Universität Passau

Fakultät für Informatik und Mathematik

Bachelorarbeit

Evaluierung experimenteller Designs zur Bestimmung des nichtfunktionalen Einflusses metrischer Konfigurationsoptionen

Autor:

Martin Bochenek

07. April, 2014

Betreuer:

Prof. Dr.-Ing. Sven Apel

Lehrstuhl für Informatik mit Schwerpunkt Softwareproduktlinien

Dr.-Ing. Norbert Siegmund

Alexander Grebhahn

Bochenek, Martin:
Evaluierung experimenteller Designs zur Bestimmung des nichtfunktionalen Einflusses metrischer Konfigurationsoptionen
Bachelorarbeit, Universität Passau, 2014.

Inhaltsangabe

Moderne konfigurierbare Softwaresysteme sind multifunktional und variantenreich und unterscheiden sich für gewöhnlich maßgeblich in ihren funktionalen und nichtfunktionalen Eigenschaften. Die Wahl der für einen Benutzer, hinsichtlich einer nichtfunktionalen Eigenschaft, optimalen Variante erweist sich aufgrund der unzähligen Möglichkeiten dabei häufig als problematisch.
Die vorliegende Arbeit setzt sich daher mit der Anwendung experimenteller Designs im Bereich der konfigurierbaren Systeme auseinander, um die Auswahl durch die mithilfe der Versuchspläne gewonnenen Modelle zu erleichtern.
Zu diesem Zweck wird zunächst ein kurzer Überblick über die Begrifflichkeiten und Eigenschaften konfigurierbarer Systemen und der statistischen Versuchsplanung im Allgemeinen geschaffen. Auf Basis dessen werden die gängigsten Methoden der statistischen Versuchsplanung identifiziert und deren Ideen und Funktionsweisen erarbeitet und präsentiert.
Ausgewählte Vertreter der experimentellen Designs werden anschließend implementiert und in „SPL Conqueror“ integriert. Der „NFPMsimulator“, ein für diese Aufgabe entwickelter Simulator für nichtfunktionale Eigenschaften, ermöglicht die nachfolgende performante Evaluierung der experimentellen Designs im Bezug auf die Eignung für den gewählten Einsatzbereich.
Die abschließende Bewertung der durchgeführten Experimente fördert einige Einschränkungen bezüglich der Validität des gewählten Ansatzes und der verwendeten Simulationsmodelle zu Tage. Dies verhindert die eindeutige Bestimmung des „besten“ Designs im Rahmen dieser Arbeit, liefert jedoch Anknüpfungspunkte für bessere Modelle und zukünftige Arbeiten.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

1. Einführung

Fortschritte in der Softwareentwicklung ermöglichen es, immer komplexere und im Bezug auf die bereitgestellte Funktionalität auch umfangreichere Softwareprodukte zu erstellen.
Eine Vielzahl verfügbarer Hardware-Systeme mit ihren einhergehenden limitierten Ressourcen sowie persönliche Ansprüche eines Benutzers an die Funktionen eines Programms verlangen dabei zudem mehr Flexibilität. Dieser Variabilität der Anforderungen entwachsen aber unzählige potenzielle Varianten, deren individuelle Neuprogrammierung mehr als nur unwirtschaftlich wäre [BKPS04, Kapitel 1]. Es muss also sichergestellt werden, dass nicht jede Änderung der Nutzeransprüche den Aufwand einer Neuentwicklung mit sich bringt. Um diesen teils neuen Herausforderungen zu begegnen, werden konfigurierbare Systeme immer wichtiger.

Konfigurierbare Systeme erlauben es, durch das Anbieten von Konfigurationsoptionen eine, den individuellen Ansprüchen des Endnutzers entsprechende, Programmvariante zu erzeugen. Die genannten Ansprüche an das System können allerdings vielfältiger Natur sein und lassen sich grob in funktionale und nichtfunktionale Anforderungen gliedern [RR06, Kapitel 1].
Funktionale Anforderungen an ein Produkt äußern sich dabei in den von der Software bereitgestellten Funktionen. Nichtfunktionale Anforderungen werden in der Art und Weise, wie sich das Programm bei der Bereitstellung dieser Funktionen verhält, deutlich [RR06, Kapitel 1].
Die funktionalen Eigenschaften eines Programms sind damit beispielsweise mit dem Vorhandensein einer Loggingfunktionalität oder durch eine Maximalgröße für Eingabedateien definiert. Ob die gewählte Konfiguration aber auch einer gewünschten nichtfunktionalen Eigenschaft - wie einem geringen Arbeitsspeicherbedarf - entspricht, kann häufig nicht a priori festgestellt werden.

Während es also einem Anwender bei der Konfiguration eines Systems meist leicht fällt, die Auswahl der Konfigurationsoptionen bezüglich der funktionalen Anforderungen zu treffen, kann es oft schwierig sein die Programmvariation auszuwählen, welche auch die gewünschten nichtfunktionalen Anforderungen erfüllt. Dieser Umstand liegt in der Emergenz begründet, welcher eine nichtfunktionale Eigenschaft

eines Systems unter Umständen unterliegt. So ist zum Beispiel die Performanz eines Systems abhängig vom Zusammenspiel verschiedener Faktoren und erst bei Programmausführung bestimmbar. Nicht selten kommt es zudem vor, dass nur eine partielle Konfiguration auf Basis der Wunschfunktionen erstellt werden kann. Dies ist der Fall wenn der Nutzer sich nur für bestimmte Konfigurationsoptionen - wie zum Beispiel eine Verschlüsselung - entscheidet, er aber keine Kenntnis über die verbleibenden Optionen hat oder sich über deren Verwendung nicht im Klaren ist. Die Schwierigkeit besteht dann darin, aus den verbleibenden Konfigurationsmöglichkeiten die jeweils Beste im Hinblick auf eine nichtfunktionale Eigenschaft, wie beispielsweise die Performanz, zu selektieren.

Durch die Fülle an Konfigurationsoptionen, welche in Abschnitt 2.2.1 näher erklärt werden, wächst der Konfigurationsraum und damit die Menge unterschiedlicher Programmvarianten exponentiell. Eine Auswahl der für einen individuellen Anwender passenden Variante ist daher durch einfaches Ausprobieren der Möglichkeiten nicht mehr praktikabel. Das Finden einer für den spezifischen Anwendungsfall idealen Konfiguration wird zum Problem.

Um diesