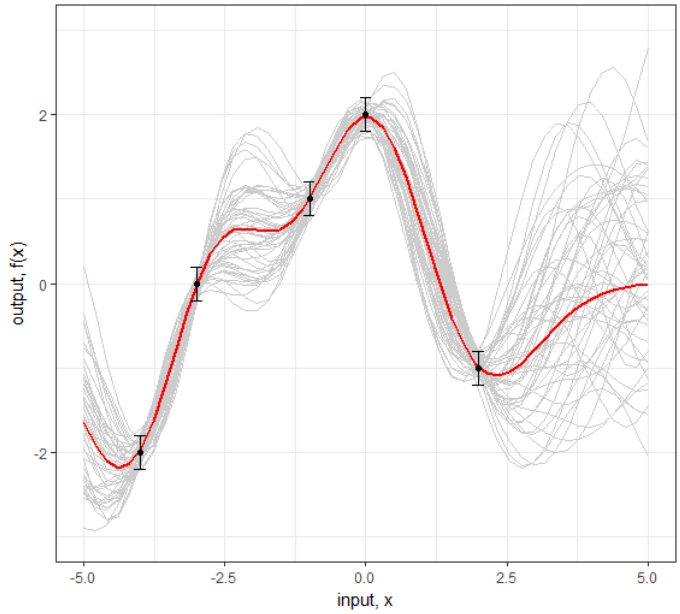

Abbildung 3.3: Bedingter Gauß-Prozess mit A-priori Beobachtungen die ein Rauschen beinhalten.

für alle $j = 1, \ldots, m$.

Sei dazu nun folgendes Regressionsmodell gegeben:

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_n \end{pmatrix}}_{\boldsymbol{y}} = \underbrace{\begin{pmatrix} 1 & x_{11} & \cdots & x_{1j} & \cdots & x_{1m} \\ 1 & x_{21} & \cdots & x_{2j} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nj} & \cdots & x_{nm} \end{pmatrix}}_{X} \underbrace{\begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{pmatrix}}_{\boldsymbol{\beta}} + \underbrace{\begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \vdots \\ \epsilon_n \end{pmatrix}}_{\boldsymbol{\epsilon}}.$$

Oder anders geschrieben:

$$\boldsymbol{y} = f(X) + \boldsymbol{\epsilon}, \qquad f(X) = X\boldsymbol{\beta}, \qquad \epsilon_i \overset{iid}{\sim} N(0, \sigma_n^2).$$

Es gibt nun zwei Möglichkeiten, wie dieses Regressionsmodell mit Hilfe der Gauß-Prozess-Regression gelöst werden kann:

- Additiver Ansatz
- Abstandsnorm Ansatz

2.1 Additiver Ansatz

Für den additiven Ansatz [Duvenaud, 2014] nehmen wir an, dass unsere Response $\boldsymbol{y}$ von $m = 2$ Faktoren $\boldsymbol{x_1} = (x_{11}, \ldots, x_{n1})$ und $\boldsymbol{x_2} = (x_{12}, \ldots, x_{n2})$ abhängig ist. Weiters seien

diese zwei Faktoren paarweise unabhängig. Dadurch können wir für jeden Faktor einen eigenen Gauß-Prozess betrachten.

$$\begin{aligned} f_1(\boldsymbol{x_1}) &\sim N(\boldsymbol{\mu_1}, k_1), \\ f_2(\boldsymbol{x_2}) &\sim N(\boldsymbol{\mu_2}, k_2), \\ f(\boldsymbol{x_1}, \boldsymbol{x_2}) = f_1(\boldsymbol{x_1}) + f_2(\boldsymbol{x_2}) &\sim N(\boldsymbol{\mu_1} + \boldsymbol{\mu_2}, k_1 + k_2). \end{aligned}$$

Dabei sind $f_1(\boldsymbol{x_1})$ die Funktionswerte, die wir erhalten, wenn wir die Werte von Faktor $\boldsymbol{x_1}$ betrachten und $f_2(\boldsymbol{x_2})$ die Funktionswerte, wenn wir Faktor $\boldsymbol{x_2}$ einsetzen. Weiters ist $\boldsymbol{\mu_1}$ die Erwartungswert-Funktion und $k_1 = k_1(\boldsymbol{x_1}, \boldsymbol{x_1^*})$ die Kovarianz-Funktion von Faktor $\boldsymbol{x_1}$. Analog sind $\boldsymbol{\mu_2}$ und $k_2 = k_2(\boldsymbol{x_2}, \boldsymbol{x_2^*})$ gegeben.

Seien nun $f(X)$ die Funktionswerte ausgewertet an den Trainingsdaten

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} \end{pmatrix}$$

und $f(X^*)$ die gesuchten Funktionswerte der Testdaten

$$X^* = \begin{pmatrix} 1 & x_{11}^* & x_{12}^* \\ 1 & x_{21}^* & x_{22}^* \\ \vdots & \vdots & \vdots \\ 1 & x_{n1}^* & x_{n2}^* \end{pmatrix}.$$

Dann erhalten wir folgende gemeinsame A-priori Verteilung von $f_1(\boldsymbol{x_1}) \sim N(\boldsymbol{\mu_1}, k_1)$ und $f_2(\boldsymbol{x_2}) \sim N(\boldsymbol{\mu_2}, k_2)$:

$$\begin{bmatrix} f_1(X) \\ f_1(X^*) \\ f_2(X) \\ f_2(X^*) \\ f_1(X) + f_2(X) \\ f_1(X^*) + f_2(X^*) \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{\mu_1} \\ \boldsymbol{\mu_1^*} \\ \boldsymbol{\mu_2} \\ \boldsymbol{\mu_2^*} \\ \boldsymbol{\mu_1} + \boldsymbol{\mu_2} \\ \boldsymbol{\mu_1^*} + \boldsymbol{\mu_2^*} \end{bmatrix}, \begin{bmatrix} K_1 & K_1^* & 0 & 0 & K_1 & K_1^* \\ K_1^{*^T} & K_1^{**} & 0 & 0 & K_i^* & K_i^{**} \\ 0 & 0 & K_2 & K_2^* & K_2 & K_2^* \\ 0 & 0 & K_2^{*^T} & K_2^{**} & K_2^* & K_2^{**} \\ K_1 & K_1^{*^T} & K_2 & K_2^{*^T} & K_1 + K_2 & K_1^* + K_2^* \\ K_1^{*^T} & K_1^{**} & K_2^{*^T} & K_2^{**} & K_1^{*^T} + K_2^{*^T} & K_1^{**} + K_2^{**} \end{bmatrix} \right),$$

wobei hier die Kovarianzmatrizen mit Hilfe der jeweiligen Kovarianz-Funktion berechnet wurden:

$$\begin{aligned} K_i &= k_i(X, X), \\ K_i^* &= k_i(X, X^*), \\ K_i^{**} &= k_i(X^*, X^*). \end{aligned}$$

Dann erhalten wir folgende bedingte Verteilung für den gesuchten Funktionwert $f_1(X^*)$:

$$\begin{aligned} &f_1(X^*)|(f_1(X) + f_2(X)) \sim \\ &\quad N\left(\boldsymbol{\mu_1^*} + K_1^{*^T}(K_1 + K_2)^{-1}\left[f_1(X) + f_2(X) - \boldsymbol{\mu_1} - \boldsymbol{\mu_2}\right], K_1^{**} - K_1^{*^T}(K_1 + K_2)^{-1}K_1^*\right). \end{aligned} \tag{3.9}$$

Falls eine Response $\boldsymbol{y}$ von mehr als nur zwei Faktoren abhängt, setzen wir statt dem Term $K_1 + K_2$ in (3.9), den Term $\sum_i K_i$.

Wie diese Methode in der Praxis angewandt wird, kann in der Arbeit von *D. K. Duvenaud* [Duvenaud, 2014] nachgelesen werden. Bei unserer praktischen Umsetzung in Kapitel 4 konnte diese Methode nicht verwendet werden, da unsere Faktoren paarweise abhängig sind.

2.2 Abstandsnorm Ansatz

Nicht immer ist es der Fall, dass die Response $\boldsymbol{y}$ von paarweise unabhängigen Faktoren abhängt. Deshalb benötigen wir einen anderen Ansatz. Beim Abstandsnorm Ansatz ändert sich im Vergleich zur Gauß-Prozess-Regression mit einem Faktor, nur die Berechnung der Kovarianz-Funktion. Alle anderen Schritte bleiben analog zu Abschnitt 1.

Für die Berechnung der Kovarianz-Funktion benötigen wir die Definition des euklidischen Abstandes [Bronstein et al., 2001b]:

Definition 3.1 (**Euklidischer Abstand**).
Seien $\boldsymbol{x} = (x_1, \ldots, x_n)$ *und* $\boldsymbol{y} = (y_1, \ldots, y_n)$ *zwei Vektoren, dann wird ihre euklidische Abstand (L_2-Norm) folgendermaßen berechnet:*

$$\|\boldsymbol{x} - \boldsymbol{y}\|_2 = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2}. \tag{3.10}$$

Die euklidische Norm (3.10) wird statt dem Abstand in der jeweiligen Kovarianz-Funktion eingesetzt. Das bedeutet, dass sich z.B. die Berechnung der im Abschnitt 1 vorgestellten quadratischen Exponentialfunktion (3.1) folgendermaßen verändert:

$$k(X, X^*) = \exp\left(-\tfrac{1}{2}\|X - X^*\|_2^2\right). \tag{3.11}$$

Der in Beispiel 3.1 vorgestellte `R`-Code zur Berechnung der quadratischen Exponentialfunktion ändert sich wie folgt:

```
> norm_berechnen <- function(x1,x2) {
+ abstand<- c(1:length(x1))
+  for (k in 1:length(x1)){
+  abstand[k] <- x1[,k]-x2[,k]
+  xnorm <- sqrt(sum(abstand^2))}
+  return(xnorm)}

> cov_funct_matrix_algo <- function(X1,X2,l=1, sigma_f=1) {
+ cov_funct_matrix <- matrix(0,nrow(X1),nrow(X2))
+  for (i in 1:nrow(cov_funct_matrix)) {
+  for (j in 1:ncol(cov_funct_matrix)) {
+  X_norm <- norm_berechnen(X1[i,],X2[j,])
+  cov_funct_matrix[i,j] <- sigma_f^2*exp(-0.5*(X_norm/l)^2)}}
+ return(cov_funct_matrix)}
```

Eine praktische Umsetzung dieser Methode wird in Kapitel 4 vorgestellt.

3 Plausible Wahl der Kovarianz-Funktion

Wie wir in den vorherigen Abschnitten 1 und 2 gesehen haben, spielt die Wahl der Kovarianz-Funktion eine zentrale Rolle in der Berechnung der Gauß-Prozess-Regression. Welche Kovarianz-Funktion am geeignesten für das jeweilige Problem ist, hängt vor allem von den gegebenen Daten ab.

In diesem Abschnitt wollen wir einige bekannte Kovarianz-Funktionen (Unterabschnitt 3.1) auflisten und beschreiben wie man diese kombinieren (Unterabschnitt 3.2) kann. Weiters werden Methoden zur Evaluierung (Unterabschnitt 3.3) der verschiedenen Kovarianz-Funktionen für ein gegebenes Regressionsproblem vorgestellt.

3.1 Stationäre Kovarianz-Funktionen

In dieser Arbeit wollen wir uns hauptsächlich auf stationäre und isotrope Kovarianz-Funkionen beschränken. Beispiele für Kovarianz-Funktionen mit anderen Eigenschaften findet man im grundlegenden Kapitel 4 der Arbeit von C. E. Rasmussen & C. K. I. Williams
[Rasmussen and Williams, 2006]

Definition 3.2 (**Stationäre und isotrope Kovarianz-Funktion**).
Eine Kovarianz-Funktion $k(\boldsymbol{x}, \boldsymbol{x})$ ist stationär, falls der Wert der Funktion nur von $\boldsymbol{x} - \boldsymbol{x}^*$ abhängt. Solche stationären Funktionen sind invariant bezüglich Translationen. Das bedeutet, dass die Verteilung von bestimmten Daten $\boldsymbol{x}$ gleich bleibt, wenn man alle Punkte um den selben Wert verschiebt.

Hängt der Wert der Funktion zusätzlich nur von $|\boldsymbol{x} - \boldsymbol{x}^*|$ ab, nennt man die Kovarianz-Funktion auch isotrop. Die Funktion ist dann nicht nur invariant gegenüber Translationen sondern auch Rotationen.

Eine in der Praxis sehr beliebte Kovarianz-Funktion ist die quadratische Exponentialfunktion. Für $l = 1$ und $\sigma_f = 1$ haben wir diese Funktion (3.1) schon im vorherigen Abschnitt eingeführt.

$$k_{SE}(\boldsymbol{x}, \boldsymbol{x^*}) = \sigma_f^2 \exp\left(-\frac{|\boldsymbol{x} - \boldsymbol{x^*}|^2}{2l^2}\right) + \sigma_n^2 I, \qquad \boldsymbol{x} \in \mathbb{R}^n,\ l \in \mathbb{R},\ \sigma_f^2, \sigma_n^2 \in \mathbb{R}_+. \tag{3.12}$$

Die Parameter $l \in \mathbb{R}$ und $\sigma_f^2, \sigma_n^2 \in \mathbb{R}_+$ werden auch als Hyperparameter bezeichnet. Je nachdem auf welchen Wert die Hyperparameter gesetzt werden, ändert sich die Kovarianz-Funktion. In Abb. 3.4 erkennen wir, dass die Funktion glatter wird je größer man den Paramater l wählt. Wie die Werte der Hyperparameter berechnet werden wird in Abschnitt 4 beschrieben.

Neben der quadratischen Exponentialfunktion gibt es noch einige andere stationäre Kovarianz-Funktionen. Einige Beispiele sind in der folgenden Tabelle 3.1 aufgelistet und in Abb. 3.5 grafisch dargestellt.

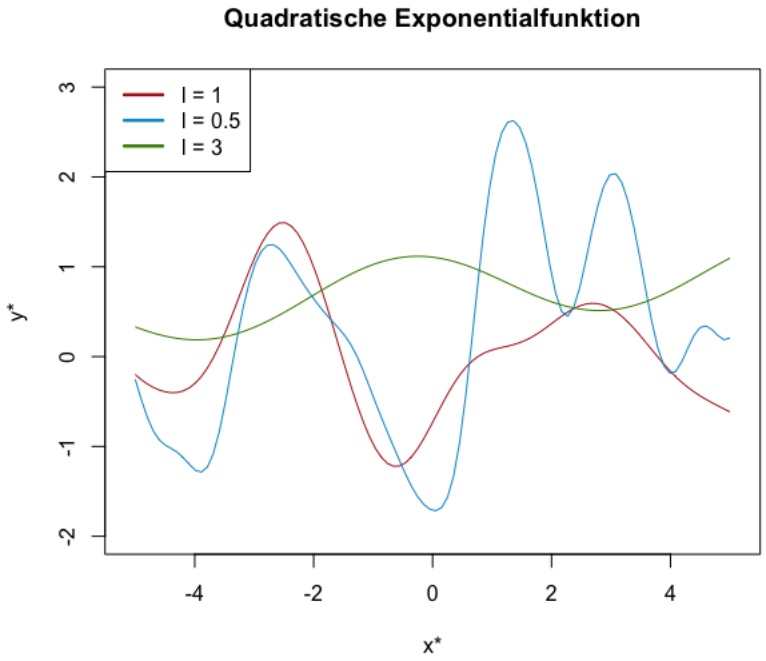

Abbildung 3.4: Quadratische Exponentialfunktion.

- **Periodische Exponentialfunktion**

$$k_{PE}(\boldsymbol{x}, \boldsymbol{x}^*) = \sigma_f^2 \exp\left(-\frac{2}{l^2}\sin^2\left(\pi\frac{\boldsymbol{x}-\boldsymbol{x}^*}{p}\right)\right) + \sigma_n^2 I, \qquad l, p \in \mathbb{R},\ \sigma_f^2, \sigma_n^2 \in \mathbb{R}_+.$$

- **Rational quadratische Exponentialfunktion**

$$k_{RQ}(\boldsymbol{x}, \boldsymbol{x}^*) = \left(1 + \frac{|\boldsymbol{x}-\boldsymbol{x}^*|^2}{2\alpha l^2}\right)^{-\alpha} + \sigma_n^2 I, \qquad l, \alpha \in \mathbb{R},\ \sigma_n^2 \in \mathbb{R}_+.$$

- **Gamma-Exponentialfunktion**

$$k_{GE}(\boldsymbol{x}, \boldsymbol{x}^*) = \exp\left(-\left(\frac{|\boldsymbol{x}-\boldsymbol{x}^*|}{l}\right)^{\gamma}\right) + \sigma_n^2 I \qquad l, \gamma \in \mathbb{R},\ \sigma_n^2 \in \mathbb{R}_+.$$

- **Matern Funktion mit** $\nu = p + \frac{1}{2}$

$$k_{MATERN_{\nu=p+\frac{1}{2}}}(\boldsymbol{x}, \boldsymbol{x}^*) = \exp\left(-\frac{\sqrt{2\nu|\boldsymbol{x}-\boldsymbol{x}^*|}}{l}\right)\frac{\Gamma(p+1)}{\Gamma(2p+1)}\sum_{i=0}^{p}\frac{(p+i)!}{i!(p-i)!}\left(\frac{\sqrt{8\nu|\boldsymbol{x}-\boldsymbol{x}^*|}}{l}\right)^{p-i} + \sigma_n^2 I, \qquad l, p \in \mathbb{R},\ \sigma_n^2 \in \mathbb{R}_+.$$

Tabelle 3.1: Beispiele für stationärer Kovarianz-Funktionen.

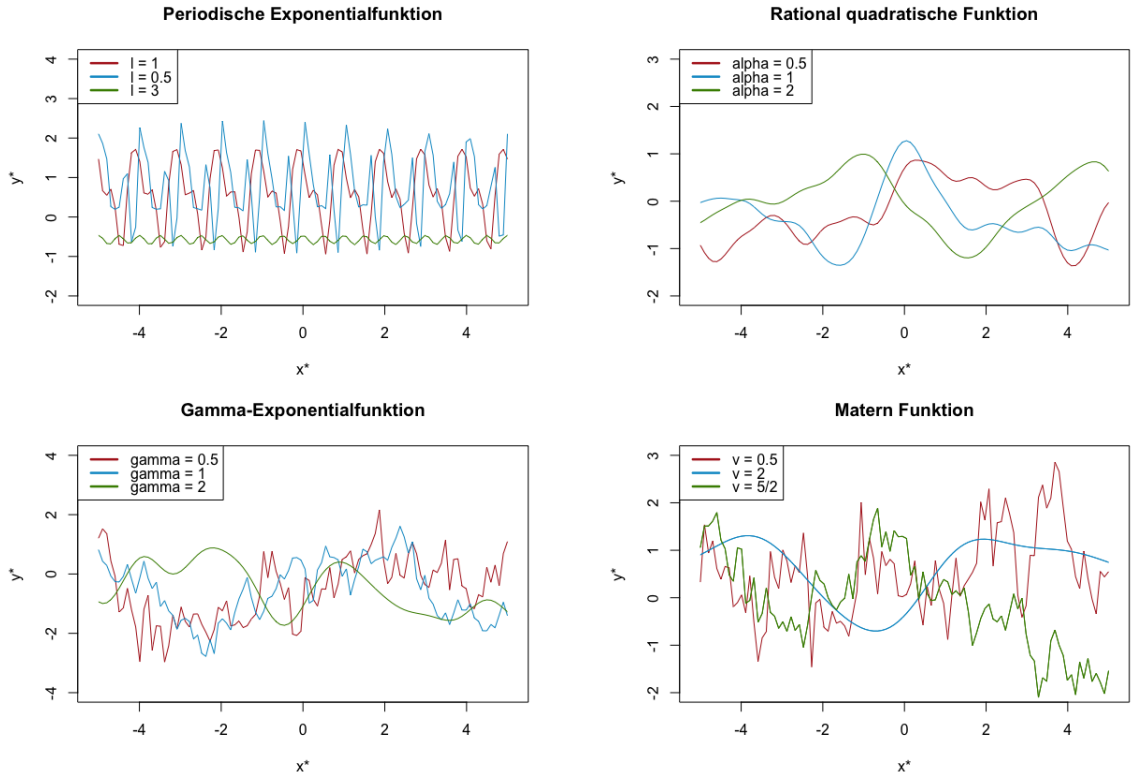

Abbildung 3.5: Weitere Beispiele für stationäre Kovarianz-Funktionen.

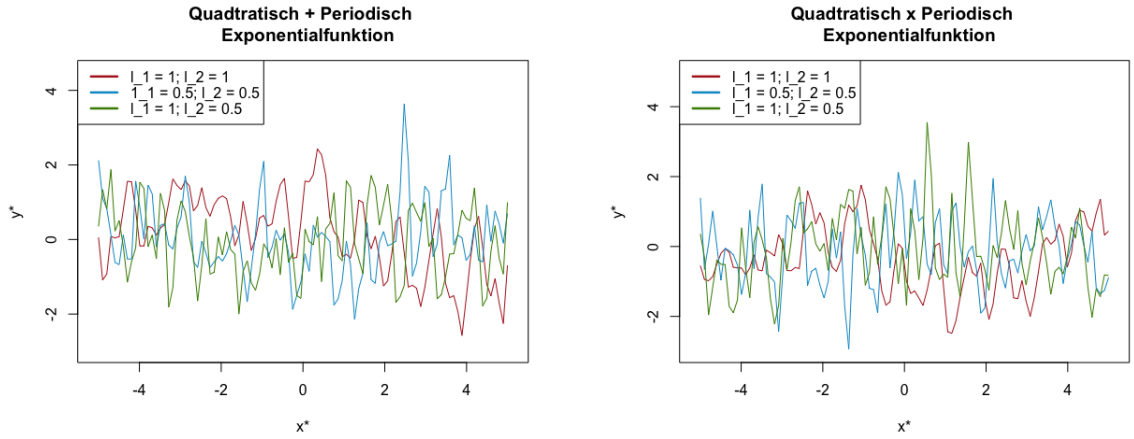

Abbildung 3.6: Kombination mit Hilfe von Addition und Multiplikation.

3.2 Kombinationen von Kovarianz-Funktionen

Wenn keine der in Unterabschnitt 3.1 vorgestellten Kovarianz-Funktionen die gewünschte Struktur für das gegebene Problem liefert, können auch einfache Funktionen miteinander kombiniert werden. Dies kann mit Hilfe von Addition

$$k_a(\boldsymbol{x}, \boldsymbol{x}^*) + k_b(\boldsymbol{x}, \boldsymbol{x}^*)$$

oder mittels Multiplikation

$$k_a(\boldsymbol{x}, \boldsymbol{x}^*) \cdot k_b(\boldsymbol{x}, \boldsymbol{x}^*)$$

geschehen. Abb. 3.6 liefert zwei Beispiele wie man Kovarianz-Funktionen miteinander kombinieren kann.

3.3 Modellselektion

Es gibt eine Vielzahl von Kovarianz-Funktionen und Kombinationen von Kovarianz-Funktionen. Die Frage, die nun aufkommt: Wie finden wir heraus, welche Kovarianz-Funktion für das gegebene Problem geeignet ist?

Eine Möglichkeit ist die marginale log-Likelihood $p(\boldsymbol{y}|\boldsymbol{x})$ der Gauß-Prozess-Regression mit verschiedenen Kovarianz-Funktionen zu berechnen und die Ergebnisse zu vergleichen. Dazu müssen wir zunächst die marginale log-Likelihood für Gauß-Prozesse einführen.

Definition 3.3 (**Marginale log-Likelihood für Gauß-Prozess-Regression**)**.**
Zur Erinnerung: Eine marginale Likelihood ist folgendermaßen definiert:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \int p(\boldsymbol{y}|f, \boldsymbol{x}) p(f|\boldsymbol{x}) df.$$

Unter einem Gauß-Prozess ist f A-priori Gaußverteilt $f|\boldsymbol{x} \sim N(0, K)$. Sei $K = k(\boldsymbol{x}, \boldsymbol{x})$, dann kann diese Verteilung wie folgt geschrieben werden:

$$\log p(\boldsymbol{f}|\boldsymbol{x}) = -\tfrac{1}{2}\boldsymbol{y}^{\boldsymbol{T}} K^{-1}\boldsymbol{y} - \tfrac{1}{2}\log|K| - \tfrac{n}{2}\log 2\pi.$$

Da weiters $\boldsymbol{y} \sim N(0, K + \sigma_n^2 I)$ gilt folgt nun für die marginale Log-Likelihood:

$$\log p(\boldsymbol{y}|\boldsymbol{x}) = -\tfrac{1}{2}\boldsymbol{y}^{\boldsymbol{T}}(K + \sigma_n^2 I)^{-1}\boldsymbol{y} - \tfrac{1}{2}\log|K + \sigma_n^2 I| - \tfrac{n}{2}\log 2\pi. \tag{3.13}$$

Für die Modellselektion berechnen wir die marginale Log-Likelihood für verschiedene Kovarianz-Funktionen und vergleichen diese miteinander. Es gilt, je größer die marginale log-Likelihood, desto eher ist die Kovarianz-Funktion für das Problem geeignet. Allerdings funktioniert diese Methode nur dann wenn die Response nur von einem Faktor oder von mehreren paarweise unabhängigen Faktoren abhängig ist.

Eine weitere allgemeinere Möglichkeit ist einen Teil der beobachteten Daten, die Trainingsdaten, in zwei Mengen zu teilen. Der erste Teil der Daten $\boldsymbol{x}$ wird genutzt um das

Modell zu berechnen, bleiben also Trainingsdaten. Der zweite Teil $\boldsymbol{x}_*$ wird zur Validierung genutzt. Damit wird ein Teil der ursprünglichen Trainingsdaten $\boldsymbol{x}_*$ zu Testdaten, dessen Response $\boldsymbol{y}_*$ wir schätzen wollen. Nachdem die Test- und Trainingsdaten festgelegt wurden, werden mit Hilfe von verschiedenen Kovarianz-Funktionen Gauß-Prozess-Regressionen durchgeführt. Die geschätzten A-posteriori Erwartungswerte $\overline{\boldsymbol{f}_*}$ der verschiedenen Gauß-Prozess-Regressionen werden mit den wahren Werten von $\boldsymbol{x}_*$ verglichen. Je näher der berechnete A-posteriori Erwartungswert $\overline{\boldsymbol{f}_*}$ beim wahren Wert $\boldsymbol{y}_*$ liegt, desto eher ist die Kovarianz-Funktion für das gegebene Problem geeignet.

4 Plausible Wahl der Hyperparameter

Wie wir in Abschnitt 3 gesehen haben, besitzt jede Kovarianz-Funktion eine bestimmte Anzahl an frei wählbaren Parametern, die sogenannten Hyperparameter $\boldsymbol{\theta}$. Da die Gauß-Prozess-Regression ein nicht-parametrisches Modell darstellt, ist nicht immer klar ersichtlich, welche Werte diese Hyperparameter annehmen sollen. Somit wollen wir in diesem Abschnitt Methoden vorstellen, die dabei helfen geeignete Hyperparameter zu finden.

In der Literatur [Rasmussen and Williams, 2006] werden folgende Methoden vorgeschlagen:

- Testen von verschiedenen Werten (Unterabschnitt 4.1),
- Maximierung der marginalen Likelihood-Funktion (Unterabschnitt 4.3).

Da wir aber in der praktischen Umsetzung (Kapitel 4) Hyperparameter finden wollen, die den Gesamtabstand zwischen interpolierten und gemessenen Punkten minimieren, betrachten wir im Folgenden auch zwei eigens entwickelte Methoden:

- Minimierung des Gesamtabstandes mit der L_2-Norm (Unterabschnitt 4.4),
- Minimierung des Gesamtabstandes mit der L_1-Norm (Unterabschnitt 4.5).

Es gibt noch weitere Methoden zur Bestimmung von Hyperparametern, wie z.B. eine Kreuzvalidierung [Rasmussen and Williams, 2006] oder eine Markov-Chain-Monte-Carlo Methode [Rasmussen and Williams, 1996]. Auf diese Methoden wird allerdings in dieser Arbeit nicht weiter eingegangen.

4.1 Testen von verschiedenen Werten

Eine sehr einfache Möglichkeit eine Idee zu bekommen, welche Werte für die Hyperparameter geeignet sind, ist das Testen von verschiedenen Werten. Dazu wählen wir wie zuvor im Unterabschnitt Modellselektion 3.3 einen Teil der Trainingsdaten $\boldsymbol{x}$ als Testdaten $\boldsymbol{x}_*$. Weiters wählen wir für jeden Hyperparameter $\boldsymbol{\theta}$ ein Intervall von möglichen Einstellungen, die dieser Hyperparameter annehmen kann. Anschließend berechnen wir für jede Einstellung des Hyperparameters den A-posteriori Erwartungswertvektor $\overline{\boldsymbol{f}_*}$ der Testdaten $\boldsymbol{x}_*$. Dieser Erwartungswertvektor $\overline{\boldsymbol{f}_*}$ wird anschließend mit den gemessenen Funktionsvektor $\boldsymbol{y}_*$ der Testdaten verglichen. Es gilt, je geringer der Abstand zwischen den extrapolierten und gemessenen Punkten ist, desto besser ist die Einstellung des Hyperparameters für das

Problem geeignet.

Beispiel 3.4 (Testen von verschiedenen Werten)**.**
Nehmen wir an, dass wie in Beispiel 3.2 folgende A-priori Beobachtungen gegeben sind:

$$\boldsymbol{x} = (-4, -3, -1, 0, 2),$$
$$\boldsymbol{y} = (-2, -1, 0, 1, 2).$$

Dann wählen wir folgende Werte aus unseren A-priori Beobachtungen als Testdaten:

$$\boldsymbol{x_*} = (-3, -1),$$
$$\boldsymbol{y_*} = (-1, 0). \tag{3.14}$$

Die übrigen Werte bleiben Trainingsdaten:

$$\boldsymbol{x} = (-4, 0, 2),$$
$$\boldsymbol{y} = (-2, 1, 2). \tag{3.15}$$

Wir wollen die Hyperparameter $\boldsymbol{\theta} = (l, \sigma_f, \sigma_n)$ nun so bestimmen, dass der Erwartungsvektor $\overline{\boldsymbol{f_*}}$ möglichst nahe an den gemessenen Funktionsvektor $\boldsymbol{y_*} = (-1, 0)$ herankommt. Dazu wählen wir als Kovarianz-Funktion die quadratische Exponentialfunktion (3.12) und die möglichen Bereiche der einzelnen Hyperparameter werden folgendermaßen gesetzt:

$$l \in [0.1, 1]\,,\ \sigma_f \in [0.1, 1]\,,\ \sigma_n \in [0, 1]\,.$$

Dann können die Hyperparameter $\boldsymbol{\theta} = (l, \sigma_f, \sigma_n)$ wie folgt bestimmt werden.

```
> l = seq(0.1,1, 0.1)
> sigma_f<- seq(0.1,1,0.1)
> sigma_n <- seq(0,1, 0.1)

> new_Hyper <- matrix(Inf,4 ,1)

> for (i in 1:length(l)){
+ for (j in 1:length(sigma_n)){
+  for (h in 1:length(sigma_f)){
+  K.xx<- cov_funct_matrix_algo(x,x,l[i],sigma_f[h])
+  noise <- sigma_n[j]^2*diag(nrow(K.xx))
+  K.xx_noise <- K.xx+noise
+  if (det( K.xx_noise) == 0){
+   ges_Abstand = Inf } else{
+   K.xstar<- cov_funct_matrix_algo(x.star,x,l[i],sigma_f[h])
+   f.star.bar <-K.xstar%*%solve(K.xx_noise)%*%y
+   abstand<- abs(y.star-f.star.bar)
+   ges_Abstand <- sum(abstand) }
+  if (ges_Abstand <= new_Hyper[4,1]){
+   new_Hyper[1,1] <- l[i]
+   new_Hyper[2,1] <- sigma_f[h]
+   new_Hyper[3,1] <- sigma_n[j]
+   new_Hyper[4,1] <- ges_Abstand }}}}
```

```
> new_Hyper
   [,1]
[1,] 1.0000000
[2,] 0.9000000
[3,] 0.4000000
[4,] 0.4023372
```

Als Ergebnis für die Hyperparameter $\boldsymbol{\theta} = (l, \sigma_f, \sigma_n)$ erhalten wir mit dieser Methode also folgende Werte:

$$\hat{\boldsymbol{\theta}} = (1, 0.9, 0.4). \tag{3.16}$$

Führen wir mit diesen Werten (4.1) nun eine Gauß-Prozess-Regression aus, erhalten wir als Erwartungswertvektor $\overline{\boldsymbol{f}_*}$:

$$\overline{\boldsymbol{f}_*} = (-1.01, 0.40). \tag{3.17}$$

Vergleichen wir diesen Erwartungswertvektor (3.17) mit den gemessenen Werten $\boldsymbol{y}_* = (-1, 0)$, erhalten wir einen Gesamtabstand von 0.402.

Diese Methode liefert uns somit eine einfache Möglichkeit um Hyperparameter zu bestimmen. Allerdings ist diese Methode zeitlich sehr aufwändig und liefert, wie wir in Beispiel 3.4 gesehen haben, noch keine optimalen Werte. Deshalb wollen wir uns nun mit anderen Methoden beschäftigen. Diese Methoden benötigen allerdings Optimierungsverfahren, die wir im nächsten Abschntt einführen wollen.

4.2 Optimierungsverfahren

Für die nächsten drei Methoden zur Bestimmung der Hyperparameter (Unterabschnitte 4.3 , 4.5 & 4.4), die in dieser Arbeit vorgestellt werden, benötigen wir Optimierungsverfahren, um entsprechende Werte für die Parameter zu erhalten. Deshalb werden in diesem Abschnitt einige Methoden vorgestellt, die zur Hyperparameterbestimmung verwendet werden können.

CG-Verfahren

Das k*onjugierte Gradienten-Verfahren* oder kurz CG [Nocedal and Wright, 1999] ist eine iterative Methode um zwei Arten von Optimierungsproblemen zu lösen:

- Lineare Optimierungsprobleme mit Hilfe des linearen CG-Verfahrens
- Nichtlineare Optimierungsprobleme mit Hilfe von nichtlinearen CG-Verfahren

Da das nichtlineare CG-Verfahren, das wir für unsere Berechnungen benötigen, auf das lineare CG-Verfahren von *M.R. Hestenes und E.L. Stiefel* [Hestenes and Stiefel, 1952] aufbaut, wollen wir dieses Verfahren genauer betrachten.

Gegeben sei ein lineares Gleichungssystem

$$A\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{b} \in \mathbb{R}^n,$$

wobei A eine symmetrische und positiv definite $n \times n$-Matrix ist. Dieses Problem kann nun äquivalent als Minimierungsproblem umgeschrieben werden

$$\phi(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}.$$

Beide Probleme liefern dieselbe Lösung. Somit können mit Hilfe eines CG-Verfahrens sowohl lineare Systeme gelöst als auch quadratische konvexe Funktionen minimiert werden.

Für das CG-Verfahren benötigen wir außerdem den Gradienten des Minimierungsproblems.

$$\nabla\phi(\boldsymbol{x}) = A\boldsymbol{x} - \boldsymbol{b} = r(\boldsymbol{x}).$$

Das CG-Verfahren erzeugt nun eine Menge von Vektoren $\{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_l\}$, die konjugiert sind.

Definition 3.4 (Konjugierte Vektoren).
Eine Menge von nichtnegativen Vektoren $\{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_l\}$ ist konjugiert bezüglich einer positiv definiten Matrix A, falls

$$\boldsymbol{p}_i^T A\boldsymbol{p}_j = 0 \ \forall i \neq j.$$

Um diese Menge der konjugeierten Vektoren zu erzeugen, wird jeder neue Vektor $\boldsymbol{p}_k$ als Linearkombination der sogenannten größten Abstiegsrichtung $-\nabla\phi(\boldsymbol{x}_k) = -\boldsymbol{r}_k$ und dem vorherigen Vektor $\boldsymbol{p}_{k-1}$ berechnet.

$$\boldsymbol{p}_k = -\boldsymbol{r}_k + \beta_k\boldsymbol{p}_{k-1},$$

wobei der Wert β_k so gewählt wird, dass $\boldsymbol{p}_{k-1}$ und $\boldsymbol{p}_k$ konjugiert bezüglich A sind:

$$\beta_k = \frac{\boldsymbol{r}_k^T A\boldsymbol{p}_{k-1}}{\boldsymbol{p}_{k-1}^T A\boldsymbol{p}_{k-1}}.$$

Die so entstehende Menge an konjugierten Vektoren $\{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_l\}$ wird auch als Menge der konjugierten Richtungen bezeichnet. Mit Hilfe dieser Richtungen und der konjugierten Richtungsmethode kann unsere Funktion $\phi(\cdot)$ in n Schritten erhalten werden. Den dazugehörigen Beweis findet man in [Nocedal and Wright, 1999].

Konjugierte Richtungsmethode

Sei dazu der Startpunkt $\boldsymbol{x}_0 \in \mathbb{R}^n$ und eine Menge an konjugierten Richtungen $\{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_l\}$ gegeben. Dann können wir eine Sequenz $\{\boldsymbol{x}_k\}$ erzeugen durch

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\boldsymbol{p}_k,$$

wobei α_k der eindimensionale Minimierer der Funktion $\phi(\cdot)$ entlang von $\boldsymbol{x}_k + \alpha\boldsymbol{p}_k$ ist. Dieser Minimierer ist explizit gegeben durch

$$\alpha_k = -\frac{\boldsymbol{r}_k^T\boldsymbol{p}_k}{\boldsymbol{p}_k^T A\boldsymbol{p}_k}.$$

α_k wird auch als Schrittlänge bezeichnet.

Durch umschreiben der Berechnungen für α_k und β_k

$$\alpha_k = \frac{\boldsymbol{r}_k^T \boldsymbol{r}_k}{\boldsymbol{p}_k^T A \boldsymbol{p}_k} \qquad \beta_{k+1} = \frac{\boldsymbol{r}_{k+1}^T \boldsymbol{r}_{k+1}}{\boldsymbol{r}_k^T \boldsymbol{r}_k},$$

erhalten wir folgenden CG-Algorithmus.

Algorithmus 3.1 (**CG-Verfahren**)**.**
Gegeben $\boldsymbol{x}_0$;
Setze $\boldsymbol{r}_0 = A\boldsymbol{x}_0 - b$, $\boldsymbol{p}_0 = -\boldsymbol{r}_0$, $k = 0$;
while $\boldsymbol{r}_k \neq 0$:

$$\begin{aligned}
&\alpha_k = \frac{\boldsymbol{r}_k^T \boldsymbol{r}_k}{\boldsymbol{p}_k^T A \boldsymbol{p}_k};\\
&\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}k;\\
&\boldsymbol{r}_{k+1} = \boldsymbol{r}_k + \alpha_k A \boldsymbol{p}_k;\\
&\beta_{k+1} = \frac{\boldsymbol{r}_{k+1}^T \boldsymbol{r}_{k+1}}{\boldsymbol{r}_k^T \boldsymbol{r}_k};\\
&\boldsymbol{p}_{k+1} = -\boldsymbol{r}_{k+1} + \beta_{k+1} \boldsymbol{p}_k;\\
&k = k + 1;
\end{aligned}$$

end(while)

Dieser Algorithmus wurde nun von *R. Fletcher & C.M. Reeves* [Fletcher and Reeves, 1964] erweitert um damit auch nichtlineare Optimierungsprobleme zu lösen:

$$\min_x f(\boldsymbol{x}).$$

Für die sogenannte *Fletcher-Reeves-Methode* werden im obigen Algorithmus 3.1 folgende Veränderungen durchgeführt:

- Für die Berechnung von α_k verwenden wir eine *Line-Search-Methode* (Anhang A), die uns das approximative Minimum der nichtlinearen Funkion f entlang $\boldsymbol{p}_k$ liefert.
- Der einfache Gradient $\nabla\phi = \boldsymbol{r}_k$ wird durch den Gradienten der nichtlinearen Funktion f ersetzt.

Durch diese Änderungen entsteht nun folgender Algorithmus:

Algorithmus 3.2 (**Fletcher-Reeves-CG-Verfahren**)**.**
Gegeben $\boldsymbol{x}_0$;
Berechne $f_0 = f(\boldsymbol{x}_0)$, $\nabla f_0 = \nabla f(\boldsymbol{x}_0)$;
Setze $\boldsymbol{p}_0 = -\nabla f_0$, $k = 0$;
while $\nabla f_0 \neq 0$:

Berechne α_k und setze $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$;
Berechne ∇f_{k+1};

$$\beta_{k+1} = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k};$$
$$\boldsymbol{p}_{k+1} = -\nabla f_{k+1} + \beta k + 1 \boldsymbol{p}_k;$$
$$k = k + 1;$$

end(while)

In `R` kann das CG-Verfahren mit Hilfe der Funktion `optim` und der gewählten Methode `CG` durchgeführt werden.

```
> optim(par, fn, gr, method= "CG")
```

Dabei sind die Argumente der Funktion folgendermaßen definiert:

- `par`: Vektor mit den jeweiligen Startwerten.
- `fn`: Funktion, die minimiert werden soll.
- `gr`: Funktion, die den Gradienten ausgibt. Falls kein Gradient gegeben ist, wird eine Finite-Differenzen-Approximation genutzt um den Gradienten zu erhalten.

Die Toleranz ϵ der Funktion `optim` ist auf 10^{-8} eingestellt. Das bedeutet, dass der Algorithmus stoppt, sobald er den Wert der Funktion $fn(x)$ nicht mehr als um einen Faktor $\epsilon \times (|fn(x)| + \epsilon)$ verringern kann.

BFGS-Verfahren

Die *Broyden-Fletcher-Goldfarb-Shanno-Methode* oder kurz BFGS-Methode [Pfeiffer, 2017] gehört zu den Quasi-Newton-Methoden und wird genutzt um nichtlineare Optimierungsprobleme iterativ zu lösen.

Wie beim CG-Verfahren wollen wir mit einer Menge von Richtungsvektoren $\{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_l\}$ und einer bestimmten Schrittlänge α_k eine nichtlineare Funktion f minimieren. Bei der BFGS-Methode werden allerdings die Richtungen $\{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_l\}$ anders berechnet.

Wir erhalten die Richtung $\boldsymbol{p}_k$ indem wir folgende Newton-Gleichung lösen:

$$\boldsymbol{p}_k = H_k \nabla f(\boldsymbol{x}_k).$$

Mit $\nabla f(\boldsymbol{x}_k)$ wird der Gradient der nichtlinearen Funktion f bezeichnet. H_k ist eine Approximation für die Inverse-Hesse-Matrix $\nabla^2 f(\boldsymbol{x}_k)^{-1}$. Die Approximation H_k muss in jedem Schritt upgedatet werden.

Sei S_m die Menge der symmetrischen Matrizen, dann erhalten wir die Approximation H_{k+1}, wenn wir folgendes Problem lösen:

$$\min_{H \in S_m} \| c\left(H - H_k c^T\right) \|_F^2,$$
$$\text{sodass } H(\underbrace{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}_{s_k}) = \underbrace{\nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)}_{y_k}.$$

Als Lösung dieses Problems erhalten wir für H_{k+1} :

$$H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho_k s_k s_k^T, \quad \rho_k = \frac{1}{y_k^T s_k}.$$

Die Schrittlänge α_k wird wie beim CG-Verfahren mit Hilfe einer *Line-Search-Methode* (Anhang A) bestimmt.

Der Algorithmus der BFGS-Methode [Nocedal and Wright, 1999] kann folgendermaßen formuliert werden:

Algorithmus 3.3 (**BFGS-Algorithmus**)**.**
Gegeben sei der Startpunkt $\boldsymbol{x}_0$, Kovergenztoleranz ϵ, Inverse Hesse-Approximation H_0;
Setze $k = 0$;
while $\|\nabla f(\boldsymbol{x}_k)\| > \epsilon$:
Berechne $\boldsymbol{p}_k = -H_k \nabla f(\boldsymbol{x}_k)$;
Berechne α_k und setzte $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$;
Definiere $s_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ und $y_k = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)$;

$$H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho_k s_k s_k^T, \quad \rho_k = \frac{1}{y_k^T s_k};$$

$k = k + 1$;

end(while)

So wie das CG-Verfahren kann die BFGS-Methode in `R` mit Hilfe der Funktion `optim` durchgeführt werden. Dazu wählen wir als Option die Methode `BFGS`.

```
> optim(par, fn, gr, method= "BFGS")
```

Dabei sind wie zuvor:

- `par`: Vektor mit den jeweiligen Startwerten.
- `fn`: Funktion, die minimiert werden sollte.
- `gr`: Funktion, die den Gradienten ausgibt. Falls kein Gradient gegeben ist, wird eine Finite-Differenzen-Approximation genutzt um den Gradienten zu erhalten.

Da das BFGS-Verfahren in `R` mit Hilfe derselben Methode berechnet wird, wie das CG-Verfahren, liegt die Toleranz ϵ ebenfalls bei 10^{-8}.

Simulated Annealing

Simulated Annealing [Klinz, 2015] zählt zu den Methoden der Metaheuristiken und geht auf die Arbeit von *S. Kirkpatrick et al.* [Kirkpatrick et al., 1983] zurück.

Die Basis für Simulated Annealing bildet allerdings der klassische *Metropolis Algorithmus* [Metropolis et al., 1953] der ursprünglich für den Bereich der Thermodynamik entwickelt wurde. Für den Metropolis-Algorithmus benötigen wir den Begriff der Nachbarschaft.

Definition 3.5 (**Nachbarschaft**)**.**
Sei X eine Menge von zulässigen Lösungen für eine Zielfunktion f. Dann bildet eine Nachbarschaft N die Lösung $x \in X$ auf einen Nachbarn von x ab, falls

$$N(x) \subseteq X.$$

Wir wollen nun mit Hilfe des Metropolis-Algorithmus eine Funktion f minimieren.

Algorithmus 3.4 (**Metropolis-Algorithmus**)**.**
Sei $c \in \mathbb{R}^+$ ein beliebiger Kontrollparameter, X Menge aller zulässigen Lösungen von f;
Wähle eine beliebige zulässige Startlösung $x \in X$;
while Abbruchsbedingung nicht erfüllt:
Wähle zufälligen Nachbarn x^* von x:

$$x^* \in N(x);$$

if $f(x^*) \leq f(x)$:
$x = x^*$;
else
$x = x^*$ mit Wahrscheinlichkeit:

$$\exp\left(\frac{f(x) - f(x^*)}{c}\right); \tag{3.18}$$

end(while)

Die Wahrscheinlichkeit (3.18) gibt die Verschlechterung an, die wir in Kauf nehmen, wenn wir x^* statt x als neues Minimum wählen. Als Abbruchsbedingung wählen wir z.B. die maximale Anzahl an Iterationen, die durchgeführt werden sollen.

Der Unterschied bei Simulated Annealing gegenüber dem Metropolis Algorithmus ist, dass der Kontrollparameter c nicht nur einen fixen Wert annimmt, sondern im Laufe des Algorithmus verkleinert wird. Somit erhalten wir folgenden Simulated Annealing Algorithmus.

Algorithmus 3.5 (**Simulated Annealing**)**.**
Gegeben sei eine Folge von Kontrollparametern $c_0, c_1, c_2, \cdots \in \mathbb{R}^+$ mit

$$\lim_{k \to \infty} c_k = 0;$$

Weiters eine Folge $L_0, L_1, L_2, \cdots \in \mathbb{N}$;

Wähle zulässige Startlösung $x \in X$;
Setze $k = 0$;
while Abbruchsbedingung nicht erfüllt:
Für L_0 bis L_{k-1} wähle einen zufälligen Nachbarn x^* von x:

$$x^* \in N(x);$$

if $f(x^*) \leq f(x)$:
$x = x^*$;
else
$x = x^*$ mit Wahrscheinlichkeit:

$$\exp\left(\frac{f(x) - f(x^*)}{c_k}\right);$$

$k = k + 1$;
end(while)

Als Abbruchsbedingung wählen wir die maximale Anzahl an Iterationen.

In `R` kann der Simulated Annealing Algorithmus mit Hilfe der Funktion `optim` und der Methode `SANN` verwendet werden.

```
> optim(par, fn, method= "SANN")
```

Bei dieser Methode benötigen wir nur die jeweiligen Startwerte `par` und die Funktion `fn`, die minimiert werden sollte. Die Toleranz ϵ liegt wieder bei 10^{-8}.

Da die Berechnung bei Simulated Annealing sehr leicht in einem lokalen Minimum hängen bleibt, wurde von *Tsallis* und *Stariolo* [Tsallis and Stariolo, 1996] eine erweiterte Version des Simulated Annealing Algorithmus entwickelt, der sogenannte *Generalisierte Simulated Annealing* (kurz GenSA) Algorithmus. Deshalb werden wir auch in den späteren Berechnungen nur diesen Algorithmus wählen.

Dieser Algorithmus kann mit Hilfe des `R` -Pakets `GenSA` [Xiang et al., 2013] und der gleichnamigen Funktion `GenSA` durchgeführt werden.

```
> library("GenSA")
> GenSA(par , fr, lower, upper, control=list(maxit))
```

Dabei sind die Argumente der Funktion folgendermaßen definiert:

- `par`: Vektor mit den jeweiligen Startwerten.
- `fn`: Funktion, die minimiert werden sollte.
- `lower`: Vektor der gleichen Länge wie `par`. Gibt die untere Grenze der Minimierer an.
- `upper`: Vektor der gleichen Länge wie `par`. Gibt die obere Grenze der Minimierer an.
- `maxit`: Gibt die maximale Anzahl an Iterationen an.

Die Toleranz ϵ der `R`-Funktion `GenSA` liegt bei 10^{-7}.

Differential Evolution

Differential Evolution oder kurz DE [Fleetwood, 2014] gehört zu den evolutionären Algorithmen und wurde von *R. Storn und K. Price* [Storn and K.Price, 1997] entwickelt. Bei evolutionären Methoden wird in jeder Generierung eine Menge an Parametervektoren zu einer Menge von neuen Parametervektoren transformiert. Dies geschieht in einem sogenannten Mutations- und Rekombinierungsschritt. Am Ende werden jene Parametervektoren behalten, welche die gegebene Zielfunktion am Besten minimieren. Dies geschieht im Selektionsschritt.

Für die Minimierung einer Funktion f gehen wir davon aus, dass diese Funktion D Parameter besitzt. Weiters sei N die Anzahl an Parametervektoren in einer Population. Jeder Parametervektor besitzt folgende Form:

$$\boldsymbol{x}_{i,G} = [x_{1,i,G}, \ldots, x_{D,i,G}]\,,\ i = 1, \ldots, N.$$

Mit G bezeichnet man die sogenannte Generatorennummer. Sie gibt an wie oft eine neue Menge an Parametervektoren generiert wird.

Für die Bestimmung der Initialisierungsvektoren wählen wir für jeden Parameter eine Ober- und eine Untergrenze:

$$\boldsymbol{x}_j^L \leq \boldsymbol{x}_{j,i,1} \leq \boldsymbol{x}_j^U\,.$$

Aus diesem Intervall $\left[\boldsymbol{x}_j^L, \boldsymbol{x}_j^U\right]$ wählen wir zufällig N Initialisierungsparameter. Für jeden dieser N Parameter werden solange folgende Schritte durchgeführt bis eine vorher bestimmte Anzahl an Generierungsschritten erreicht wurde.

- **Mutation**
  Der Mutationsschritt dient dazu den Suchraum zu erweitern. Dazu werden für einen Parametervektor $\boldsymbol{x}_{i,G}$ drei weitere Vektoren $\boldsymbol{x}_{r_1,G}$, $\boldsymbol{x}_{r_2,G}$ und $\boldsymbol{x}_{r_3,G}$ zufällig aus der Menge der Parameter ausgewählt. Dabei müssen sich die vier Parameter in ihren Indizes i, r_1, r_2, r_3 unterscheiden.
  Mit Hilfe der drei gewählten Vektoren $\boldsymbol{x}_{r_1,G}$, $\boldsymbol{x}_{r_2,G}$ und $\boldsymbol{x}_{r_3,G}$ wird der sogenannte Donor-Vektor (Gebervektor) $\boldsymbol{v}_{i,G+1}$ folgendermaßen berechnet:

  $$\boldsymbol{v}_{i,G+1} = \boldsymbol{x}_{r_1,G} + F(\boldsymbol{x}_{r_2,G} - \boldsymbol{x}_{r_3,G}),\ F \in [0, 2]\,.$$

- **Rekombinierung**
  Im Rekombinierungsschritt wird der sogenannte Trail-Vektor (Folgevektor) $\boldsymbol{u}_{j,i,G+1}$ bestimmt. Dabei entspricht der Trail-Vektor dem vorher bestimmten Donor-Vektor mit einer bestimmten Wahrscheinlichkeit CR. Andernfalls wird der Trail-Vektor gleich dem Vektor $\boldsymbol{x}_{j,i,G}$ gesetzt:

  $$\boldsymbol{u}_{j,i,G+1} = \begin{cases} \boldsymbol{v}_{j,i,G+1} & \text{falls } rand_{j,i} \leq CR \text{ oder } j = I_{rand}, \\ \boldsymbol{x}_{j,i,G} & \text{falls } rand_{j,i} > CR \text{ und } j \neq I_{rand} \end{cases} \quad i = 1, \ldots, N;\ j = 1, \ldots, D.$$

  $rand_{j,i}$ ist eine gleichverteilte Zufallszahl zwischen 0 und 1, und I_{rand} eine zufällige ganze Zahl zwischen 1 und D. I_{rand} verhindert, dass $\boldsymbol{v}_{i,G+1} = \boldsymbol{x}_{i,G}$.

- **Selektion**
  Im letzten Schritt, dem Selektionsschritt, werden die Funktionswerte von $\boldsymbol{x}_{i,G}$ mit den Funktionswerten vom Trail-Vektor $\boldsymbol{u}_{j,i,G+1}$ verglichen. Gilt nun

  $$f(\boldsymbol{u}_{j,i,G+1}) \leq f(\boldsymbol{x}_{i,G}),$$

  dann kommt $\boldsymbol{u}_{j,i,G+1}$ in die neue Menge der Parametervektoren. Andernfalls bleibt $\boldsymbol{x}_{i,G}$ in der neuen Menge enthalten.

In `R` kann Differential Evolution mit Hilfe des `R` -Pakets `DEoptim` [Mullen et al., 2011] und der gleichnamigen Funktion `DEoptim` berechnet werden.

```
> library("DEoptim")
> DEoptim(fr, lower, upper, control=list(itermax))
```

Dabei sind die Argumente der Funktion folgendermaßen definiert:

- `fn`: Funktion, die minimiert werden sollte.
- `lower`: Vektor der gleichen Länge wie `par`. Gibt die untere Grenze der Minimierer an.
- `upper`: Vektor der gleichen Länge wie `par`. Gibt die obere Grenze der Minimierer an.
- `itermax`: Gibt die maximale Anzahl an Iterationen an.

Die `R`-Funktion `DEoptim` ist somit das einzige Optimierungsverfahren, das wir in dieser Arbeit verwenden, welches keine Startparameter benötigt. Die Toleranz ϵ der `R`-Funktion `DEoptim` liegt bei 10^{-6}.

Beispiel 3.5 (Vergleich der Optimierungsmethoden)**.**

Wir wollen die Minimierer für die folgende *Rosenbrock Funktion* [Rosenbrock, 1960] erhalten:

$$f(\boldsymbol{x}) = f(x_1, x_2) = 100(x_2 - x_1 \times x_1)^2 + (1 - x_1)^2. \tag{3.19}$$

Wie wir leicht durch einsetzen in die Gleichung (3.19) erkennen, liegt das globale Minimum dieser Funktion bei $(1, 1)$:

$$f(1, 1) = 0.$$

Dieses globale Minimum ist ebenfalls in Abb. 3.7 eingezeichnet.

Wir wollen nun herausfinden, wie genau die vorgestellten Optimierungsverfahren, an diese Lösung herankommen, wenn wir die Rosenbrock Funktion (3.19) minimieren. Dazu implementieren wir zunächst die Funktion, sowie ihren Gradienten in `R`.

```
> fr <- function(x) {
+ x1 <- x[1]
+ x2 <- x[2]
+ 100 * (x2 - x1 * x1)^2 + (1 - x1)^2}

> grr <- function(x) {
+ x1 <- x[1]
+ x2 <- x[2]
+ c(-400 * x1 * (x2 - x1 * x1) - 2 * (1 - x1),
+   200 *  (x2 - x1 * x1))}
```

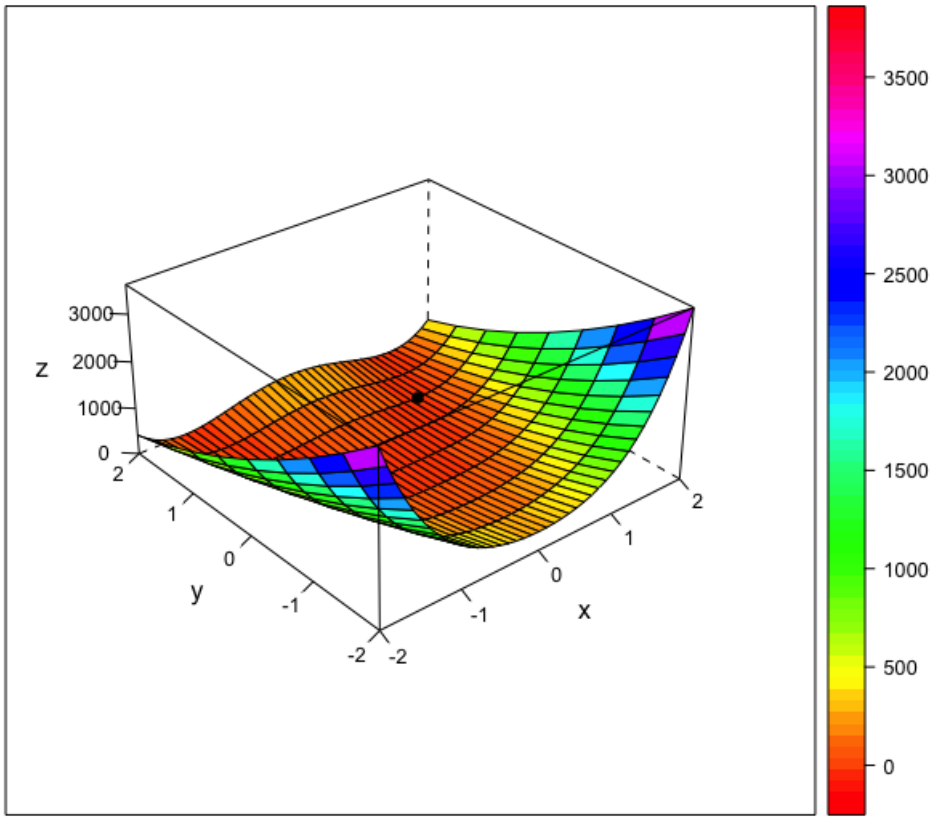

Abbildung 3.7: Rosenbrock-Funktion mit globalen Minimum bei $(1, 1)$.

Im nächsten Schritt wenden wir die obigen Optimierungsverfahren mit verschiedenen Startwerten an. Als Startwerte $\boldsymbol{st}$ wählen wir folgende Vektoren:

$$\boldsymbol{st}_1 = (-1.2, 1), \ \boldsymbol{st}_2 = (0.1, 0.1), \ \boldsymbol{st}_3 = (2, 2).$$

Für das GenSA, sowie das DE-Verfahren wählen wir den Vektor $(0, 0)$ als untere Grenze und $(10, 10)$ als obere Grenze. Die Ergebnisse der Optimierungsverfahren sind in den Tabellen 3.2 – 3.4 aufgelistet.

	CG ohne Gradient	**CG mit Gradient**	**BFGS ohne Gradient**	**BFGS mit Gradient**	**SANN**	**GenSA**	**DE**
x_1	-0.765	-0.765	1.000	1.000	1.010	1.000	1.000
x_2	0.593	0.593	1.000	1.000	1.021	1.000	1.000
$f(\boldsymbol{x})$	3.121	3.107	0.000	0.000	0.000	0.000	0.000

Tabelle 3.2: Optimierung Rosenbrock Funktion mit Startwerten $(-1.2, 1)$.

	CG ohne Gradient	**CG mit Gradient**	**BFGS ohne Gradient**	**BFGS mit Gradient**	**SANN**	**GenSA**	**DE**
x_1	0.724	0.724	1.000	1.000	1.016	1.000	1.000
x_2	0.523	0.523	1.000	1.000	1.034	1.000	1.000
$f(\boldsymbol{x})$	0.076	0.076	0.000	0.000	0.001	0.000	0.000

Tabelle 3.3: Optimierung Rosenbrock Funktion mit Startwerten $(0.1, 0.1)$.

Wie wir anhand der Tabellen 3.2 – 3.4 erkennen, liefern die Optimierungsverfahren BFGS, GenSA und DE immer das globale Minimum an der Stelle $\boldsymbol{x} = (1, 1)$. Hingegen erreichen die beiden CG-Verfahren sowie das SANN-Verfahren nie das globale Minimum an der Stelle $\boldsymbol{x} = (1, 1)$. Vor allem das CG-Verfahren erreicht Werte, die weit vom globalen Minimum entfernt liegen. Der Grund dafür ist, dass das CG-Verfahren anfälliger dafür ist in einem lokalen Minimum hängen zu bleiben als alle anderen Verfahren. Des Weiteren erkennen wir, dass das CG-Verfahren sehr stark von den gewählten Startwerten abhängt. Je nach Wahl liefert das CG-Verfahren Minima deren Funktionswerte nahe 0 liegen oder aber wie bei den Startwerten $\boldsymbol{st}_1 = (1.2, 1)$ deutlich davon abweichen. (Vgl. Tabelle 3.2 und Abb. 3.8)

Weiteres erkennen wir anhand der Tabellen und Grafiken, dass es keinen Unterschied macht, ob wir beim CG-Verfahren oder BFGS-Verfahren den Gradienten in die Minimierung miteinbeziehen oder nicht. Diese Tatsache nutzten wir, indem wir bei der folgenden Hyperparameterbestimmung, den Gradienten nicht berechnen. Außerdem erkennen wir einen Unterschied zwischen den beiden Simulated Annealing Verfahren. Das gewöhnliche Simulated Annealing Verfahren (SANN) bleibt leichter in einem lokalen Minimum hängen als das Generalisierte Simulated Annealing Verfahren (GenSA). Deshalb werden wir im Laufe der weiteren Arbeit nur mehr das GenSA-Verfahren als Simulated Annealing Methode verwenden.

	CG ohne Gradient	**CG mit Gradient**	**BFGS ohne Gradient**	**BFGS mit Gradient**	**SANN**	**GenSA**	**DE**
x_1	1.437	1.437	1.000	1.000	1.012	1.000	1.000
x_2	2.067	2.067	1.000	1.000	1.022	1.000	1.000
$f(\boldsymbol{x})$	0.191	0.191	0.000	0.000	0.000	0.000	0.000

Tabelle 3.4: Optimierung Rosenbrock Funktion mit Startwerten $(2, 2)$.

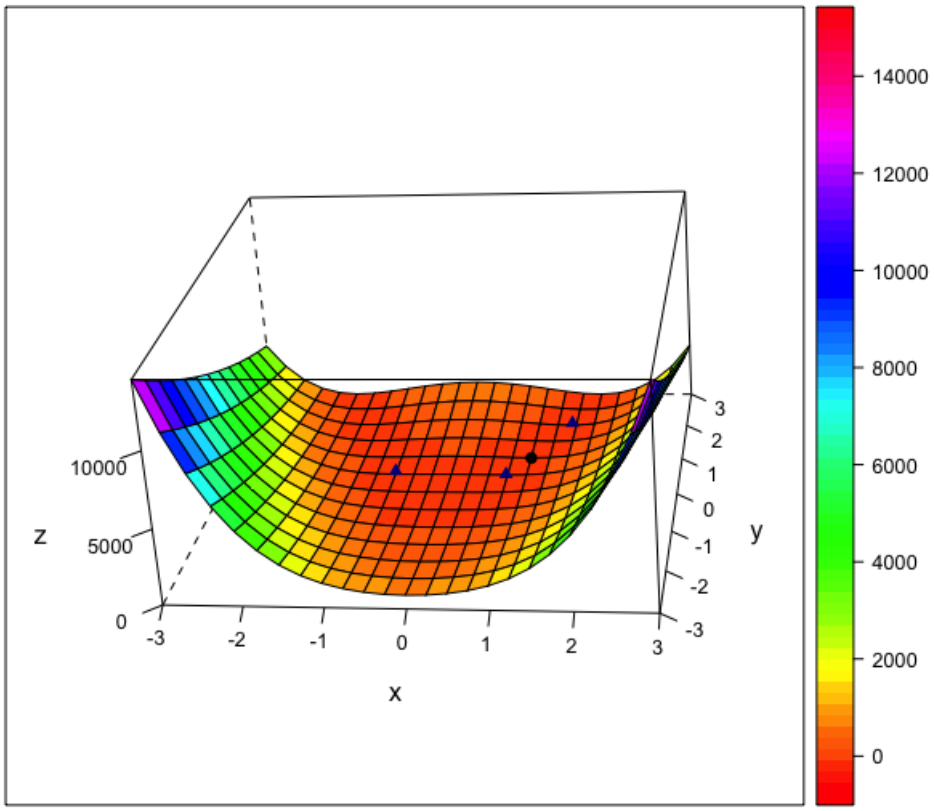

Abbildung 3.8: Rosenbrock-Funktion mit den eingezeichneten lokalen Minima, die wir durch das CG-Verfahren erhalten.

4.3 Maximierung der marginalen log-Likelihood-Funktion

Die folgende Idee zur Maximierung der marginalen log-Likelihood-Funktion basiert auf der Arbeit von *C. E. Rasmussen & C. K. I. Williams* [Rasmussen and Williams, 2006].

Im Unterabschnitt 3.3 führten wir die marginale log-Likelihood-Funktion (3.13) für die Gauß-Prozess-Regression ein:

$$\log p(\boldsymbol{y}|\boldsymbol{x},\theta) = -\tfrac{1}{2}\boldsymbol{y}^T\left[k(\boldsymbol{x},\boldsymbol{x}) + \sigma_n^2 I\right]^{-1} - \tfrac{1}{2}\log|\left[k(\boldsymbol{x},\boldsymbol{x}) + \sigma_n^2 I\right]^{-1}| - \tfrac{n}{2}\log 2\pi.$$

Die marginale log-Likelihood-Funktion besteht also aus drei Termen, die wir folgendermaßen interpretieren können:

Term 1:

$$-\frac{\boldsymbol{y}^T\left[k(\boldsymbol{x},\boldsymbol{x}) + \sigma_n^2 I\right]^{-1}}{2}.$$

Der sogenannte Data-Fit-Term ist der einzige Term, der den Funktionswert der Trainingsdaten benötigt. Er gibt ein Maß an wie die Hyperparameter an die Daten angepasst sind.

Term 2:

$$\frac{n\log 2\pi}{2}.$$

Die Normalisierungskonstante spielt bei der Interpretation der marginalen log-Likelihood-Funktion keine Rolle.

Term 3:

$$\frac{\log|\left[k(\boldsymbol{x},\boldsymbol{x}) + \sigma_n^2 I\right]^{-1}|}{2}.$$

Der Komplexitätsstrafterm ist nur abhängig von der Kovarianz-Funktion. Dieser Term gibt das Maß zwischen ausreichender und zu hoher Komplexität des Modells an.

Um die Werte der Hyperparameter bestimmen zu können, suchen wir nach den partiellen Ableitungen der marginalen log-Likelihood bezüglich der Hyperparameter $\boldsymbol{\theta}$.

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\log p(\boldsymbol{y}|\boldsymbol{x},\theta) &= \frac{1}{2}\boldsymbol{y}^T k(\boldsymbol{x},\boldsymbol{x})^{-1}\frac{dk}{d\theta_j}k(\boldsymbol{x},\boldsymbol{x})^{-1}\boldsymbol{y} - \frac{1}{2}tr\left(k(\boldsymbol{x},\boldsymbol{x})\frac{dk}{d\theta_j}\right)\\ &= \frac{1}{2}tr\left(\left(\alpha\alpha^T - k(\boldsymbol{x},\boldsymbol{x})\right)\frac{dk}{d\theta_j}\right),\ \alpha = k(\boldsymbol{x},\boldsymbol{x})^{-1}\boldsymbol{y}.\end{aligned}$$

Da die Berechnung der partiellen Ableitungen komplex ist, nutzen wir die in Unterabschnitt 4.2 vorgestellten Optimierungsverfahren. Diese Methoden bestimmen das Maximum der log-Likelihood-Funktion, und dadurch erhalten wir die Werte der geeigneten Hyperparameter.

Wichtig zu erwähnen ist, dass die vorgestellten Optimierungsverfahren die gegebene Funktionen minimieren. Somit muss für die Maximierung die negative marginale log-Likelihood-Funktion minimiert werden:

$$-\log p(\boldsymbol{y}|\boldsymbol{x},\theta) = \tfrac{1}{2}\boldsymbol{y}^T\left[k(\boldsymbol{x},\boldsymbol{x}) + \sigma_n^2 I\right]^{-1} + \tfrac{1}{2}\log|\left[k(\boldsymbol{x},\boldsymbol{x}) + \sigma_n^2 I\right]^{-1}| + \tfrac{n}{2}\log 2\pi.$$

	l	σ_f^2	σ_n^2	$\hat{y}_{*1}$	$\hat{y}_{*2}$	L_1	L_2	**negative log-Likelihood**
BFGS	7.140	9.369	0.052	−1.237	0.234	0.471	0.111	**5.814**
CG	4.475	3.287	0.068	−1.324	0.208	0.532	0.148	6.063
GenSA	5.971	6.930	0.100	−1.249	0.214	**0.464**	**0.108**	6.003
DE	5.971	6.925	0.100	−1.249	0.214	**0.464**	**0.108**	6.003

Tabelle 3.5: Maximierung der log-Likelihood-Funktion mit Startwerten $(1, 0.880, 0.180)$.

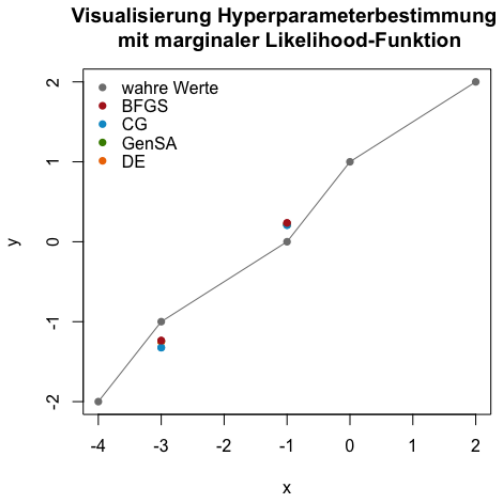

Abbildung 3.9: Maximierung der log-Likelihood-Funktion mit Startwerten $(1, 0.880, 0.180)$.

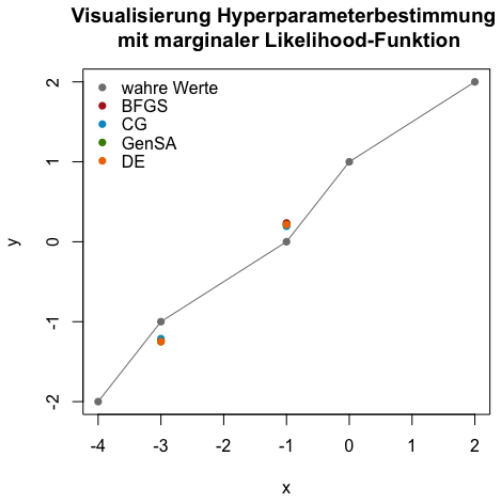

Abbildung 3.10: Maximierung der log-Likelihood-Funktion mit Startwerten $(10, 10, 10)$.

	l	σ_f^2	σ_n^2	$\hat{y}_{*1}$	$\hat{y}_{*2}$	L_1	L_2	**negative log-Likelihood**
BFGS	7.214	9.624	0.052	−1.236	0.234	0.470	0.111	**5.814**
CG	9.222	8.494	0.047	−1.212	0.194	**0.406**	**0.083**	6.162
GenSA	5.971	6.930	0.100	−1.249	0.214	0.464	0.108	6.003
DE	5.971	6.925	0.100	−1.249	0.214	0.464	0.108	6.003

Tabelle 3.6: Maximierung der log-Likelihood-Funktion mit Startwerten $(10, 10, 10)$.

Beispiel 3.6.
Gegeben seien die selben Trainings- und Testdaten wie in Beispiel 3.4. Weiters benutzen wir wieder die quadratische Exponentialfunktion als Kovarianz-Funktion (3.12).

```
> log_likeli <- function(para) {
+ K <- cov_funct_matrix_algo(x,x,para[1],para[2])
+ noise <- para[3]*diag(nrow(K))
+ K_noise <- K+noise
+
+ -(-(1/2)*t(y)%*%solve( K_noise)%*%y-(1/2)*log(det(K_noise))-
+ (length(y)/2)%*%log(2*pi))}
```

Mit Hilfe der verschiedenen Optimierungsverfahren (Abschnitt 4.2) wollen wir die Werte der Hyperparameter $\boldsymbol{\theta} = (l, \sigma_f^2, \sigma_n^2)$ bestimmen. Die Hyperparameter sind dann geeignet für die anschließende Gauß-Prozess-Regression, falls der geschätzte Erwartungswertvektor $\overline{\boldsymbol{f_*}}$ möglichst nahe an den gemessenen Funktionsvektor $\boldsymbol{y_*} = (-1, 0)$ herankommt.

In Tabelle 3.5 und Abb. 3.9 werden die Ergebnisse der Hyperparameterbestimmung aufgelistet, wenn wir $(1, 0.880, 0.180)$ als Startwerte für $\boldsymbol{\theta}$ wählen.

Wie wir anhand Tabelle 3.5 erkennen, wird unsere log-Likelihood-Funktion am Besten mit Hilfe der BFGS-Methode maximiert, da ihr negativer log-Likelihood-Funktionswert am kleinsten ist. Für die Minimierung des Gesamtabstandes mit der L_1- und der L_2-Norm liefern die BFGS-, GenSA- und DE-Verfahren die besten Ergebnisse. Die CG-Methode liefert hingegen das schlechteste Ergebnis. Der Grund dafür liegt darin, dass das CG-Verfahren anfälliger dafür ist, in einem lokalen Minimum hängen zu bleiben, als die anderen Methoden.

Wählen wir allerdings andere Startwerte $(10, 10, 10)$ erkennen wir in Tabelle 3.6 und Abb. 3.10, dass mit Hilfe des CG-Verfahrens der geringste L_1-Abstand geliefert wird. Hingegen ist der negative log-Likelihood-Funktionswert mit der CG-Methode am höchsten.

Da wir in unserer praktischen Umsetzung in Kapitel 4 den L_1- Abstand minimieren wollen, und dieser Abstand im Vergleich zu den nachfolgenden Methoden (vgl. Tabelle 3.7 – 3.10) noch recht groß ist, wird die Maximierung der log-Likelihood in Kapitel 4 nicht verwendet.

4.4 Minimierung Gesamtabstand mit Hilfe der L_2-Norm

Wie zuvor erwähnt, wollen wir in unserer späteren praktischen Umsetzung in Kapitel 4, den Gesamtabstand zwischen den interpolierten und gemessenen Punkten minimieren. Eine Möglichkeit ist nun diesen Abstand mit Hilfe der L_2-Norm und der in Unterabschnitt 4.2 vorgestellten Optimierungsverfahren zu minimieren.

Dazu wählen wir einen Teil der Trainingsdaten als Testdaten $\boldsymbol{x_*}$ aus. Die restlichen Trainingsdaten bleiben Trainingsdaten $\boldsymbol{x}$. Anschließend erstellen wir eine Funktion, die uns den Gesamtabstand zwischen extrapolierten und gemessenen Punkten mit Hilfe der L_2-Norm liefert.

Sei nun $\overline{\boldsymbol{f}_*}$ jener Wert des A-posteriori Erwartungswertes den wir erhalten, falls wir die Gauß-Prozess-Regression mit einer bestimmten Einstellung der Hyperparameter durchgeführt haben. $\boldsymbol{y}_*$ ist der Vektor der eigentlich gemessenen Funktionswerte, der vorher bestimmten Testdaten. Dann gibt folgende Funktion den Gesamtabstand zwischen interpolierten und gemessenen Punkten an:

$$L_2 = \sum \left(\boldsymbol{y}_{*i} - \overline{\boldsymbol{f}_{*i}}\right)^2. \tag{3.20}$$

Diese Funktion wird nun mit Hilfe von in Abschnitt 4.2 beschriebenen Optimierungsverfahren minimiert. Als Ergebnis erhalten wir die gesuchten Hyperparameter.

Beispiel 3.7.
Gegeben seien die selben Trainings- und Testdaten wie in Beispiel 3.4. Als Kovarianz-Funktion wurde erneut die quadratische Exponentialfunktion (3.12) gewählt.

```
> min_abs <- function(para)
+ {
+ K <- cov_funct_matrix_algo(x,x,para[1],para[2])
+ noise <- para[3]*diag(nrow(K))
+ K_noise <- K+noise
+
+ if (det( K_noise) == 0){
+  Inf } else{
+  K.xstar<- cov_funct_matrix_algo(x.star,x,para[1],para[2])
+
+  f.star.bar <-K.xstar%*%solve(K_noise)%*%y
+
+  abstand<- (y.star-f.star.bar)
+  sum(abstand^2)} }
```

Mit Hilfe der verschiedenen Optimierungsverfahren aus Unterabschnitt 4.2 werden nun die Werte der Hyperparameter $\boldsymbol{\theta} = (l, \sigma_f^2, \sigma_n^2)$ bestimmt. Wie zuvor suchen wir jene Hyperparameter, mit denen die Gauß-Prozess-Regression ähnliche Erwartungswertvektoren $\overline{\boldsymbol{f}_*}$ liefert wie der gemessene Funktionsvektor $\boldsymbol{y}_* = (-1, 0)$.

In Tabelle 3.7 und Abb. 3.11 werden die Ergebnisse der Hyperparameterbestimmung mit Startwerten $(1, 0.880, 0.180)$ aufgelistet. Auch bei diesem Verfahren liefern die BFGS-, GenSA- und DE-Methoden die geringsten L_1- und L_2-Abstände. Die Werte des CG-Verfahrens liegen höher, wenn wir die Gesamtabstände L_1 und L_2 vergleichen. Hingegen liefert das CG-Verfahren den deutlich kleinsten negativen log-Likelihood-Funktionswert. Vergleichen wir die Ergebnisse der Tabelle 3.7 mit den Ergebnissen, die wir erhalten, wenn wir die Methode der Maximierung der marginalen log-Likelihood-Funktion verwenden (Tabelle 3.5), erkennen wir deutlich, dass sich die Werte von L_1 und L_2 deutlich verringert haben.

Dafür sind die negativen log-Likelihood-Funktionswerte deutlich höher als zuvor. Somit liefert die Minimierung des Gesamtabstandes mit Hilfe der L_2-Norm schon eine gute Möglichkeit um auch einen geringen L_1-Abstand zu erhalten.

	l	σ_f^2	σ_n^2	$\hat{y}_{*1}$	$\hat{y}_{*2}$	L_1	L_2	negative log-Likelihood
BFGS	18.971	1.743	0.017	−1.005	0.148	**0.154**	0.022	41.997
CG	2.632	0.919	0.413	−1.019	0.207	0.225	0.043	**7.731**
GenSA	17.966	100.000	0.989	−1.020	0.147	0.167	0.022	9.420
DE	24.859	52.499	0.272	−1.019	0.144	0.163	**0.021**	8.322

Tabelle 3.7: Minimierung Gesamtabstand mit L_2-Norm mit Startwerten $(1, 0.880, 0.180)$.

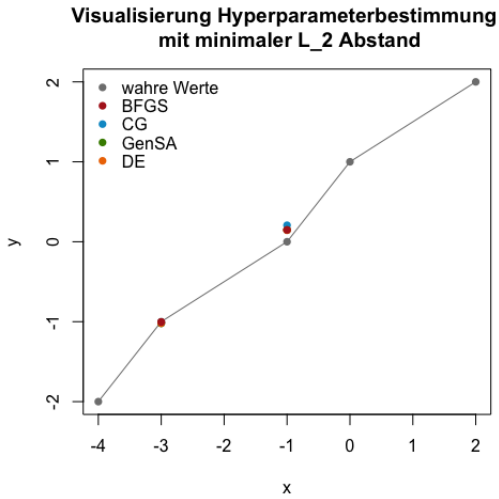

Abbildung 3.11: Minimierung Gesamtabstand mit L_2-Norm mit Startwerten $(1, 0.880, 0.180)$.

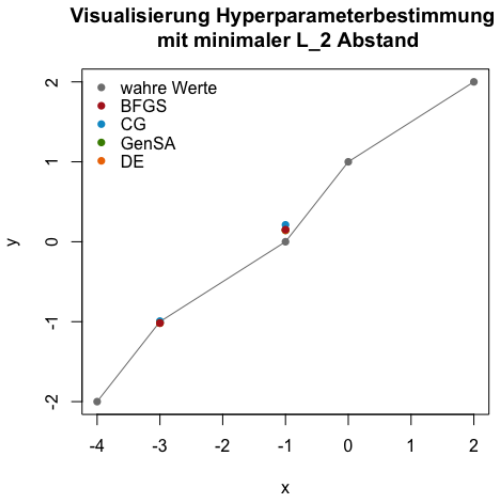

Abbildung 3.12: Minimierung Gesamtabstand mit L_2-Norm mit Startwerten $(10, 10, 10)$.

	l	σ_f^2	σ_n^2	$\hat{y}_{*1}$	$\hat{y}_{*2}$	L_1	L_2	negative log-Likelihood
BFGS	−14.588	15.717	0.241	−1.017	0.151	0.168	0.023	**7.885**
CG	2.258	12.388	6.878	−0.993	0.212	0.219	0.045	11.580
GenSA	48.026	71.206	0.100	−1.016	0.142	**0.158**	0.023	10.782
DE	24.859	52.499	0.272	−1.019	0.144	0.163	**0.021**	8.322

Tabelle 3.8: Minimierung Gesamtabstand mit L_2-Norm mit Startwerten $(10, 10, 10)$.

Verwenden wir $(10, 10, 10)$ als Startwerte, erkennen wir in Tabelle 3.8 kaum eine Veränderung in den L_1- und L_2-Abständen. Lediglich die Werte der negativen log-Likelihood-Funktionswerte haben sich verändert.

4.5 Minimierung Gesamtabstand mit Hilfe der L_1-Norm

Eine weitere Möglichkeit um einen geringen Gesamtabstand zwischen interpolierten und gemessenen Punkten zu erhalten, ist die Minimierung des Gesamtabstandes mit Hilfe der L_1-Norm.

Wie schon bei der vorherigen Abstandsminimierung (Unterabschnitt 4.4), wählen wir einen Teil der Trainingsdaten als Testdaten $\boldsymbol{x_*}$ aus. Die restlichen Trainingsdaten bleiben Trainingsdaten $\boldsymbol{x}$. Sei weiters $\overline{\boldsymbol{f_*}}$ wieder jener Wert des A-posteriori Erwartungswertes den wir erhalten, falls wir die Gauß-Prozess-Regression mit einer bestimmten Einstellung der Hyperparameter durchgeführt haben. $\boldsymbol{y}_*$ ist der Vektor der eigentlich gemessenen Funktionswerte, der vorher bestimmten Testdaten. Dann gibt folgende Funktion den Gesamtabstand zwischen interpolierten und gemessenen Punkten mit Hilfe der L_1-Norm an:

$$L_1 = \sum |\boldsymbol{y}_{*i} - \overline{\boldsymbol{f}_{*i}}|. \tag{3.21}$$

Diese Funktion wird nun mit Hilfe von in Unterabschnitt 4.2 beschriebenen Optimierungsverfahren minimiert. Als Ergebnis erhalten wir die gesuchten Hyperparameter.

Beispiel 3.8.
Gegeben seien die selben Trainings- und Testdaten wie in Beispiel 3.4. Als Kovarianz-Funktion wurde erneut die quadratische Exponentialfunktion (3.12) gewählt.

```
> min_abs <- function(para)
+ {
+ K <- cov_funct_matrix_algo(x,x,para[1],para[2])
+ noise <- para[3]*diag(nrow(K))
+ K_noise <- K+noise
+
+ if (det( K_noise) == 0){
+  Inf } else{
+  K.xstar<- cov_funct_matrix_algo(x.star,x,para[1],para[2])
+
+  f.star.bar <-K.xstar%*%solve(K_noise)%*%y
+
+  abstand<- abs(y.star-f.star.bar)
+  sum(abstand)} }
```

Mit Hilfe der verschiedenen Optimierungsverfahren aus Unterabschnitt 4.2 werden nun die Werte der Hyperparameter $\boldsymbol{\theta} = (l, \sigma_f^2, \sigma_n^2)$ bestimmt. Wie zuvor suchen wir jene Hyperparameter, mit denen die Gauß-Prozess-Regression ähnliche Erwartungswertvektoren $\overline{\boldsymbol{f_*}}$ liefert wie der gemessene Funktionsvektor $\boldsymbol{y_*} = (-1, 0)$.

In Tabelle 3.9 und Abb. 3.13 werden die Ergebnisse der Hyperparameterbestimmung mit Startwerten $(1, 0.880, 0.180)$ aufgelistet.

Wir erhalten vor allem mit Hilfe des DE- und GenSA-Verfahrens, die geringsten L_1-Abstände. Hingegen, liefern die Optimierungsverfahren CG-, BFGS- und DE die geringsten negativen log-Likelihood-Funktionswerte, sowie geringe L_2-Abstände. Setzen wir die Startwerte auf $(10, 10, 10)$ erkennen wir in Tabelle 3.10 eine Verbesserung in den L_1-Abständen beim BFGS-, CG, sowie GenSA-Verfahren. Das DE-Verfahren ändert sich nicht, da es von keinem Startparameter abhängig ist.

Des Weiteren erkennen wir sowohl in Tabelle 3.9 als auch Tabelle 3.10, dass die Minimierung des L_1-Abstandes ähnliche Funktionswerte $\hat{\boldsymbol{y}}_*$ liefert, wie die Methode der Minimierung des Gesamtabstandes mit Hilfe der L_2-Norm (Unterabschnitt 4.4). Trotzdem werden wir in der praktischen Umsetzung in Kapitel 4 die Methode der Minimierung des Gesamtabstandes mit Hilfe der L_1-Norm verwenden, da wir an einem geringen L_1-Abstand interessiert sind. Allgemein erkennen wir, dass bei allen Methoden das DE-Verfahren die kleinsten L_1-Abstände liefert.

5 Plausible Wahl der Erwartungswert-Funktion

In der Praxis wird oft angenommen, dass die Erwartungswert-Funktion des A-priori Gauß-Prozesses 0 ist. Diese Annahme verursacht keine drastischen Einschränkungen für den A-posteriori Erwartungswertvektor, da der A-posteriori Erwartungswertvektor einen anderen Wert als 0 annehmen kann.

Trotzdem kann die Bestimmung von plausiblen A-priori Erwartungswert-Funktionen genutzt werden um die Ergebnisse des A-posteriori Erwartungswertvektors zu verbessern. Somit wollen wir in diesem Abschnitt vorstellen, wie wir eine Erwartungswert-Funktion in die Gauß-Prozess-Regression miteinbeziehen können.

5.1 Simple Kriging

Die Einbeziehung einer fixen Erwartungswert-Funktion $\boldsymbol{m}(\boldsymbol{x})$ in die Gauß-Prozess-Regression wird als Simple Kriging [Rasmussen and Williams, 2006] bezeichnet. Diese fixe Erwartungswert-Funktion kann z.B. ein lineares Modell sein, das mit Hilfe einer multiplen linearen Regression erstellt wurde:

$$\boldsymbol{m}(\boldsymbol{x}) = \beta_0 + \boldsymbol{x}_1\beta_1 + \cdots + \boldsymbol{x}_m\beta_m.$$

Bezeichne mit $\boldsymbol{x}_*$ die Testdaten und mit $\boldsymbol{x}$ die Trainingsdaten. Die Verteilung des A-priori Gauß-Prozesses ist wie folgt gegeben:

$$f(\boldsymbol{x}) \sim N\left(\boldsymbol{m}(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}_*)\right).$$

Dadurch ändert sich die Berechnung des A-posteriori Erwartungswerte $\overline{\boldsymbol{f}_*}$ folgendermaßen:

$$\overline{\boldsymbol{f}_*} = \boldsymbol{m}(\boldsymbol{x}_*) + k(\boldsymbol{x}_*, \boldsymbol{x})\left(k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 I\right)^{-1}(\boldsymbol{y} - \boldsymbol{m}(\boldsymbol{x})). \tag{3.22}$$

	l	σ_f^2	σ_n^2	$\hat{y}_{*1}$	$\hat{y}_{*2}$	L_1	L_2	negative log-Likelihood
BFGS	1.000	0.880	0.180	−1.000	0.395	0.395	0.159	8.592
CG	1.000	0.880	0.180	−1.000	0.395	0.395	0.159	8.592
GenSA	11.375	48.950	1.341	−1.000	0.159	0.159	0.395	9.502
DE	28.415	73.779	0.321	−1.000	0.146	**0.146**	**0.021**	**8.557**

Tabelle 3.9: Minimierung Gesamtabstand mit L_1-Norm mit Startwerten $(1, 0.880, 0.180)$.

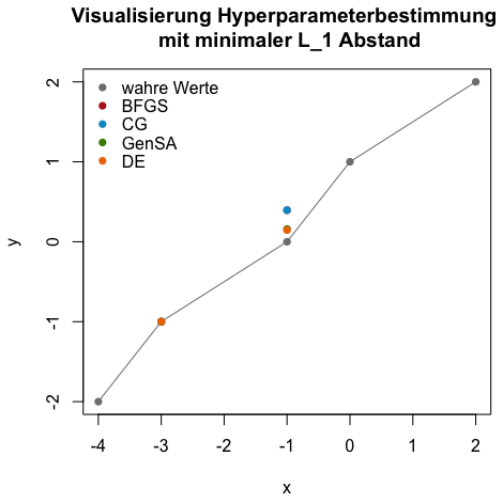

Abbildung 3.13: Minimierung Gesamtabstand mit L_1-Norm mit Startwerten $(1, 0.880, 0.180)$.

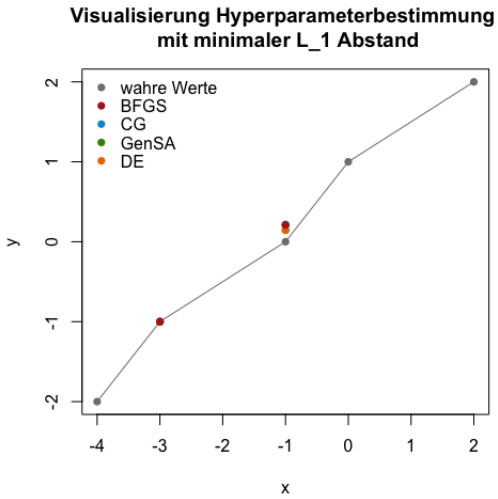

Abbildung 3.14: Minimierung Gesamtabstand mit L_1-Norm mit Startwerten $(10, 10, 10)$.

	l	σ_f^2	σ_n^2	$\hat{y}_{*1}$	$\hat{y}_{*2}$	L_1	L_2	negative log-Likelihood
BFGS	2.498	12.755	6.376	−1.000	0.209	0.209	0.044	11.446
CG	2.032	12.278	7.071	−1.000	0.217	0.217	0.047	11.668
GenSA	35.308	53.913	0.152	−1.000	0.145	**0.145**	**0.021**	9.421
DE	28.415	73.779	0.321	−1.000	0.146	0.146	**0.021**	**8.557**

Tabelle 3.10: Minimierung Gesamtabstand mit L_1-Norm mit Startwerten $(10, 10, 10)$.

Die A-posteriori Varianz-Kovarianzmatrix bleibt gleich wie zuvor (3.8):

$$cov(\boldsymbol{f_*}) = k(\boldsymbol{x_*}, \boldsymbol{x_*}) - k(\boldsymbol{x_*}, \boldsymbol{x})[k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 I]^{-1} k(\boldsymbol{x}, \boldsymbol{x_*}).$$

In der praktischen Umsetzung in Kapitel 4 werden wir diese Methode anwenden, um die A-priori Erwartungswert-Funktionen in die Gauß-Prozess-Regression mit einzubeziehen, da wir mit unseren Daten problemlos lineare Modelle erstellen können.

5.2 Universal Kriging

Falls wir keine fixe Erwartungswert-Funktion spezifizieren können, gibt es das sogenannte Universal-Kriging [Rasmussen and Williams, 2006].

Bei dieser Methode bestimmen wir eine Menge an beliebigen fixen Funktionen $\boldsymbol{h}(\boldsymbol{x})$, den sogenannten Basisfunktionen mit Koeffizienten $\boldsymbol{\beta}$. Dann kann unser A-priori Gauß-Prozess mit Erwartungswert-Funktion folgendermaßen geschrieben werden:

$$g(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{h}(\boldsymbol{x})^T \boldsymbol{\beta},\ \ f(\boldsymbol{x}) \sim N(0, k(\boldsymbol{x}, \boldsymbol{x_*})). \tag{3.23}$$

Wir addieren also zu unserem Gauß-Prozess $f(\boldsymbol{x})$ mit A-priori Erwartungswert-Funktion 0 die Menge der fixen Basisfunktionen $\boldsymbol{h}(\boldsymbol{x})$ mit Parameter $\boldsymbol{\beta}$ hinzu. Der Parameter $\boldsymbol{\beta}$ kann einerseits wie die Hyperparameter in Abschnitt 4 bestimmt werden. Andererseits können wir auch annehmen, dass $\boldsymbol{\beta}$ A-priori einer Gaußverteilung entspricht.

Sei nun $\boldsymbol{\beta}$ Gaußverteilt mit Erwartungswertvektor $\boldsymbol{b}$ und Varianz-Kovarianzmatrix B. Dann können wir den obigen Gauß-Prozess (3.23) umschreiben als:

$$g(\boldsymbol{x}) \sim N\left(\boldsymbol{h}(\boldsymbol{x})^T \boldsymbol{b}, k(\boldsymbol{x}, \boldsymbol{x_*}) + \boldsymbol{h}(\boldsymbol{x})^T B \boldsymbol{h}(\boldsymbol{x_*})\right). \tag{3.24}$$

Die Vorhersage für den A-posteriori Erwartungswertvektor und der A-posteriori Varianz-Kovarianzmatrix erhalten wir indem wir die Erwartungswert-Funktion und die Kovarianz-Funktion von (3.24) in die Gleichungen (3.6) des Gauß-Prozesses mit Erwartungswert-Funktion 0 einsetzen. Sei dazu $\boldsymbol{x}$ der Vektor der Trainingsdaten und $\boldsymbol{x_*}$ der Vektor der Testdaten. Weiters sei H die Matrix aller Basisfunktionen $\boldsymbol{h}(\boldsymbol{x})$ ausgewertet an den Trainingsdaten und analog H_* die Matrix der Basisfunktionen ausgewertet an den Testdaten. Außerdem sind folgende Gleichungen definiert:

$$\overline{\boldsymbol{\beta}} = \left(B^{-1} + H\left(k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2\right)^{-1} H^T\right)^{-1} \left(H\left(k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2\right)^{-1} \boldsymbol{y} + B^{-1}\boldsymbol{b}\right),$$
$$R = H_* - H\left(k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2\right)^{-1} k(\boldsymbol{x}, \boldsymbol{x_*}).$$

Dann sind der A-posteriori Erwartungswertvektor und die A-posteriori Varianz-Kovarianzmatrix folgendermaßen definiert:

$$\overline{\boldsymbol{g_*}} = H^{*T}\overline{\boldsymbol{\beta}} + k(\boldsymbol{x}, \boldsymbol{x_*})^T \left(k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2\right)^{-1} \left(\boldsymbol{y} - H^T\overline{\boldsymbol{\beta}}\right) = \overline{\boldsymbol{f_*}} + R^T\overline{\boldsymbol{\beta}}, \tag{3.25}$$
$$cov(\boldsymbol{g_*}) = cov(\boldsymbol{f_*}) + R^T \left(B^{-1} + H\left(k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2\right)^{-1} H^T\right)^{-1} R.$$

Betrachten wir die Grenzen der obigen Gleichungen (3.25), erkennen wir, dass die A-priori Werte von $\boldsymbol{\beta}$ uninteressant sind. Deshalb vernachlässigen wir den Einfluss von B und $\boldsymbol{b}$,

d.h. wir nahmen an, dass $B^{-1} \to 0$. Dann erhalten wir A-Posteriori Schätzungen, die unabhängig von B und $\boldsymbol{b}$ sind.

$$\overline{\boldsymbol{g_*}} = \overline{\boldsymbol{f_*}} + R^T \overline{\boldsymbol{\beta}},$$
$$cov(\boldsymbol{g_*}) = cov(\boldsymbol{f_*}) + R^T \left(H \left(k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 \right)^{-1} H^T \right)^{-1} R,$$

mit

$$\overline{\boldsymbol{\beta}} = \left(H \left(k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 \right)^{-1} H^T \right)^{-1} H \left(k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 \right)^{-1} \boldsymbol{y},$$
$$R = H_* - H \left(k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 \right)^{-1} k(\boldsymbol{x}, \boldsymbol{x_*}).$$

In der weiteren Arbeit findet die Methode des Universal-Kriging keine Anwendung, da wir für unsere *ALICe 1* Daten Erwartungswert-Funktionen bestimmen können.

Kapitel 4

Praktische Anwendung

Wir wollen die in Kapitel 2 und 3 vorgestellte Theorie der Gauß-Prozess-Regression nutzen um die Alterung von Lithium-Ionen-Zellen zu modellieren. Durch diese Methode erhoffen wir uns eine Verbesserung der Ergebnisse, die durch herkömmliche lineare Regression [Gößler, 2015] erhalten wurden.

Für die praktische Umsetzung der Gauß-Prozess-Regression werden wir folgende Punkte ausführen:

- Bestimmung von geeigneten Test- und Trainingsdaten (Abschnitt 1).
- Bestimmung plausibler Kovarianz-Funktionen (Abschnitt 2).
- Bestimmung von plausiblen Hyperparametern (Abschnitt 3).
- Bestimmung von plausiblen Erwartungswert-Funktionen (Abschnitt 4).

Das Endergebnis der Gauß-Prozess-Regression wird anschließend mit dem Endergebnis der linearen Regression, die im Zuge der Masterarbeit von *G. Gößler* [Gößler, 2015] durchgeführt wurde, verglichen (Abschnitt 5).

1 Wahl der Test- und Trainingsdaten

Bevor wir mit der eigentlichen Gauß-Prozess-Regression beginnen können, benötigen wir einen Test-und Trainingsdatensatz. Diese Datensätze werden aus einer Teilmenge der Daten gebildet, die im Rahmen des *K2 Projektes E3T3 (ALICe)* [Cifrain, 2015] des *VIRTUAL VEHICLE Research Center (ViF)* erstellt worden sind. Diese Daten werden im Folgenden als *ALICe 1* Daten bezeichnet.

Wir wollen uns vor der Wahl der Test-und Trainingsdaten kurz mit den urspünglichen *ALICe 1* Daten und deren Entstehung beschäftigen.

1.1 ALICe 1 Daten

Der Ausgang der *ALICe 1* Daten bildet ein Versuchsplan der von *G.Pregartner* und *W.Prochaska* [Pregartner, 2012] entwickelt wurde. Dieser Versuchsplan ist in Tabelle 4.1 aufgelistet.

LP	**T**	**CC**	**ADC**	**PDC**	**F**	**SoC**	**dSoC**
C01	−10	0.2	0	0.2	0.000333	15	0.01
C02	40	0.2	0	0.2	0.000333	15	0.01
C03	5	0.2	0	0.2	0.000333	55	0.01
C04	40	0.2	0	0.2	0.000333	55	0.01
C05	−10	0.2	0	0.2	0.000333	95	0.01
C06	40	0.2	0	0.2	0.000333	95	0.01
L01	20	0.2	2	12	0.03	25	2.5
L02	5	2.4	1	12	0.1	25	2.5
L03	40	0.2	2	12	0.2	25	2.5
L04	−10	0.2	0.2	0.2	0.5	25	2.5
L05	40	0.2	0.2	0.2	0.5	25	2.5
L06	20	2.4	0.2	0.2	0.5	25	2.5
L07	40	2.4	1	3	0.03	55	2.5
L08	40	0.2	1	12	0.03	80	2.5
L09	5	0.2	1	10	0.2	80	2.5
L10	20	2.4	2	12	0.2	80	2.5
L11	40	0.2	0.2	0.2	0.5	80	2.5
L12	−10	2.4	0.2	0.2	0.5	80	2.5
L13	20	0.2	1	3	0.03	25	15
L14	40	2.4	2	3	0.03	25	15
L15	−10	0.2	0.2	12	0.03	25	15
L16	40	2.4	0.2	12	0.03	25	15
L17	5	2.4	0.2	0.2	0.5	25	15
L18	40	2.4	2	3	0.03	80	15
L19	−10	0.8	0.2	12	0.03	80	15
L20	40	2.4	0.2	12	0.03	80	15
L21	5	0.2	0.2	0.2	0.5	80	15
L22	20	0.2	0.2	0.2	0.000333	55	50
L23	−10	2.4	0.2	0.2	0.1	55	50
L24	40	0.2	2	12	0.2	55	50
L25	−10	0.2	0.2	0.2	0.000333	55	80
L26	40	0.2	0.2	0.2	0.000333	55	80
L27	40	0.2	0.2	6	0.000333	55	80
L28	40	2.4	0.2	0.2	0.03	55	80
L29	5	0.2	0.2	12	0.03	55	80
L30	40	2.4	2	12	0.03	55	80
L31	20	0.2	1	12	0.1	55	80
L32	5	2.4	1	12	0.1	55	80
L33	−10	0.2	0.2	0.2	0.5	55	80
L34	20	2.4	0.2	0.2	0.5	55	80
L35	40	0.2	2	3	0.5	55	80

Tabelle 4.1: Ausgangsversuchsplan für ALICe 1 Experiment, $n = 41$.

Für diesen Versuchsplan wurden 7 Faktoren verwendet, die eine unterschiedliche Anzahl an Einstellungen besitzen. Die Beschreibung der Faktoren befindet sich in Kapitel 2, Abschnitt 2.

Faktoren	**Faktor-Einstellungen.**								
T	−10	5	20	40					
CC	0.2	0.8	2.4						
ADC	0	0.2	1	2	3	4	6	8	
PDC	0.2	1	3	4	6	8	10	12	14
F	0.000333	0.03	0.06	0.1	0.2	0.5			
SoC	15	25	55	80	95				
dSoC	0.01	2.5	15	50	80				

Tabelle 4.2: Auflistung der Faktoren und ihre Einstellungen.

Im Laufe des *K2 Projektes E3T3 (ALICe)* wurde dieser Versuchsplan angepasst. Details zu diesen Anpassungen können in der Arbeit con *G. Gößler* [Gößler, 2015], sowie im *K2 Projekt E3T3 (ALICe) Projektbericht* [Cifrain, 2015] nachgelesen werden. Des Weiteren wurden die Faktoreinstellungen im Rahmen der Arbeit von *G. Gößler* [Gößler, 2015] auf ein Intervall zwischen −1 und 1 skaliert. Dazu wurde für die Faktoren T, CC, ADC, PDC, SoC und $dSoC$ folgende Skalierungsformel [Stadlober, 2015] verwendet:

$$x_{neu} = \frac{x - (x_{min} + x_{max})/2}{(x_{max} - x_{min})/2}, \tag{4.1}$$

wobei mit x der Wert des Faktors vor der Skalierung bezeichnet wird. Mit x_{min} bezeichnen wir den minimalen Wert und mit x_{max} den maximalen Wert des Faktors.

Für den Faktor F verwenden wir für die Skalierung eine logarithmierte Form der Formel (4.1):

$$x_{neu} = \frac{\log x - (\log x_{min} + \log x_{max})/2}{(\log x_{max} - \log x_{min})/2}.$$

x, x_{min} und x_{max} sind analog definiert wie zuvor.

Die minimalen sowie maximalen Werte der einzelnen Faktoren sind in Tabelle 4.3 aufgelistet. Am Ende erhalten wir einen Versuchsplan (Tabelle 4.4) mit 41 verschiedenen Einstellungen. Diese verschiedenen Einstellungen werden auch als Lastpunkte bezeichnet.

Mit Hilfe dieses Versuchsplans (Tabelle 4.4) wurden Alterungsexperimente durchgeführt. Dabei war geplant, dass jeder Lastpunkt mit drei Zellen getestet wird. Das heißt, für jeden Lastpunkt sollten am Ende des Experiments genau drei Werte existieren, die angeben, ab wie vielen Tagen eine Zelle nicht mehr einsatzfähig ist. Eine Batteriezelle gilt als nicht einsatzfähig, sobald ihre Kapazität unter 70% liegt. Der Tag an dem eine Zelle diese Schranke unterschreitet wird als Lebensende oder auch End-of-Life (kurz EoL) bezeichnet.

Faktoren	**minimaler Wert**	**maximaler Wert**
T	−10	40
CC	0.2	2.4
ADC	0	8
PDC	0.2	14
F	0.000333	0.5
SoC	15	95
dSoC	0.01	80

Tabelle 4.3: Auflistung minimaler und maximaler Werte der Faktoren.

Im Laufe des *K2 Projektes E3T3 (ALICe)* erreichten allerdings nur wenige Zellen ihr tatsächliches Lebensende. Des Weiteren waren aus verschiedenen Gründen nicht immer alle Lastpunkte mit genau drei Zellen getestet worden. Um diese Fehler im Experiment zu beheben wurden im Rahmen der Arbeit von *G. Gößler* [Gößler, 2015] zunächst die Lebensdauern der Lithium-Ionen-Zellen extrapoliert. Dazu verwendete *G. Gößler* ein einfaches exponentielles Modell:

$$y(t) = \beta_1 \exp(-\beta_2 t) \;\; \beta_1, \beta_2 \in \mathbb{R}.$$

Mit $y(t)$ bezeichnet man die Kapazität der Zelle nach t Tagen.

Die so entstandenen Daten wurden anschließend noch balanciert, damit die am Anfang getroffene Annahmen, dass jeder Lastpunkt an drei Zellen getestet wird, wieder erfüllt ist. Dazu werden unnötige Beobachtungen gelöscht, falls für einen Lastpunkt mehr als drei EoL-Werte vorliegen. Für Lastpunkte, die weniger als drei EoL-Werte besitzen, wurden Beobachtungen hinzugefügt. Details zur Extrapolation sowie zum Balancieren der Daten können in der Arbeit von *G. Gößler* [Gößler, 2015] nachgelesen werden.

Am Ende dieser Extrapolation und Balancierung der Daten erhalten wir unseren *ALICe 1* Datensatz (Tabelle 4.5 - Tabelle 4.7) aus dem wir im Folgenden unsere Test- und Trainingsdaten ermitteln.

1.2 Erstellung Test- und Trainingsdaten

Bevor wir den Test- und Trainingsdatensatz erzeugen, müssen wir die *ALICe 1* Daten nach ihren EoL-Werten mitteln (siehe Tabelle 4.8). Der Grund dafür ist, dass wir - wie schon zuvor erwähnt - für jeden Lastpunkt des Versuchsplans drei EoL-Werte erhalten. Eine Gauß-Prozess-Regression mit ungemittelten Daten würde zu einer singulären Kovarianz-Funktion führen. Diese Matrix wäre nicht mehr invertierbar und wir würden dann keine Ergebnisse für den A-posteriori Erwartungswertvektor und der A-posteriori Varianz-Kovarianz-matrix erhalten.

Das Mitteln der Daten wurde mit Hilfe der `R`-Funktion `aggregate` und der Funktion `mean` durchgeführt.

LP	T	CC	ADC	PDC	F	SoC	dSoC
C01	−1.00	−1.00	−1.00	−1.00	−1.00	−1.00	−1.00
C02	1.00	−1.00	−1.00	−1.00	−1.00	−1.00	−1.00
C03	−0.40	−1.00	−1.00	−1.00	−1.00	0.00	−1.00
C04	0.20	−1.00	−1.00	−1.00	−1.00	0.00	−1.00
C05	−1.00	−1.00	−1.00	−1.00	−1.00	1.00	−1.00
C06	1.00	−1.00	−1.00	−1.00	−1.00	1.00	−1.00
L01	0.20	−0.83	−0.50	0.71	0.24	−0.75	−0.94
L02	−0.40	1.00	−0.75	0.71	0.57	−0.75	−0.94
L03	1.00	−0.83	−0.50	0.71	0.75	−0.75	−0.94
L04	−1.00	−0.83	−0.95	−0.97	−1.00	−0.75	−0.94
L05	1.00	−0.83	−0.95	−0.97	−1.00	−0.75	−0.94
L06	0.20	1.00	−0.95	−0.97	−1.00	−0.75	−0.94
L07	1.00	1.00	−0.75	−0.57	0.24	0.00	−0.94
L08	−1.00	−0.83	−0.75	0.43	0.24	0.63	−0.94
L09	−0.40	−0.83	−0.75	0.43	0.75	0.63	−0.94
L10	−1.00	0.00	−0.50	0.43	0.75	0.13	−0.94
L11	1.00	−0.83	−0.95	−0.97	−1.00	0.63	−0.94
L12	−1.00	0.00	−0.95	−0.97	−1.00	0.13	−0.94
L13	0.20	−0.83	−0.75	−0.57	0.24	−0.75	−0.63
L14	1.00	1.00	−0.50	−0.57	0.24	−0.75	−0.63
L15	−1.00	−0.83	−0.95	−0.14	0.24	0.00	−0.63
L16	1.00	1.00	−0.95	0.71	0.24	−0.75	−0.63
L17	−0.40	1.00	−0.95	−0.97	−1.00	−0.75	−0.63
L18	1.00	1.00	−0.50	−0.57	0.24	0.63	−0.63
L19	−1.00	−0.67	−0.95	−0.14	0.24	0.38	−0.63
L20	1.00	1.00	−0.95	0.71	0.24	0.63	−0.63
L21	−0.40	−0.83	−0.95	−0.97	−1.00	0.63	−0.63
L22	0.20	−0.83	−0.95	−0.97	−0.97	0.00	0.25
L23	−1.00	−0.83	−0.95	−0.97	−1.00	0.00	0.25
L24	1.00	−0.83	−0.50	0.71	0.75	0.00	0.25
L25	−1.00	−0.83	−0.95	−0.97	−0.97	0.00	0.25
L26	1.00	−0.83	−0.95	−0.97	−0.97	0.00	1.00
L27	−1.00	−0.83	−0.95	−0.57	−0.97	0.00	0.25
L28	1.00	1.00	−0.95	−0.97	−1.00	0.00	0.75
L29	−0.40	−0.83	−0.95	0.71	0.24	0.13	0.75
L30	1.00	1.00	0.25	0.71	0.24	0.00	0.75
L31	0.20	−0.83	−0.75	0.71	0.57	0.00	1.00
L32	−0.40	0.00	−0.75	0.71	0.57	0.00	0.25
L33	−1.00	−0.83	−0.68	−0.71	1.00	0.00	0.25
L34	0.20	1.00	−0.95	−0.97	−1.00	−0.13	0.50
L35	1.00	−0.83	−0.50	−0.57	1.00	0.00	1.00

Tabelle 4.4: Skalierter Versuchsplan ALICe 1, $n = 41$.

LP	EoL	T	CC	ADC	PDC	F	SoC	dSoC
C01	5625.80	−1.00	−1.00	−1.00	−1.00	−1.00	−1.00	−1.00
C01	5833.13	−1.00	−1.00	−1.00	−1.00	−1.00	−1.00	−1.00
C01	5113.88	−1.00	−1.00	−1.00	−1.00	−1.00	−1.00	−1.00
C02	4452.66	1.00	−1.00	−1.00	−1.00	−1.00	−1.00	−1.00
C02	3328.98	1.00	−1.00	−1.00	−1.00	−1.00	−1.00	−1.00
C02	5169.87	1.00	−1.00	−1.00	−1.00	−1.00	−1.00	−1.00
C03	2434.21	−0.40	−1.00	−1.00	−1.00	−1.00	0.00	−1.00
C03	2503.79	−0.40	−1.00	−1.00	−1.00	−1.00	0.00	−1.00
C03	2845.07	−0.40	−1.00	−1.00	−1.00	−1.00	0.00	−1.00
C04	4030.89	0.20	−1.00	−1.00	−1.00	−1.00	0.00	−1.00
C04	957.79	0.20	−1.00	−1.00	−1.00	−1.00	0.00	−1.00
C04	2686.41	0.20	−1.00	−1.00	−1.00	−1.00	0.00	−1.00
C05	3159.68	−1.00	−1.00	−1.00	−1.00	−1.00	1.00	−1.00
C05	793.79	−1.00	−1.00	−1.00	−1.00	−1.00	1.00	−1.00
C05	1546.57	−1.00	−1.00	−1.00	−1.00	−1.00	1.00	−1.00
C06	1347.33	1.00	−1.00	−1.00	−1.00	−1.00	1.00	−1.00
C06	1098.40	1.00	−1.00	−1.00	−1.00	−1.00	1.00	−1.00
C06	1174.16	1.00	−1.00	−1.00	−1.00	−1.00	1.00	−1.00
L01	3902.74	0.20	−0.83	−0.50	0.71	0.24	−0.75	−0.94
L01	3728.50	0.20	−0.83	−0.50	0.71	0.24	−0.75	−0.94
L01	3843.74	0.20	−0.83	−0.50	0.71	0.24	−0.75	−0.94
L02	4137.66	−0.40	1.00	−0.75	0.71	0.57	−0.75	−0.94
L02	4022.69	−0.40	1.00	−0.75	0.71	0.57	−0.75	−0.94
L02	4937.05	−0.40	1.00	−0.75	0.71	0.57	−0.75	−0.94
L03	1564.97	1.00	−0.83	−0.50	0.71	0.75	−0.75	−0.94
L03	1452.00	1.00	−0.83	−0.50	0.71	0.75	−0.75	−0.94
L03	1424.61	1.00	−0.83	−0.50	0.71	0.75	−0.75	−0.94
L04	3806.33	−1.00	−0.83	−0.95	−0.97	−1.00	−0.75	−0.94
L04	3066.00	−1.00	−0.83	−0.95	−0.97	−1.00	−0.75	−0.94
L04	3564.59	−1.00	−0.83	−0.95	−0.97	−1.00	−0.75	−0.94
L05	1806.85	1.00	−0.83	−0.95	−0.97	−1.00	−0.75	−0.94
L05	1035.53	1.00	−0.83	−0.95	−0.97	−1.00	−0.75	−0.94
L05	1596.28	1.00	−0.83	−0.95	−0.97	−1.00	−0.75	−0.94
L06	4187.37	0.20	1.00	−0.95	−0.97	−1.00	−0.75	−0.94
L06	4242.52	0.20	1.00	−0.95	−0.97	−1.00	−0.75	−0.94
L06	4699.00	0.20	1.00	−0.95	−0.97	−1.00	−0.75	−0.94
L07	551.37	1.00	1.00	−0.75	−0.57	0.24	0.00	−0.94
L07	606.53	1.00	1.00	−0.75	−0.57	0.24	0.00	−0.94
L07	578.95	1.00	1.00	−0.75	−0.57	0.24	0.00	−0.94
L08	2150.43	−1.00	−0.83	−0.75	0.43	0.24	0.63	−0.94
L08	2209.67	−1.00	−0.83	−0.75	0.43	0.24	0.63	−0.94
L08	1849.00	−1.00	−0.83	−0.75	0.43	0.24	0.63	−0.94

Tabelle 4.5: ALICe 1 Datensatz Teil 1, $n = 3 \times 14 = 42$.

LP	EoL	T	CC	ADC	PDC	F	SoC	dSoC
L09	2492.11	−0.40	−0.83	−0.75	0.43	0.75	0.63	−0.94
L09	1970.76	−0.40	−0.83	−0.75	0.43	0.75	0.63	−0.94
L09	2080.07	−0.40	−0.83	−0.75	0.43	0.75	0.63	−0.94
L10	939.65	−1.00	0.00	−0.50	0.43	0.75	0.13	−0.94
L10	1732.00	−1.00	0.00	−0.50	0.43	0.75	0.13	−0.94
L10	1684.50	−1.00	0.00	−0.50	0.43	0.75	0.13	−0.94
L11	827.00	1.00	−0.83	−0.95	−0.97	−1.00	0.63	−0.94
L11	854.13	1.00	−0.83	−0.95	−0.97	−1.00	0.63	−0.94
L11	762.64	1.00	−0.83	−0.95	−0.97	−1.00	0.63	−0.94
L12	3247.00	−1.00	0.00	−0.95	−0.97	−1.00	0.13	−0.94
L12	3113.49	−1.00	0.00	−0.95	−0.97	−1.00	0.13	−0.94
L12	2895.58	−1.00	0.00	−0.95	−0.97	−1.00	0.13	−0.94
L13	3825.00	0.20	−0.83	−0.75	−0.57	0.24	−0.75	−0.63
L13	4274.00	0.20	−0.83	−0.75	−0.57	0.24	−0.75	−0.63
L13	2563.70	0.20	−0.83	−0.75	−0.57	0.24	−0.75	−0.63
L14	448.30	1.00	1.00	−0.50	−0.57	0.24	−0.75	−0.63
L14	566.71	1.00	1.00	−0.50	−0.57	0.24	−0.75	−0.63
L14	683.12	1.00	1.00	−0.50	−0.57	0.24	−0.75	−0.63
L15	3453.00	−1.00	−0.83	−0.95	−0.14	0.24	0.00	−0.63
L15	3164.29	−1.00	−0.83	−0.95	−0.14	0.24	0.00	−0.63
L15	2841.58	−1.00	−0.83	−0.95	−0.14	0.24	0.00	−0.63
L16	1407.20	1.00	1.00	−0.95	0.71	0.24	−0.75	−0.63
L16	773.65	1.00	1.00	−0.95	0.71	0.24	−0.75	−0.63
L16	1265.29	1.00	1.00	−0.95	0.71	0.24	−0.75	−0.63
L17	4611.09	−0.40	1.00	−0.95	−0.97	−1.00	−0.75	−0.63
L17	4529.51	−0.40	1.00	−0.95	−0.97	−1.00	−0.75	−0.63
L17	4569.67	−0.40	1.00	−0.95	−0.97	−1.00	−0.75	−0.63
L18	623.00	1.00	1.00	−0.50	−0.57	0.24	0.63	−0.63
L18	525.81	1.00	1.00	−0.50	−0.57	0.24	0.63	−0.63
L18	508.91	1.00	1.00	−0.50	−0.57	0.24	0.63	−0.63
L19	3074.64	−1.00	−0.67	−0.95	−0.14	0.24	0.38	−0.63
L19	2539.28	−1.00	−0.67	−0.95	−0.14	0.24	0.38	−0.63
L19	1843.11	−1.00	−0.67	−0.95	−0.14	0.24	0.38	−0.63
L20	706.83	1.00	1.00	−0.95	0.71	0.24	0.63	−0.63
L20	529.88	1.00	1.00	−0.95	0.71	0.24	0.63	−0.63
L20	625.91	1.00	1.00	−0.95	0.71	0.24	0.63	−0.63
L21	3017.50	−0.40	−0.83	−0.95	−0.97	−1.00	0.63	−0.63
L21	3301.32	−0.40	−0.83	−0.95	−0.97	−1.00	0.63	−0.63
L21	2934.00	−0.40	−0.83	−0.95	−0.97	−1.00	0.63	−0.63
L22	2555.07	0.20	−0.83	−0.95	−0.97	−0.97	0.00	0.25
L22	2482.99	0.20	−0.83	−0.95	−0.97	−0.97	0.00	0.25
L22	2519.03	0.20	−0.83	−0.95	−0.97	−0.97	0.00	0.25

Tabelle 4.6: ALICe 1 Datensatz Teil 2, $n = 3 \times 14 = 42$.

LP	EoL	T	CC	ADC	PDC	F	SoC	dSoC
L23	2760.10	−1.00	−0.83	−0.95	−0.97	−1.00	0.00	0.25
L23	2605.00	−1.00	−0.83	−0.95	−0.97	−1.00	0.00	0.25
L23	3150.72	−1.00	−0.83	−0.95	−0.97	−1.00	0.00	0.25
L24	566.21	1.00	−0.83	−0.50	0.71	0.75	0.00	0.25
L24	530.47	1.00	−0.83	−0.50	0.71	0.75	0.00	0.25
L24	488.00	1.00	−0.83	−0.50	0.71	0.75	0.00	0.25
L25	4203.31	−1.00	−0.83	−0.95	−0.97	−0.97	0.00	0.25
L25	1090.17	−1.00	−0.83	−0.95	−0.97	−0.97	0.00	0.25
L25	2701.42	−1.00	−0.83	−0.95	−0.97	−0.97	0.00	0.25
L26	712.77	1.00	−0.83	−0.95	−0.97	−0.97	0.00	1.00
L26	570.99	1.00	−0.83	−0.95	−0.97	−0.97	0.00	1.00
L26	546.45	1.00	−0.83	−0.95	−0.97	−0.97	0.00	1.00
L27	868.11	−1.00	−0.83	−0.95	−0.57	−0.97	0.00	0.25
L27	840.72	−1.00	−0.83	−0.95	−0.57	−0.97	0.00	0.25
L27	597.64	−1.00	−0.83	−0.95	−0.57	−0.97	0.00	0.25
L28	624.50	1.00	1.00	−0.95	−0.97	−1.00	0.00	0.75
L28	458.24	1.00	1.00	−0.95	−0.97	−1.00	0.00	0.75
L28	525.57	1.00	1.00	−0.95	−0.97	−1.00	0.00	0.75
L29	4564.75	−0.40	−0.83	−0.95	0.71	0.24	0.13	0.75
L29	4730.73	−0.40	−0.83	−0.95	0.71	0.24	0.13	0.75
L29	4170.71	−0.40	−0.83	−0.95	0.71	0.24	0.13	0.75
L30	191.44	1.00	1.00	0.25	0.71	0.24	0.00	0.75
L30	129.01	1.00	1.00	0.25	0.71	0.24	0.00	0.75
L30	184.17	1.00	1.00	0.25	0.71	0.24	0.00	0.75
L31	1455.00	0.20	−0.83	−0.75	0.71	0.57	0.00	1.00
L31	1804.26	0.20	−0.83	−0.75	0.71	0.57	0.00	1.00
L31	1802.30	0.20	−0.83	−0.75	0.71	0.57	0.00	1.00
L32	491.09	−0.40	0.00	−0.75	0.71	0.57	0.00	0.25
L32	306.64	−0.40	0.00	−0.75	0.71	0.57	0.00	0.25
L32	398.41	−0.40	0.00	−0.75	0.71	0.57	0.00	0.25
L33	1639.00	−1.00	−0.83	−0.68	−0.71	1.00	0.00	0.25
L33	1774.15	−1.00	−0.83	−0.68	−0.71	1.00	0.00	0.25
L33	1756.70	−1.00	−0.83	−0.68	−0.71	1.00	0.00	0.25
L34	448.91	0.20	1.00	−0.95	−0.97	−1.00	−0.13	0.50
L34	454.00	0.20	1.00	−0.95	−0.97	−1.00	−0.13	0.50
L34	500.01	0.20	1.00	−0.95	−0.97	−1.00	−0.13	0.50
L35	446.22	1.00	−0.83	−0.50	−0.57	1.00	0.00	1.00
L35	494.46	1.00	−0.83	−0.50	−0.57	1.00	0.00	1.00
L35	507.27	1.00	−0.83	−0.50	−0.57	1.00	0.00	1.00

Tabelle 4.7: ALICe 1 Datensatz Teil 3, $n = 3 \times 13 = 39$.

```
> alice1_mean <- aggregate(x = alice1_daten, FUN = mean,
+              by =list(alice1$CODE), na.rm = TRUE)
```

Nach Mittelung der *ALICe 1* Daten (Tabelle 4.8) wählen wir aus 41 Lastpunkten zufällig 8 Lastpunkte aus, die unsere Testdaten $\boldsymbol{x}_*$ beschreiben sollen. Die restlichen 33 Lastpunkte werden als Trainingsdaten $\boldsymbol{x}$ verwendet. Der Grund dafür, dass wir genau 8 Lastpunkte als Testdaten $\boldsymbol{x}_*$ auswählen liegt darin, dass wir genug Lastpunkte für die Trainingsdaten (ca. $\frac{4}{5}$) behalten wollen.

Mit diesen zufällig gewählten Test- und Trainingsdaten führen wir eine einfache Gauß-Prozess-Regression (Kapitel 3, Abschnitt 2) mit mehreren Faktoren durch. Die Erwartungswert-Funktion wird dabei auf 0 gesetzt. Des Weiteren wählen wir als Kovarianz-Funktion die quadratische Exponentialfunktion (3.11) mit folgenden Hyperparametern:

$$l = 1, \sigma_f = 1, \sigma_n = 0.$$

Um zu entscheiden, ob die gewählten Test- und Trainingsdaten für die Gauß-Prozess-Regression geeignet sind, vergleichen wir die geschätzten A-posteriori Erwartungswerte der Gauß-Prozess-Regression mit den gemessenen bzw. extrapolierten EoL-Werten der einzelnen Testdaten. Je näher die Werte zusammen liegen, desto besser ist das Ergebnis.

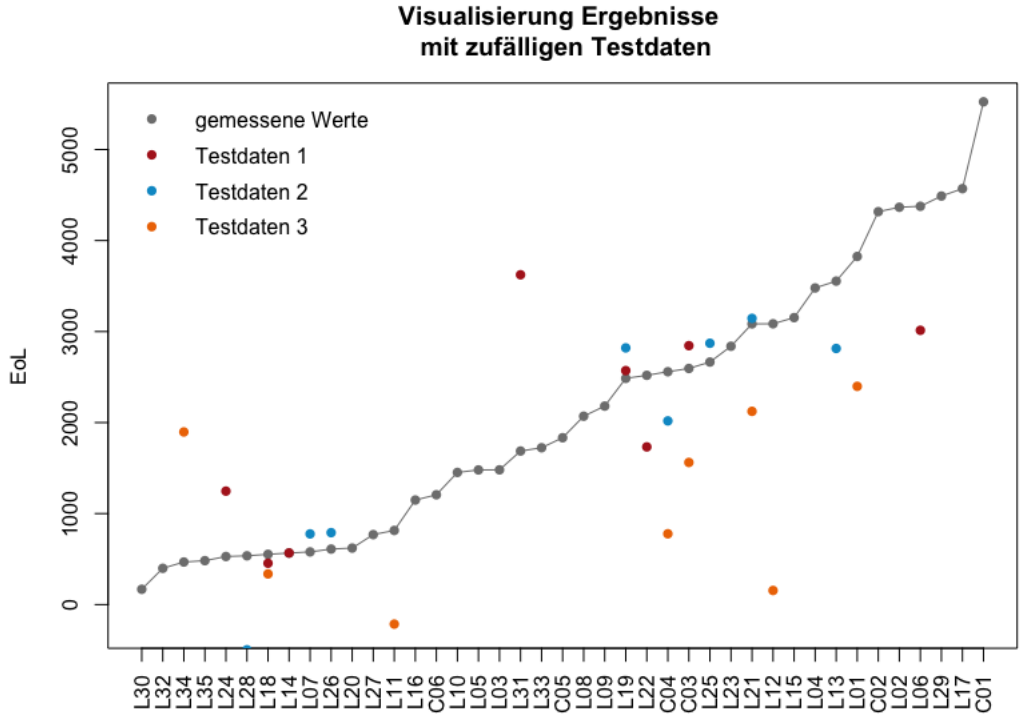

Abbildung 4.1: Visualisierung Ergebnisse mit zufälligen Testdaten.

LP	EoL	T	CC	ADC	PDC	F	SoC	dSoC
$C01$	5524.27	−1.00	−1.00	−1.00	−1.00	−1.00	−1.00	−1.00
$C02$	4317.17	1.00	−1.00	−1.00	−1.00	−1.00	−1.00	−1.00
$C03$	2594.36	−0.40	−1.00	−1.00	−1.00	−1.00	0	−1.00
$C04$	2558.36	0.20	−1.00	−1.00	−1.00	−1.00	0	−1.00
$C05$	1833.35	−1.00	−1.00	−1.00	−1.00	−1.00	1.00	−1.00
$C06$	1206.63	1.00	−1.00	−1.00	−1.00	−1.00	1.00	−1.00
$L01$	3825.00	0.20	−0.83	−0.50	0.71	0.24	−0.75	−0.94
$L02$	4365.80	−0.40	1.00	−0.75	0.71	0.57	−0.75	−0.94
$L03$	1480.53	1.00	−0.83	−0.50	0.71	0.75	−0.75	−0.94
$L04$	3478.97	−1.00	−0.83	−0.95	−0.97	−1.00	−0.75	−0.94
$L05$	1479.55	1.00	−0.83	−0.95	−0.97	−1.00	−0.75	−0.94
$L06$	4376.30	0.20	1.00	−0.95	−0.97	−1.00	−0.75	−0.94
$L07$	578.95	1.00	1.00	−0.75	−0.57	0.24	0	−0.94
$L08$	2069.70	−1.00	−0.83	−0.75	0.43	0.24	0.63	−0.94
$L09$	2180.98	−0.40	−0.83	−0.75	0.43	0.75	0.63	−0.94
$L10$	1452.05	−1.00	0	−0.50	0.43	0.75	0.13	−0.94
$L11$	814.59	1.00	−0.83	−0.95	−0.97	−1.00	0.63	−0.94
$L12$	3085.36	−1.00	0	−0.95	−0.97	−1.00	0.13	−0.94
$L13$	3554.23	0.20	−0.83	−0.75	−0.57	0.24	−0.75	−0.63
$L14$	566.045	1.00	1.00	−0.50	−0.57	0.24	−0.75	−0.63
$L15$	3152.96	−1.00	−0.83	−0.95	−0.14	0.24	0	−0.63
$L16$	1148.72	1.00	1.00	−0.95	0.71	0.24	−0.75	−0.63
$L17$	4570.09	−0.40	1.00	−0.95	−0.97	−1.00	−0.75	−0.63
$L18$	552.57	1.00	1.00	−0.50	−0.57	0.24	0.63	−0.63
$L19$	2485.68	−1.00	−0.67	−0.95	−0.14	0.24	0.38	−0.63
$L20$	620.88	1.00	1.00	−0.95	0.71	0.24	0.63	−0.63
$L21$	3084.27	−0.40	−0.83	−0.95	−0.97	−1.00	0.63	−0.63
$L22$	2519.03	0.20	−0.83	−0.95	−0.97	−0.97	0	0.25
$L23$	2838.61	−1.00	−0.83	−0.95	−0.97	−1.00	0	0.25
$L24$	528.23	1.00	−0.83	−0.50	0.71	0.75	0	0.25
$L25$	2664.97	−1.00	−0.83	−0.95	−0.97	−0.97	0	0.25
$L26$	610.07	1.00	−0.83	−0.95	−0.97	−0.97	0	1.00
$L27$	768.82	−1.00	−0.83	−0.95	−0.57	−0.97	0	0.25
$L28$	536.10	1.00	1.00	−0.95	−0.97	−1.00	0	0.75
$L29$	4488.73	−0.40	−0.83	−0.95	0.71	0.24	0.13	0.75
$L30$	168.21	1.00	1.00	0.25	0.71	0.24	0	0.75
$L31$	1687.19	0.20	−0.83	−0.75	0.71	0.57	0	1.00
$L32$	398.71	−0.40	0	−0.75	0.71	0.57	0	0.25
$L33$	1723.28	−1.00	−0.83	−0.68	−0.71	1.00	0	0.25
$L34$	467.64	0.20	1.00	−0.95	−0.97	−1.00	−0.13	0.50
$L35$	482.65	1.00	−0.83	−0.50	−0.57	1.00	0	1.00

Tabelle 4.8: Gemittelte ALICe 1 Daten, $n = 41$.

Testdaten 1: C03,L06,L14,L18,L19,L22,L24,L31				
Lastpunkt	**Geschätzter Wert**	**Gemessener Wert**	**Absoluter Fehler**	**Relativer Fehler**
C03	2845.08	2594.36	250.72	9.66%
L06	3013.65	4376.30	1362.65	31.14%
L14	567.55	566.04	1.50	0.27%
L18	454.53	552.57	98.04	17.74%
L19	2570.93	2485.68	85.26	3.43%
L22	1732.62	2519.03	786.41	31.22%
L24	1246.82	528.23	718.59	136.04%
L31	3623.75	1687.19	1936.56	114.78%
Gesamtabstand $(l = 1, \sigma_f = 1, \sigma_n = 0)$			5239.73	

Tabelle 4.9: Testdaten 1

Testdaten 2: C04,L07,L13,L19,L21,L25,L26,L28				
Lastpunkt	**Geschätzter Wert**	**Gemessener Wert**	**Absoluter Fehler**	**Relativer Fehler**
C04	2018.38	2558.36	539.99	21.11%
L07	776.66	578.95	197.71	34.15%
L13	2814.16	3554.23	740.07	20.82%
L19	2820.40	2485.68	334.72	13.47%
L21	3144.86	3084.27	60.59	1.96%
L25	2871.29	2664.97	206.33	7.74%
L26	790.46	610.07	180.39	29.57%
L28	−496.54	536.10	1032.64	192.62%
Gesamtabstand $(l = 1, \sigma_f = 1, \sigma_n = 0)$			3292.45	

Tabelle 4.10: Testdaten 2

Testdaten 3: C03,C04,L01,L11,L12,L18,L21,L34				
Lastpunkt	**Geschätzter Wert**	**Gemessener Wert**	**Absoluter Fehler**	**Relativer Fehler**
C03	1561.72	2594.36	1032.64	39.80%
C04	777.91	2558.36	1780.45	69.59%
L01	2398.00	3825.00	1426.99	37.31%
L11	−214.23	814.59	1028.82	126.30%
L12	155.13	3085.36	2930.22	94.97%
L18	337.02	552.57	215.56	39.01%
L21	2123.28	3084.27	960.99	31.16%
L34	1896.64	467.64	1429.00	305.58%
Gesamtabstand $(l = 1, \sigma_f = 1, \sigma_n = 0)$			10804.67	

Tabelle 4.11: Testdaten 3

Wie wir in den Tabellen 4.9 – 4.11 und Abb. 4.1 erkennen, hängt das Endergebnis der Gauß-Prozess-Regression stark davon ab, welche Test- und Trainingsdaten gewählt werden. Des Weiteren erkennen wir, dass vor allem bei *Testdaten 1* und *Testdaten 2* sehr viele Testpunkte gut getroffen werden, allerdings liefern immer wieder ein bis zwei Punkte Ergebnisse, die weit von den gemessenen Punkten entfernt sind und dadurch das Gesamtergebnis stark beeinflussen. *Testdaten 3* liefert hingegen das schlechteste Ergebnis, da nur ein Testpunkt wirklich gut getroffen wurde. Ein Grund dafür kann sein, dass die Werte der gewählten Testdaten eher hohe EoL-Werte besitzen, die nicht gemessen, sondern extrapoliert wurden.

Um nun gute Ergebnisse für die Gauß-Prozess-Regression zu erhalten, wollen wir jene 8 Testpunkte bestimmen, die bezüglich einer gewählten Kovarianz-Funktion, den kleinsten Gesamtabstand zwischen den interpolierten und den gemessenen EoL-Werten liefern.

Für die Bestimmung führen wir eine Kreuzvalidierung durch. Dabei werden abwechselnd immer zwei Testpunkte variabel gesetzt und die restlichen 6 Testpunkte bleiben fest. Anschließend wird für jede mögliche Wahl der zwei variablen Testpunkte eine einfache Gauß-Prozess-Regression durchgeführt. Als Kovarianz-Funktion wird wieder die quadratische Exponentialfunktion mit den Hyperparametern $l = 1, \sigma_f = 1$ und $\sigma_n = 0$ gewählt. Die Erwartungswert-Funktion wird 0 gesetzt. Verbessert sich der Gesamtabstand für die aktuelle Wahl der zwei variablen Testpunkte, wird diese Einstellung gespeichert. Ansonsten wird sie verworfen und eine neue Einstellung wird gewählt. Dies wird solange wiederholt, bis es keine Verbesserung des Gesamtabstandes gibt.

Mit Hilfe dieses Verfahren erhalten wir folgende 8 Lastpunkte als Testdaten:

$$L08, L09, L14, L19, L21, L25, L26, L33. \tag{4.2}$$

Der Gesamtabstand für diese Testdaten beträgt: 1195.26. Genauere Details sind in Tabelle 4.12 aufgelistet. Wie wir anhand der Abb. 4.2 erkennen, liegen diese Testdaten im mittleren Bereich bezüglich ihrer EoL-Werte.

2 Plausible Wahl der Kovarianz-Funktion

Wie wir in Kapitel 3, Abschnitt 3 gesehen haben, gibt es eine Vielzahl an plausiblen Kovarianz-Funktionen. Wir wollen nun herausfinden, welche Kovarianz-Funktion für unsere im Abschnitt 1 gewählten Test- und Trainingsdaten (4.2) am geeignetsten sind. Beim Testen beschränken wir uns dabei auf folgende vier Kovarianz-Funktionen:

- **Quadratische Exponentialfunktion**

$$k_{SE}(\boldsymbol{x}, \boldsymbol{x^*}) = \sigma_f^2 \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x^*}\|^2}{2l^2}\right) + \sigma_n^2 I, \qquad l, \sigma_f^2 = 1,\ \sigma_n^2 = 0.$$

- **Rational quadratische Exponentialfunktion**

$$k_{RQ}(\boldsymbol{x}, \boldsymbol{x}^*) = \left(1 + \frac{\|\boldsymbol{x} - \boldsymbol{x}^*\|^2}{2\alpha l^2}\right)^{-\alpha} + \sigma_n^2 I, \qquad l, \alpha = 1,\ \sigma_n^2 = 0.$$

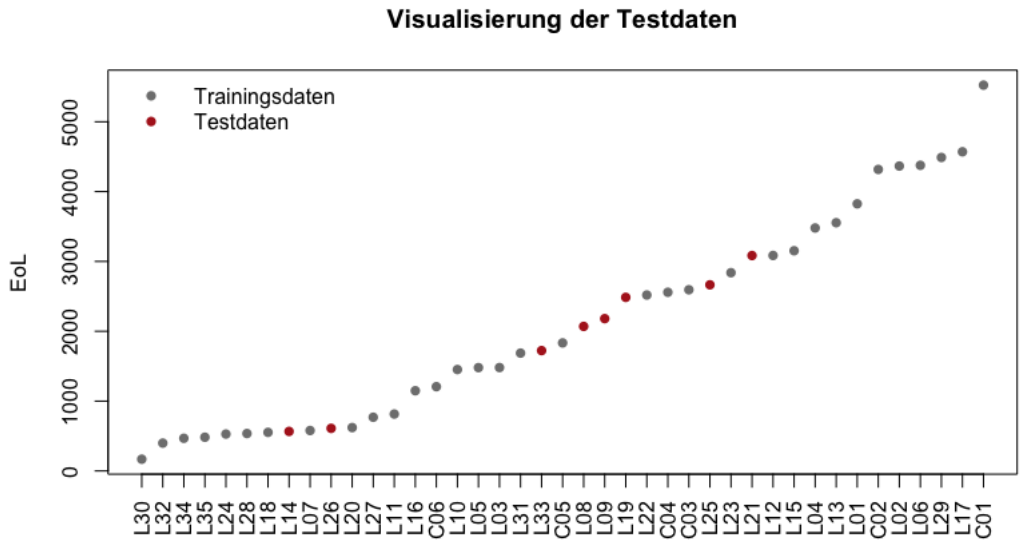

Abbildung 4.2: Visualisierung Lage der berechneten Testdaten.

- **Gamma-Exponentialfunktion**

$$k_{GE}(\boldsymbol{x}, \boldsymbol{x}^*) = \exp\left(-\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}^*\|}{l}\right)^{\gamma}\right) + \sigma_n^2 I \qquad l, \gamma = 1,\ \sigma_n^2 = 0.$$

- **Matern Funktion mit** $\nu = p + \frac{1}{2}$

$$k_{MATERN_{\nu=p+\frac{1}{2}}}(\boldsymbol{x}, \boldsymbol{x}^*) = \exp\left(-\frac{\sqrt{2\nu}\|\boldsymbol{x} - \boldsymbol{x}^*\|}{l}\right) \frac{\Gamma(p+1)}{\Gamma(2p+1)} \sum_{i=0}^{p} \frac{(p+i)!}{i!(p-i)!} \left(\frac{\sqrt{8\nu}\|\boldsymbol{x} - \boldsymbol{x}^*\|}{l}\right)^{p-i} + \sigma_n^2 I, \qquad l, p = 1,\ \sigma_n^2 = 0.$$

Da unsere Test- und Trainingsdaten so gewählt worden sind, dass sie mit Hilfe der quadratischen Exponentialfunktion den geringsten Gesamtabstand liefern, liegt der Verdacht nahe, dass weiterhin die quadratische Exponentialfunktion die geeignetste Wahl ist. Um diesen Verdacht zu bestätigen, führen wir mit allen oben erwähnten Kovarianz-Funktionen eine einfache Gauß-Prozess-Regression durch. Die Erwartungswert-Funktion setzten wir dabei auf 0.

Sowohl der Vergleich der Gesamtabstände (Tabelle 4.12 – 4.15) als auch die grafische Darstellung der Ergebnisse (Abb. 4.3) bestätigen unseren Verdacht, dass die quadratische Exponentialfunktion weiterhin die geeignetste Wahl für unsere Kovarianz-Funktion ist. Die Gesamtabstände der anderen Kovarianz-Funktionen liegen hingegen weit vom Wert der quadratischen Exponentialfunktion entfernt.

Wichtig zu erwähnen ist hier, dass die Hyperparameter der einzelnen Kovarianz-Funktionen noch nicht optimiert wurden. Doch auch der Vergleich der verschiedenen Kovarianz-Funktionen mit optimierten Hyperparametern in Abschnitt 3 liefert uns ein ähnliches Ergebnis.

Quadratische Exponentialfunktion				
Lastpunkt	**Geschätzter Wert**	**Gemessener Wert**	**Absoluter Fehler**	**Relativer Fehler**
L08	2116.30	2069.70	46.60	2.25%
L09	2133.72	2180.98	47.27	2.17%
L14	695.19	566.04	129.15	22.82%
L19	2820.72	2485.68	335.05	13.48%
L21	3138.87	3084.27	54.60	1.77%
L25	2873.41	2664.97	208.44	7.82%
L26	766.88	610.07	156.82	25.70%
L33	1940.62	1723.28	217.34	12.61%
Gesamtabstand $(l = 1, \sigma_f = 1, \sigma_n = 0)$			1195.26	

Tabelle 4.12: Optimale Testdaten mit quadratischer Exponentialfunktion.

Rationale quadratische Exponentialfunktion				
Lastpunkt	**Geschätzter Wert**	**Gemessener Wert**	**Absoluter Fehler**	**Relativer Fehler**
L08	2179.77	2069.70	110.07	5.32%
L09	2005.24	2180.98	175.74	8.06%
L14	1116.30	566.04	550.26	97.21%
L19	2616.19	2485.68	130.51	5.25%
L21	2344.06	3084.27	740.21	24.00%
L25	2849.25	2664.97	184.28	6.91%
L26	1423.19	610.07	813.12	133.28%
L33	2375.19	1723.28	651.90	37.83%
Gesamtabstand $(l = 1, \alpha = 1, \sigma_n = 0)$			3356.10	

Tabelle 4.13: Optimale Testdaten mit rational quadratischer Exponentialfunktion.

Wählen wir nun die Testdaten 3: $C03, C04, L01, L11, L12, L18, L21, L34$ erkennen wir in den Tabellen 4.16 – 4.19 und Abb. 4.4, dass die quadratische Exponentialfunktion den schlechtesten Gesamtabstand liefert. Hier würde die Gamma-Exponentialfunktion die geeignetste Wahl für die Kovarianz-Funktion sein.

Allerdings erkennen wir auch, dass die Gesamtabstände aller Gauß-Prozess-Regressionen berechnet mit den zufällig gewählten Testdaten größere Werte liefern, als die Gauß-Prozess-Regressionen mit den in Abschnitt 1 berechneten Testdaten:

$$L08, L09, L14, L19, L21, L25, L26, L33.$$

Einen Vergleich dazu liefert Tabelle 4.20. Deshalb werden wir im weiteren Verlauf die berechneten Testdaten beibehalten und nur mehr die quadratische Exponentialfunktion als Kovarianz-Funktion betrachten.

Gamma-Exponentialfunktion				
Lastpunkt	**Geschätzter Wert**	**Gemessener Wert**	**Absoluter Fehler**	**Relativer Fehler**
L08	1762.31	2069.70	307.39	14.85%
L09	1588.97	2180.98	592.02	27.14%
L14	1252.48	566.04	686.44	121.27%
L19	2444.13	2485.68	41.55	1.67%
L21	2115.62	3084.27	968.66	31.41%
L25	2764.05	2664.97	99.08	3.72%
L26	1027.65	610.07	417.58	68.45%
L33	1502.04	1723.28	221.25	12.84%
Gesamtabstand $(l = 1, \gamma = 1, \sigma_n = 0)$			3333.96	

Tabelle 4.14: Optimale Testdaten mit Gamma-Exponentialfunktion.

Matern Funktion mit $\nu = p + \frac{1}{2}$				
Lastpunkt	**Geschätzter Wert**	**Gemessener Wert**	**Absoluter Fehler**	**Relativer Fehler**
L08	2063.34	2069.70	6.36	0.31%
L09	1860.57	2180.98	320.41	14.69%
L14	1551.21	566.04	985.17	174.04%
L19	2417.53	2485.68	68.14	2.74%
L21	2134.79	3084.27	949.48	30.78%
L25	2590.49	2664.97	74.48	2.79%
L26	1426.05	610.07	815.98	133.75%
L33	1926.77	1723.28	203.48	11.81%
Gesamtabstand $(l = 1, p = 1, \sigma_n = 0)$			3423.51	

Tabelle 4.15: Optimale Testdaten mit Matern Funktion mit $\nu = p + \frac{1}{2}$.

3 Plausible Wahl der Hyperparameter

Bis jetzt haben wir herausgefunden, dass die quadratische Exponentialfunktion am plausibelsten für unsere gewählten Testdaten (4.2) erscheint, wenn wir die Hyperparameter der Kovarianz-Funktionen nicht optimiert haben. Das Ergebnis dieser Gauß-Prozess-Regression (Tabelle 4.12) wollen wir nun weiter verbessern, indem wir passende Werte für die Hyperparameter der quadratischen Exponentialfunktion bestimmen.

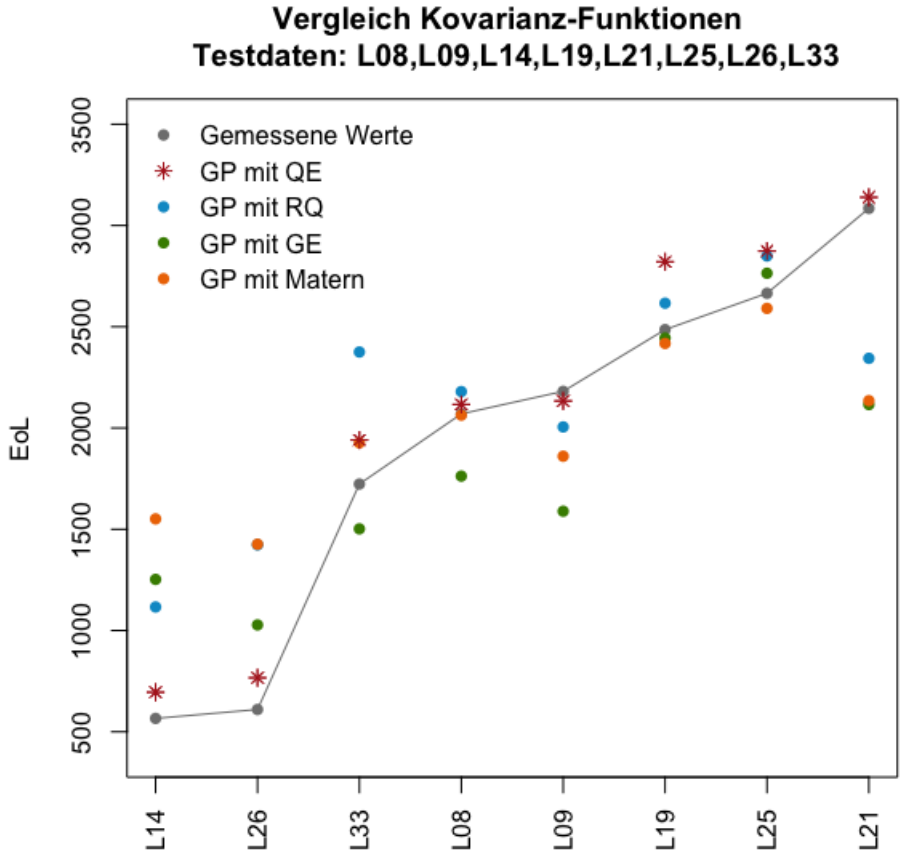

Abbildung 4.3: Vergleich Kovarianz-Funktionen ohne Optimierung der Hyperparameter.

Quadratische Exponentialfunktion				
Lastpunkt	**Geschätzter Wert**	**Gemessener Wert**	**Absoluter Fehler**	**Relativer Fehler**
C03	1561.72	2594.36	1032.64	39.80%
C04	777.91	2558.36	1780.45	69.59%
L01	2398.00	3825.00	1426.99	37.31%
L11	−214.23	814.59	1028.82	126.30%
L12	155.13	3085.36	2930.22	94.97%
L18	337.02	552.57	215.56	39.01%
L21	2123.28	3084.27	960.99	31.16%
L34	1896.64	467.64	1429.00	305.58%
Gesamtabstand $(l = 1, \sigma_f = 1, \sigma_n = 0)$			10804.67	

Tabelle 4.16: Zufällige Testdaten mit quadratischer Exponentialfunktion.

Rationale quadratische Exponentialfunktion				
Lastpunkt	**Geschätzter Wert**	**Gemessener Wert**	**Absoluter Fehler**	**Relativer Fehler**
C03	2055.93	2594.36	538.43	26.19%
C04	1524.28	2558.36	1034.08	67.84%
L01	2553.03	3825.00	1271.96	49.82%
L11	486.45	814.59	328.14	67.45%
L12	1581.69	3085.36	1503.66	95.07%
L18	368.26	552.57	184.31	50.05%
L21	1963.67	3084.27	1120.61	57.07%
L34	1780.13	467.64	1312.49	73.73%
Gesamtabstand $(l = 1, \alpha = 1, \sigma_n = 0)$			7293.682	

Tabelle 4.17: Zufällige Testdaten mit rational quadratischer Exponentialfunktion.

Gamma-Exponentialfunktion				
Lastpunkt	**Geschätzter Wert**	**Gemessener Wert**	**Absoluter Fehler**	**Relativer Fehler**
C03	2682.70	2594.36	88.34	3.41%
C04	2370.77	2558.36	187.59	7.33%
L01	1970.28	3825.00	1854.71	48.49%
L11	1330.60	814.59	516.01	63.35%
L12	2387.11	3085.36	698.24	22.63%
L18	577.51	552.57	24.94	4.51%
L21	2096.10	3084.27	988.17	32.04%
L34	1456.91	467.64	989.27	211.54%
Gesamtabstand $(l = 1, \gamma = 1, \sigma_n = 0)$			5347.264	

Tabelle 4.18: Zufällige Testdaten mit Gamma-Exponentialfunktion.

Matern Funktion mit $\nu = p + \frac{1}{2}$				
Lastpunkt	**Geschätzter Wert**	**Gemessener Wert**	**Absoluter Fehler**	**Relativer Fehler**
C03	2687.92	2594.36	93.56	3.61
C04	2447.83	2558.36	110.53	4.32%
L01	2084.96	3825.00	1740.03	45.49%
L11	1586.42	814.59	771.83	94.75%
L12	2621.27	3085.36	464.09	15.04%
L18	980.77	552.57	428.19	77.49%
L21	2160.66	3084.27	923.62	29.95%
L34	1707.96	467.64	1240.32	265.23%
Gesamtabstand $(l = 1, p = 1, \sigma_n = 0)$			5772.17	

Tabelle 4.19: Zufällige Testdaten mit Matern Funktion mit $\nu = p + \frac{1}{2}$.

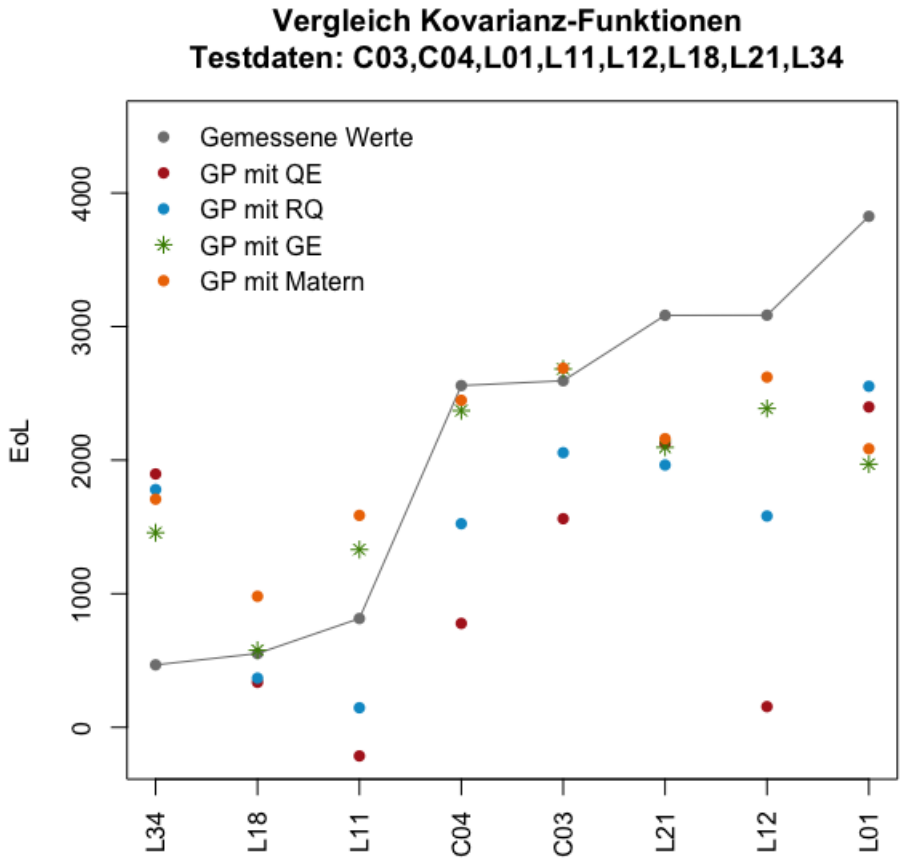

Abbildung 4.4: Vergleich Kovarianz-Funktionen ohne Optimierung der Hyperparameter.

	Quadratische Exponential	**Rational Exponential**	**Gamma Exponential**	**Matern**
Optimale Testdaten	**1195.26**	3356.10	3333.96	3423.51
Zufällige Testdaten	10804.67	7293.682	**5347.264**	5772.17

Tabelle 4.20: Vergleich Gesamtabstände für verschiedene Funktionen und Testdaten.

Ein Test von verschiedenen Hyperparametern (siehe Kapitel 3, Unterabschnitt 4.1) hat gezeigt, dass folgende Hyperparameter geeignet sind:

$$l = 1,\ \sigma_f = 20,\ \sigma_n = 0.8. \tag{4.3}$$

Mit dieser Wahl der Hyperparameter erhalten wir folgendes Ergebnis der Gauß-Prozess-Regression (Tabelle 4.21). Vergleichen wir den Gesamtabstand, sehen wir eine kleine Verbesserung (Tabelle 4.22).

Lastpunkt	**Geschätzter Wert**	**Gemessener Wert**	**Absoluter Fehler**	**Relativer Fehler**
L08	2100.65	2069.70	30.95	1.50%
L09	2109.90	2180.98	71.08	3.26%
L14	708.94	566.04	142.90	25.24%
L19	2800.30	2485.68	314.62	12.66%
L21	3082.85	3084.27	1.43	0.05%
L25	2843.75	2664.97	178.78	6.71%
L26	780.54	610.07	170.47	27.94%
L33	1913.49	1723.28	190.21	11.04%
Gesamtabstand $(l = 1, \sigma_f = 20, \sigma_n = 0.8)$			1100.432	

Tabelle 4.21: Gauß-Prozess-Regression mit getesteten Hyperparameter.

	Hyperparameter $l = 1, \sigma_f = 1, \sigma_n = 0$	**Hyperparameter** $l = 1, \sigma_f = 20, \sigma_n = 0.8$
Gesamtabstand	1195.26	$1100,432$

Tabelle 4.22: Vergleich Gauß-Prozess-Regression mit unterschiedlichen Hyperparameter.

Durch die Methode der Minimierung des Gesamtabstandes mit Hilfe der L_1-Norm (Kapitel 3, Unterabschnitt 4.5) und den verschiedenen Optimierungsverfahren (Kapitel 3, Unterabschnitt 4.2), können wir das Ergebnis der Gauß-Prozess-Regression verbessern. Als Startwerte wählen wir die durch das Testen erhaltenen Werte (4.3).

	Startwerte	**CG**	**BFGS**	**GenSA**	**DE**
l	1	0.95	0.96	0.96	**0.96**
σ_f	20	20.01	20.00	20.02	**947.65**
σ_n	0.8	0.73	0.80	0.82	**38.72**
Gesamtabstand		1014.28	1012.64	1011.34	**968.89**

Tabelle 4.23: Hyperparameterbestimmung mit verschiedenen Optimierungsverfahren.

Wie wir anhand Tabelle 4.23 und Abb. 4.5 erkennen, erhalten wie das beste Ergebnis, wenn wir den Gesamtabstand mit Hilfe der Differential Evolution Methode minimieren. Dies hat damit zu tun, dass die anderen Methoden Startwerte benötigen und deshalb auch

um diese Startwerte ihr Minimum finden. Die Differential Evolution Methode benötigt keine vorgegebenen Startwerte und findet so andere Hyperparameter, die noch einen geringeren Gesamtabstand liefern. Vor allem fallen die hohen Werte von σ_f und σ_n auf. Der Gesamtabstand konnte gegenüber der Standardeinstllung ($l = 1, \sigma_f = 1, \sigma : n = 0$) von 1195.26 auf 968.89 verbessert werden (siehe Tabelle 4.25). Das entspricht einer Verringerung um 19%.

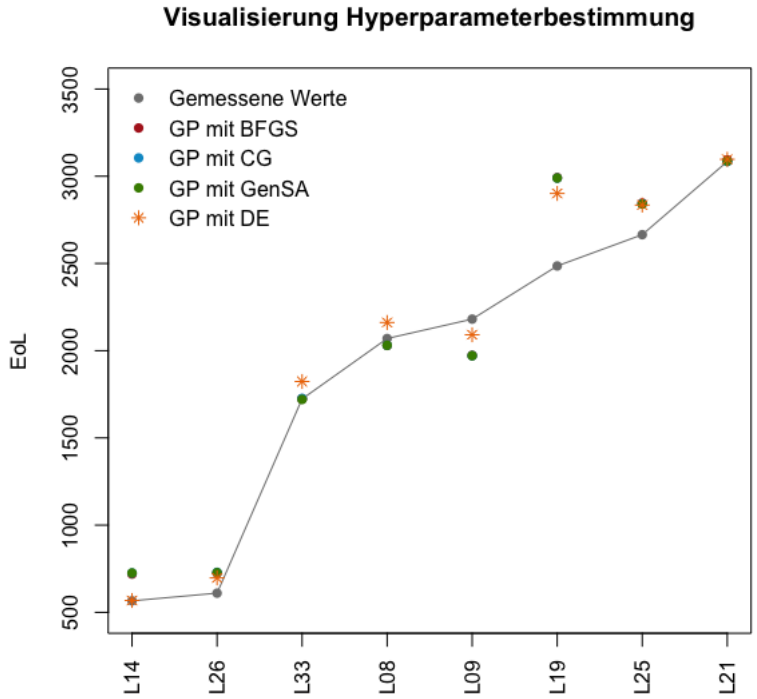

Abbildung 4.5: Hyperparameterbestimmung mit verschiedenen Optimierungsverfahren.

Da wir im vorherigen Abschnitt 2 unsere Kovarianz-Funktionen ohne optimierte Hyperparameter betrachtet haben, wollen wir nun überprüfen, ob die quadratische Exponentialfunktion auch mit optimierten Hyperparametern den geringsten Gesamtabstand gegenüber andere Kovarianz-Funktionen liefert. Dazu vergleichen wir wieder die Gesamtabstände der gewählten Kovarianz-Funktionen aus Abschnitt 2.

Wie wir anhand Tabelle 4.24 und Abb. 4.6 erkennen, erhalten wir eine deutliche Verbesserung gegenüber den Gesamtabständen ohne optimierten Hyperparameter (Vergleich Tabellen 4.12 - 4.15). Des Weiteren liefert weiterhin die quadratische Exponentialfunktion den geringsten Gesamtabstand. Deshalb werden wir in den folgenden Abschnitten die quadratische Exponentialfunktion als Kovarianz-Funktion beibehalten und ihre Hyperparameter folgendermaßen festlegen:

$$l = 0.955, \ \sigma_f = 947.652, \ \sigma_n = 38.724. \tag{4.4}$$

Kovarianz-Funktion	**Gesamtabstand**
QE mit $l = 0.955, \sigma_f = 947.652, \sigma_n = 38.724$	**968.895**
RQ mit $l = 0.955, \alpha = 968.327, \sigma_n = 0.0381$	1012.055
GE mit $l = 1.351, \gamma = 2.000, \sigma_n = 0.039$	1010.002
Matern mit $l = 0.8, p = 59.5, \sigma_n = 0$	3070.616

Tabelle 4.24: Vergleich Kovarianz-Funktionen mit optimierten Hyperparametern.

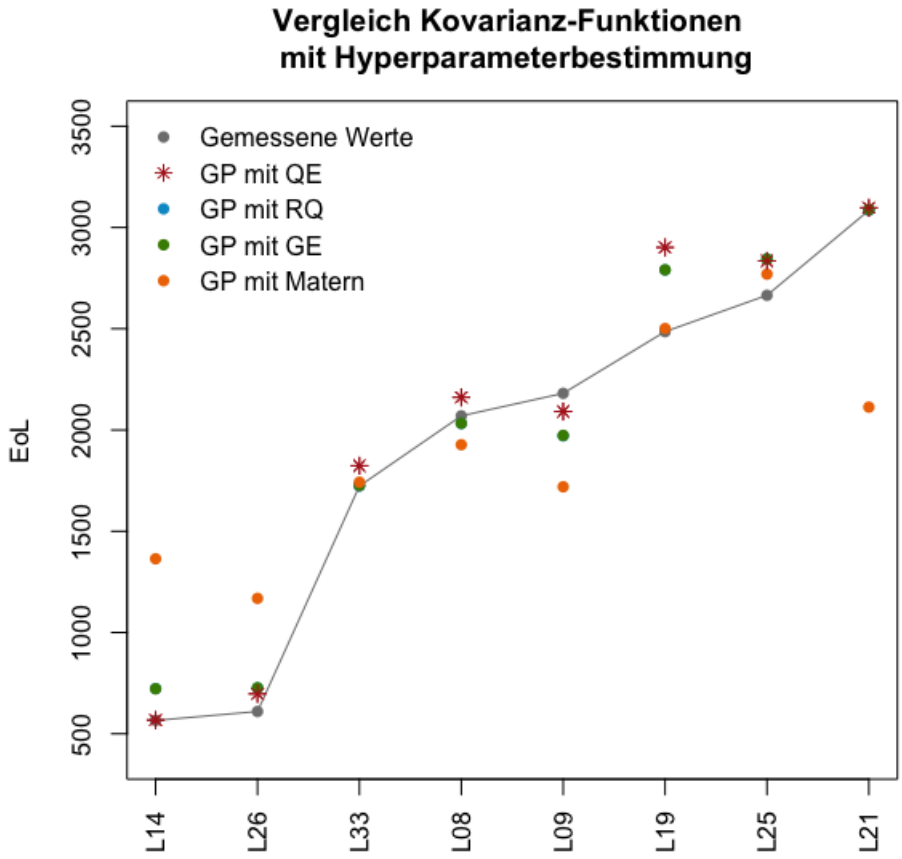

Abbildung 4.6: Vergleich Kovarianz-Funktionen mit optimierten Hyperparametern.

Lastpunkt	**Geschätzter Wert**	**Gemessener Wert**	**Absoluter Fehler**	**Relativer Fehler**
L08	2161.44	2069.70	91.74	4.43%
L09	2091.30	2180.98	89.68	4.11%
L14	567.92	566.04	1.88	0.33%
L19	2901.40	2485.68	415.72	16.72%
L21	3097.03	3084.27	12.75	0.41%
L25	2834.77	2664.97	169.80	6.37%
L26	697.34	610.07	87.27	14.31%
L33	1823.33	1723.28	100.04	5.81%
Gesamtabstand ($l = 0.955, \sigma_f = 947.652, \sigma_n = 38.724$)				968.895

Tabelle 4.25: Gauß-Prozess-Regression mit Hyperparameterbestimmung durch DE.

4 Plausible Wahl der Erwartungswert-Funktion

Um das Ergebnis der bisherigen Gauß-Prozess-Regression weiter zu verbessern, wollen wir mit Hilfe des in Kapitel 3 vorgestellten Simple Kriging (Abschnitt 5.1) eine fixe Erwartungswert-Funktion in die Berechnung miteinbeziehen. Wie vorgeschlagen, wollen wir ein lineares Regressionsmodell als Erwartungswert-Funktion betrachten.

Die linearen Modelle für *ALICe 1* Daten wurden in der Arbeit von *G. Gößler* [Gößler, 2015] bestimmt. Dazu verwendet *G. Gößler* zunächst die `R`-Funktion `leaps` um für alle Kombinationen der Faktoren lineare Regressionen durchzuführen. Anschließend werden mit Hilfe von verschiedenen statistischen Kriterien wie z.B. *BIC*, *adjustiertes* R^2, jene Modelle gesucht, die zum Beschreiben der EoL-Werte am geeignetsten erscheinen. Genauere Details zu dieser Modellsuche können in der Arbeit von *G. Gößler* [Gößler, 2015] nachgelesen werden.

Am Ende dieser Modellsuche stellen sich folgende vier Modelle als geeignet heraus:

$$
\begin{aligned}
lm_6 :&\ EoL \sim T + SoC + I(T^2) + I((ADC)^2) + I(SoC^2) + SoC : dSoC. \\
lm_7 :&\ EoL \sim T + SoC + dSoC + I(T^2) + I((ADC)^2) + I(SoC^2) + SoC : dSoC. \\
lm_{10} :&\ EoL \sim T + CC + ADC + SoC + dSoC + I(T^2) + I((ADC)^2) + I(SoC^2) \\
&\quad + CC : dSoC + SoC : dSoC. \\
lm_{11} :&\ EoL \sim T + CC + ADC + SoC + dSoC + I(T^2) + I((ADC)^2) + I(SoC^2) \\
&\quad + CC : PDC + CC : dSoC + SoC : dSoC.
\end{aligned} \tag{4.5}
$$

Mit Hilfe der `R`-Funktion `summary` können wir überprüfen, ob die gewählten Modelle geeignet sind.

```
Call:
lm(formula = alice1_mean$EoL ~ alice1_mean$T + alice1_mean$SoC +
    I(alice1_mean$T^2) + I((alice1_mean$ADC)^2) + I(alice1_mean$SoC^2) +
    alice1_mean$SoC:alice1_mean$dSoC)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-1955.83  -292.40    13.36   402.70  1054.50

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                         1693.6      369.1   4.588 5.84e-05 ***
alice1_mean$T                       -835.7      139.8  -5.977 9.22e-07 ***
alice1_mean$SoC                     2795.3     1031.0   2.711 0.010436 *
I(alice1_mean$T^2)                 -1053.8      264.5  -3.984 0.000339 ***
I((alice1_mean$ADC)^2)               880.5      412.9   2.133 0.040255 *
I(alice1_mean$SoC^2)                1612.4      357.0   4.516 7.22e-05 ***
alice1_mean$SoC:alice1_mean$dSoC    4374.4     1159.7   3.772 0.000619 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '  1

Residual standard error: 702.4 on 34 degrees of freedom
Multiple R-squared:  0.8037,Adjusted R-squared:  0.769
F-statistic:  23.2 on 6 and 34 DF,  p-value: 1.086e-10
```

Wie wir erkennen, sind alle im Modell lm_6 gewählten Faktoren signifikant. Dieses Ergebnis erhalten wir analog für die anderen Modelle.

Da wir geeignete lineare Modelle gefunden haben, die wir als Erwartungswert-Funktionen verwenden können, wollen wir herausfinden, welche Erwartungswert-Funktion am geeignetsten für unsere Gauß-Prozess-Regression ist. Dazu führen wir mit jeder Erwartungswert-Funktion eine Gauß-Prozess-Regression durch. Als Kovarianz-Funktion wählen wir die in Kapitel 4,Abschnitt 2, bestimmte quadratische Exponentialfunktion und als Hyperparameter, die in Kapitel 4, Abschnitt 3 berechneten Werte (4.4). Anschließend werden wieder die Gesamtabstände der interpolierten und gemessenen Testpunkte verglichen.

	lm_6	lm_7	lm_{10}	lm_{11}
Gesamtabstand	**751.70**	847.712	762.823	802.810

Tabelle 4.26: Vergleich Gesamtabstände für verschiedene Erwartungswert-Funktionen.

Wie wir in Tabelle 4.26 und Abb. 4.7 erkennen, erhalten wir das beste Ergebnis, wenn wir das Modell lm_6 als Erwartungs-Funktion auswählen. Das Hinzufügen der Erwartungswert-Funktion liefert einen deutlich geringeren Gesamtabstand (vgl. Tabelle 4.27), als bei einer Erwartungswert-Funktion gleich 0. Der Gesamtabstand hat sich von 968.89 auf 751.70 verringert. Das entspricht einer Verringerung um 22%.

Lastpunkt	Geschätzter Wert	Gemessener Wert	Absoluter Fehler	Relativer Fehler
L08	2044.80	2069.70	24.90	1.20%
L09	2183.35	2180.98	2.37	0.11%
L14	716.02	566.04	149.98	26.50%
L19	2746.30	2485.68	260.62	10.48%
L21	3109.75	3084.27	25.47	0.83%
L25	2840.72	2664.97	175.75	6.59%
L26	678.48	610.07	68.42	11.21%
L33	1679.09	1723.28	44.19	2.56%
Gesamtabstand $(l = 0.955, \sigma_f = 947.652, \sigma_n = 38.724)$				751.70

Tabelle 4.27: Gauß-Prozess-Regression mit lm_6 als Erwartungswert-Funktion.

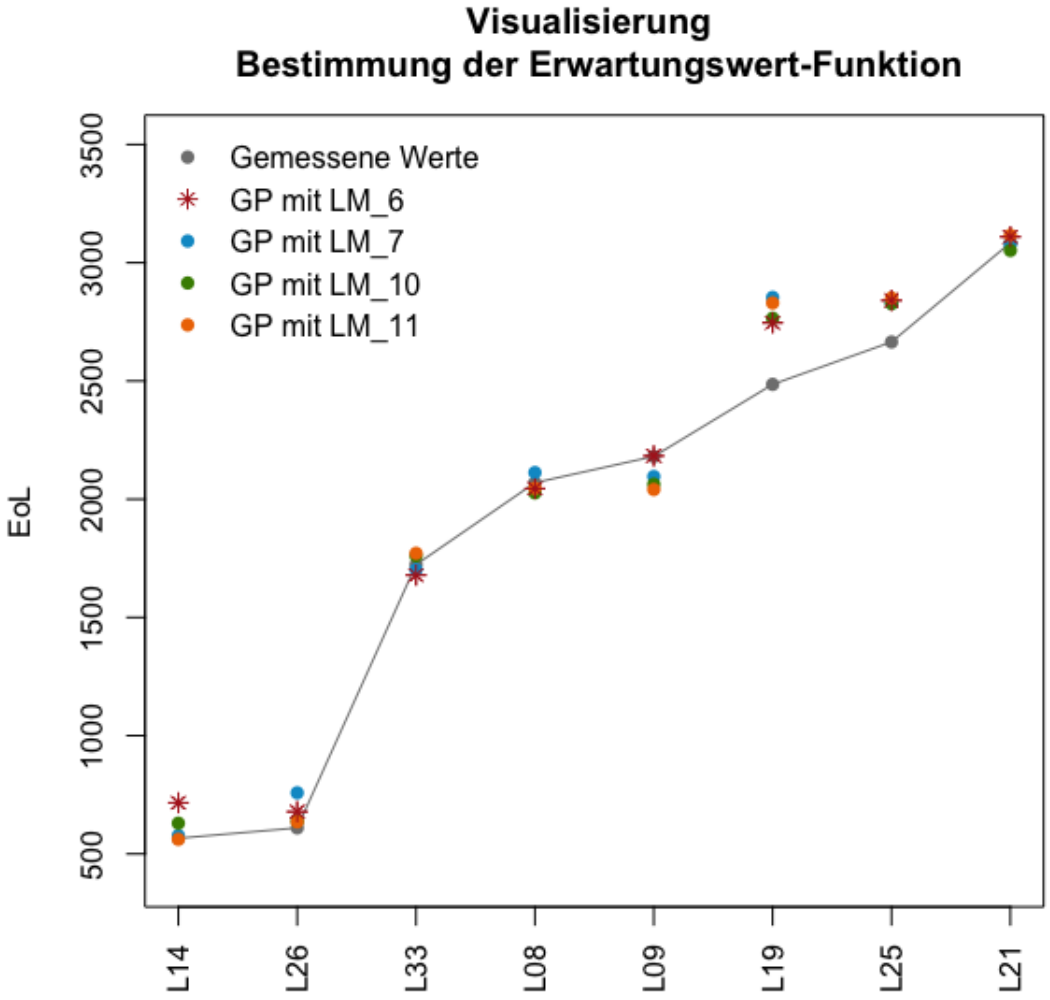

Abbildung 4.7: Bestimmung Erwartungswert-Funktion.

5 Vergleich lineare Regression und Gauß-Prozess-Regression

Da die Gauß-Prozess-Regression gute Ergebnisse für unsere *ALICe1* Daten liefert, wollen wir in diesem Abschnitt diese Ergebnisse mit jenen Ergebnissen vergleichen, die wir erhalten, falls wir eine gewöhnliche lineare Regression für die Vorhersage der EoL-Werte nutzen.

Um diesen Vergleich durchführen zu können, müssen wir zunächst die EoL-Werte der im Kapitel 4, Abschnitt 1 berechneten Testdaten:

$$L08, L09, L14, L19, L21, L25, L26, L33$$

mit Hilfe von linearen Modellen vorhersagen. Dazu verwenden wir die im vorherigen Abschnitt eingeführten Modelle (4.5) und die R-Funktion `predict`. Wichtig ist zu erwähnen, dass auch bei dieser Vorhersage wie bei der Gauß-Prozess-Regression dieselben Trainingsdaten für die Schätzung der Testdaten verwendet werden.

```
> lm6_bal <- lm(alice1_mean$EoL~alice1_mean$T+ alice1_mean$SoC +
+ I(alice1_mean$T^2) + I((alice1_mean$ADC)^2) + I(alice1_mean$SoC^2) +
+ alice1_mean$SoC:alice1_mean$dSoC )
> lm6_bal_predict <- predict(lm6_bal, interval = "confidence", level=0.95)
> lm6_bal_predict[c(14,15,20,25,27,31,32,39),]
         fit        lwr      upr
14 1783.8713 1222.48165 2345.261
15 2167.6201 1584.41941 2750.821
20  885.9336  197.03632 1574.831
25 2519.5572 2048.10830 2991.006
27 3321.4903 2676.23551 3966.745
31 2270.1768 1798.02445 2742.329
32  598.8184   68.59742 1129.039
39 1876.6875 1366.88865 2386.486
```

Für die anderen Modelle geschieht diese Berechnung analog. Anschließend werden die geschätzen Werte der linearen Modelle mit den geschätzen Werte der Gauß-Prozess-Regression verglichen. Dazu betrachten wir wieder die jeweiligen Gesamtabstände gemäß L_1-Norm zwischen interpolierten und gemessenen Werten.

In Tabelle 4.28 erkennen wir, dass die Gauß-Prozess-Regression einen deutlich geringeren Gesamtabstand liefert als die der linearen Regression. Doch wenn wir die einzelnen Lastpunkte betrachten, sehen wir in Abb. 4.8, dass vor allem die Lastpunkte $L26$ und $L09$ mit Hilfe der linearen Regression lm_6 besser getroffen werden.

Jetzt stellt sich die Frage: Welchen zeitlichen Aufwand müssen wir für die Gauß-Regression aufbringen und würde sich dieser Mehraufwand lohnen? Dazu gibt die Tabelle 4.29 Aufschluss. Als Voraussetzung ist anzunehmen, dass für alle Berechnungen die Daten in der jeweiligen Form und alle benötigten Programm-Skripte vorhanden sind. Außerdem wird bei der Gauß-Prozess-Regression angenommen, dass die linearen Modelle bekannt sind, die wir für die Erwartungswert-Funktion benötigen.

Lastpunkt	**Gemessener Wert**	GPR	lm_6	lm_7	lm_{10}	lm_{11}
L08	2074.25	**2044.80**	1783.87	1808.42	1642.02	1864.80
L09	2170.20	**2183.35**	2167.62	2202.03	2117.80	2249.77
L14	666.66	**716.02**	885.93	897.25	861.72	890.92
L19	2828.78	**2746.30**	2519.56	2548.06	2496.58	2682.66
L21	3082.65	**3109.75**	3321.49	3318.65	3309.31	3121.34
L25	2846.28	**2840.72**	2270.18	2228.58	2364.94	2270.33
L26	602.45	**678.48**	598.82	512.73	845.18	796.86
L33	1640.63	**1679.09**	1876.69	1833.71	1785.28	1515.68
Gesamtabstand		751.70	1449.62	1554.45	1619.61	1621.65

Tabelle 4.28: Gauß-Prozess-Regression und multiple lineare Regression.

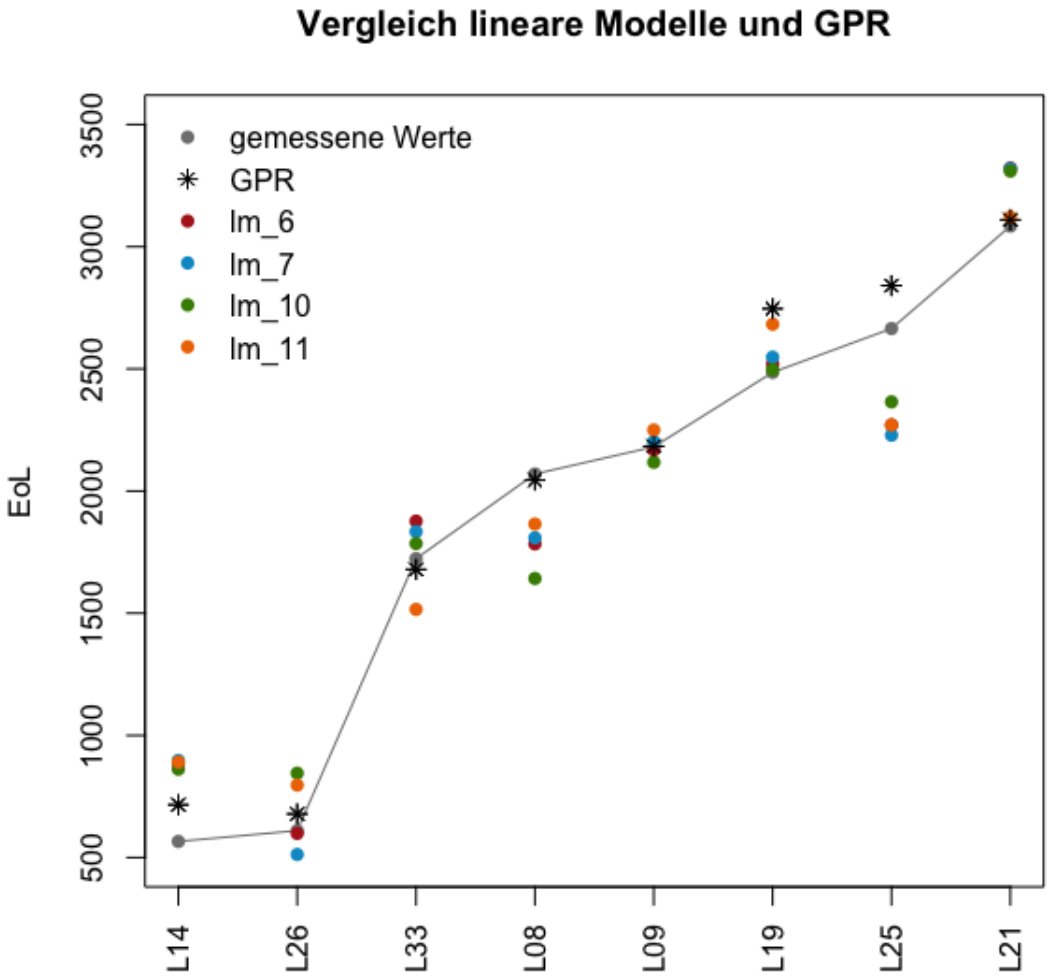

Abbildung 4.8: Vergleich Gauß-Prozess-Regression und lineare Regression.

Wie wir erkennen, benötigen wir für die Gauß-Prozess-Regression deutlich mehr Schritte und Zeit, als für eine lineare Regression. Vor allem das Finden von plausiblen Hyperparametern kostet einiges an Zeit. Des Weiteren ist die Gauß-Prozess-Regression stark von den gewählten Test- und Trainingsdaten abhängig. Sobald wir andere Testdaten als unsere in Kapitel 4, Abschnitt 1 berechneten Testdaten (4.2), wählen, erhalten wir deutlich schlechtere Ergebnisse als mit Hilfe der linearen Regression.

Dadurch, dass die lineare Regression keine wesentlich größeren Gesamtabstände liefert als die Gesamtabstände der Gauß-Prozess-Regression, ist es fraglich, ob sich der Mehraufwand wirtschaftlich für das Modellieren der Alterung von Lithium-Ionen-Zellen lohnt.

6 Zusammenfassung

In diesem Abschnitt fassen wir die Methode der Gauß-Prozess-Regression, um Lebensdauern von Lithium-Ionen-Zellen zu modellieren, kurz zusammenfassen.

Da es in unserem Fall keine Vorgaben gibt, welche Lastpunkte der *ALICe 1* Daten als Testdaten $\boldsymbol{x_*}$ und welche als Trainingsdaten $\boldsymbol{x}$ verwendet werden sollen, wählen wir für den Anfang acht, zufällige Lastpunkte als Testdaten aus (Abschnitt 1). Mit diesen zufälligen Testdaten führen wir mit verschiedenen Kovarianz-Funktionen eine Gauß-Prozess-Regression durch. Dabei verwenden wir Kovarianz-Funktionen deren Hyperparameter noch nicht optimiert wurden. Die Erwartungswert-Funktion wird auf 0 gesetzt. Durch diese Versuche mit zufälligen Testdaten erkennen wir, dass vor allem mit Hilfe der quadratischen Exponentialfunktion die geringsten Gesamtabstände erreicht werden. Dies liefert uns den Anlass dafür, dass wir mit Hilfe einer Kreuzvalidierung und der quadratischen Exponentialfunktion jene 8 Testdaten bestimmen wollen, die bis jetzt den geringsten Gesamtabstand liefern.

Nach der optimalen Wahl der Testdaten: $L08, L09, L14, L19, L21, L25, L26, L33$ überprüfen wir, ob die quadratische Exponentialfunktion ohne Optimierung der Hyperparameter weiterhin den geringsten Gesamtabstand bezüglich der L_1-Norm liefert. Dazu vergleichen wir die Gesamtabstände der verschiedenen Kovarianz-Funktionen miteinander (Abschnitt 2).

Nachdem bestätigt wurde, dass weiterhin die quadratische Exponentialfunktion den geringsten Gesamtabstand liefert, optimieren wir die Hyperparameter dieser Funktion mit verschiedenen Optimierungsverfahren. (Abschnitt 3). Auch ein weiterer Vergleich von verschiedenen Kovarianz-Funktionen mit optimierten Hyperparametern verändert nichts an dem Ergebnis, dass die quadratische Exponentialfunktion mit den Hyperparametern $l = 0.955,\ \sigma_f = 947.652,\ \sigma_n = 38.724$ am geeignetsten für die Simulation der Lebensdauern ist.

Im letzten Schritt bestimmen wir die Erwartungswert-Funktion des Gauß-Prozesses (Abschnitt 4). Dazu wählen wir verschiedene lineare Modelle, die im Laufe des *K2 Projektes E3T3 (ALICe)* [Cifrain, 2015] und der Arbeit von *G. Gößler* [Gößler, 2015] erstellt worden sind und führen eine Gauß-Prozess-Regression durch. Den geringsten Gesamtabstand erhalten wir, indem wir das Modell lm_6 als Erwartungswert-Funktion wählen.

Am Ende der Prozedur erhalten wir einen Gesamtabstand zwischen interpolierten und extrapolierten Punkten von 751.70. Ein Vergleich mit den Ergebnissen einer gewöhnlichen linearen Regression (Abschnitt 5) zeigt uns zwar, dass wir mit Hilfe einer Gauß-Prozess-Regression einen geringeren Gesamtabstand erhalten, als mit einer linearen Regression, allerdings ist der Aufwand bei einer Gauß-Prozess-Regression deutlich höher als bei einer linearen Regression.

Gauß-Prozess-Regression	**Lineare Regression**
Suchen geeigneter Test- und Trainingsdaten: Setzen wir immer 2 Parameter variabel, benötigen wir für jede Berechnung ca. 5 Minuten. **Insgesamt:** $0.5 - 1$ Std.	**Bestimmung linearer Modelle:** Mit Hilfe der `R`-Funktion `leaps` bekommen wir einen Überblick über die möglichen Modelle. Mit Hilfe von verschiedenen statistischen Kriterien wie *BIC* und *adjustiertes* R^2 werden die geeignetsten Modelle herausgefunden. **Insgesamt:** 8 Std.
Wahl Kovarianz-Funktion: Gauß-Prozess-Regression wird mit verschiedenen Kovarianz-Funktionen ausgeführt und anschließend werden ihre Gesamtabstände miteinander verglichen. **Insgesamt:** 0.25 Std.	**Berechnung der geschätzten Werte:** Mit Hilfe der `R`-Funktion `predict`, bekommen wir für jedes Modell die geschätzen EoL-Werte. **Insgesamt:** 0.25 Std.
Wahl Hyperparameter: Das Testen von Hyperparametern liefert uns gleichzeitig eine Startlösung für die einzelnen Optimierungsverfahren.Die Berechnung dieser Startlösungen dauert je nach Intervalllänge 8-48 Stunden. Die einzelnen Optimierungsverfahren haben verschiedene Berechnungsdauern. *BFGS*- und *CG*- Verfahren benötigen ca. 3 Minuten pro Startlösung. *GenSA* und *DE* Methode ca. 15 Minuten pro Startlösung. Da immer mehrere Startlösungen getestet werden sollen, um ein Hängen bleiben in einer lokalen Lösung zu vermeiden, verlängert sich die Berechnungsdauer je nach Anzahl der Startlösungen. **Insgesamt:** maximal 50 Std.	
Wahl Erwartungswert-Funktion: Wie bei der geeigneten Wahl der Kovarianz-Funktion werden für alle gegebenen Modelle Gauß-Prozess-Regressionen durchgeführt. Anschließend werden die Geamtabstände miteinander verglichen. **Insgesamt:** maximal 0.25 Std.	
Gesamtdauer: $\sim$ 52 Std.	**Gesamtdauer:** $\sim$ 8.25 Std.

Tabelle 4.29: Berechnungsdauer: Gauß-Prozess-Regression vs. lineare Regression.

Anhang A

Line-Search-Methode

Sowohl für das CG-Verfahren (Abschnitt 4.2) als auch für das BFGS-Verfahren (Abschnitt 4.2) benötigen wir für jede Iteration eine Schrittlänge α. Diese Schrittlänge α kann mit Hilfe einer sogenannten *Line-Search-Methode* [Pfeiffer, 2017] gefunden werden.

Sei dazu $\boldsymbol{x}_k$ das berechnete Minimum der Funktion f im $k-$ten Iterationsschritt. Weiters sei $\boldsymbol{p}_k$ die dazugehörige Richtung. Um einen passenden Wert für die Schrittlänge α zu erhalten, muss folgendes Subproblem gelöst werden:

$$\phi(\alpha) = f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k).$$

Es kann gezeigt werden, dass das Subproblem genau dann gelöst werden kann, falls α folgende zwei Regeln (A.1) und (A.2) erfüllt. Sei dazu $0 < c_1 < c_2 < 1$ gegeben.

- **Armijo-Regel**

$$f(\boldsymbol{x_{k+1}}) \leq f(\boldsymbol{x}_k) + c_1 \alpha \nabla f(\boldsymbol{x}_k)^T \boldsymbol{p}_k \Leftrightarrow \phi(\alpha_k) \leq \phi(0) + c_1 \alpha_k \phi'(0). \tag{A.1}$$

- **Wolfe-Regel**

$$\nabla f(\boldsymbol{x}_{k+1})^T \boldsymbol{p}_k \geq c_2 \nabla f(\boldsymbol{x}_k)^T \boldsymbol{p}_k \Leftrightarrow \phi(\alpha_k)' \geq c_2 \phi'(0). \tag{A.2}$$

Um eine geeignete Schrittlänge α zu erhalten suchen wir also ein α, dass beide Regeln erfüllt. Dazu kann folgender Line-Search-Algorithmus verwendet werden.

Algorithmus A.1 (**Line-Search-Methode für Schrittlängenbestimmung**).
Sei $\beta > 1$;
Setzte $\alpha = 1$;
if not Armijo- und Wolf-Regel für α erfüllt:
Setzte $\hat{\alpha_0} = \alpha$ und $k = 0$;
while Armijo-Regel erfüllt ist von α_k :

$$\hat{\alpha}_{k+1} = \beta \hat{\alpha}_k;$$
$$k = k + 1;$$

end (while)
Setze $\alpha_0^{min} = 0, \alpha_0^{max} = \hat{\alpha}_k, \alpha_0 = \frac{\alpha_0^{min} + \alpha_0^{max}}{2}$;
while not Armijo- und Wolfe-Regel für α_k erfüllt:
if not Armijo für α_k erfüllt:
Setze $\alpha_{k+1}^{min} = \alpha_k^{min}$ und $\alpha_{k+1}^{max} = \alpha_k$;
else
Setze $\alpha_{k+1}^{min} = \alpha_k$ und $\alpha_{k+1}^{max} = \alpha_k^{max}$;
end(if)
Setze $\alpha_{k+1} = \frac{\alpha_{k+1}^{min} + \alpha_{k+1}^{max}}{2}$;
Setzte $k = k + 1$;
end(while)
Setze $\alpha = \alpha_k$;
end(if)

Literaturverzeichnis

[Backhaus et al., 2006] Backhaus, K., Erichson, B., Plinke, W., and Weiber, R. (2006). *Multivariate Analysemethoden. Eine anwendungsorientierte Einführung.* Springer, Berlin.

[Böcker, 2004] Böcker, F. (2004). Statistik III. Vorlesungskript, Georg-August-Universiät Göttingen.

[Brain, 2006] Brain, M. (2006). How lithium-ion batteries work . `http://electronics.howstuffworks.com/everyday-tech/lithium-ion-battery1.htm`.

[Bronstein et al., 2001a] Bronstein, I., Semendjajew, K., Musiol, G., and Mühlig, H. (2001a). *Taschenbuch der Mathematik.* Harri Deutsch, Frankfurt am Main. (S. 771).

[Bronstein et al., 2001b] Bronstein, I., Semendjajew, K., Musiol, G., and Mühlig, H. (2001b). *Taschenbuch der Mathematik.* Harri Deutsch, Frankfurt am Main. (S.268; S. 624).

[Cifrain, 2015] Cifrain, M. (2015). ALICE, K2-E3T3, M36. Final project status report, Virtual Vehicle.

[Duvenaud, 2014] Duvenaud, D. K. (2014). *Automatic Model Construction with Gaussian Processes.* PhD thesis, University of Cambridge.

[Fleetwood, 2014] Fleetwood, K. (2014). An introduction to differential evolution. `http://www.maths.uq.edu.au/MASCOS/Multi-Agent04/Fleetwood.pdf`.

[Fletcher and Reeves, 1964] Fletcher, R. and Reeves, C. (1964). Function minimization by conjungate gradients. *The Computer Journal*, 7. S. 149 -154.

[Gößler, 2015] Gößler, G. (2015). Modelling the Lifespans of Lithium Ion Cells by using D-optimal Designs. Master's thesis, Graz University of Technology.

[Hestenes and Stiefel, 1952] Hestenes, M. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6). S.409 - 436.

[Hoff, 2009] Hoff, P. D. (2009). *A First Course in Bayesian Statistical Methods.* Springer, Berlin.

[Keirstead, 2012] Keirstead, J. (2012). Gaussian Processes R-Code . `https://www.r-bloggers.com/gaussian-process-regression-with-r/`.

[Kirkpatrick et al., 1983] Kirkpatrick, S., C. D. Gelatt, J., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220. S. 671-680.

[Klinz, 2015] Klinz, B. (2015). Operation Research. Vorlesungskript, Graz University of Technology.

[M.Cifrain, 2013] M.Cifrain (2013). M18Presentation. Presentation, Virtual Vehicle.

[Metropolis et al., 1953] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller., E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21. S.1087-1092.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Science, London.

[Mullen et al., 2011] Mullen, K. M., D.Ardia, D. L. Gil, D. W., and Cline, J. (2011). Deoptim: An r package for global optimization by differential evolution. *Journal of Statistical Software*, 40. S.1-26.

[Nilsson, 1998] Nilsson, N. J. (1998). *Introduction to Machine Learning*. Stanford University.

[Nocedal and Wright, 1999] Nocedal, J. and Wright, S. J. (1999). *Numerical Optimization*. Springer, New York.

[Pfeiffer, 2017] Pfeiffer, L. (2017). Nonlinear Optimisation. Vorlesungskript, Karl-Franzens Universität Graz.

[Plett, 2015] Plett, G. L. (2015). *Battery Modeling*. Artech House, Boston.

[Pregartner, 2012] Pregartner, G. (2012). Design-of-Experiment and Statistical Modeling of Large Scale Lithium Ion Cells. Master's thesis, Graz University of Technology.

[Rasmussen and Williams, 1996] Rasmussen, C. E. and Williams, C. K. I. (1996). *Gaussian Processes for Regression*. Massachusetts Institute of Technology.

[Rasmussen and Williams, 2006] Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. Massachusetts Institute of Technology.

[Rosenbrock, 1960] Rosenbrock, H. H. (1960). An automatic method for finding the greatest or least value of a function. *Computer Journal*, 3. S.175-184.

[Stadlober, 2015] Stadlober, E. (2015). Angewandte Statistik. Vorlesungskript, Graz University of Technology.

[Storn and K.Price, 1997] Storn, R. and K.Price (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11. S.341-359.

[Tsallis and Stariolo, 1996] Tsallis, C. and Stariolo, D. A. (1996). Generalized simulated annealing. *JPhysica A*. S.395-406.

[Vogel, 2013] Vogel, D. (2013). Multivariate Verfahren. Vorlesungskript, Technische Universität Dortmund.

[Warner, 2015] Warner, J. (2015). *The Handbook of Lithium-Ion Battery Pack Design.* Elsevier, Amsterdam.

[Xiang et al., 2013] Xiang, Y., Gubian, S., Suomela, B., and Hoeng, J. (2013). Generalized simulated annealing for global optimization: The gensa package. *The R Journal*, 5/1. S.13-28.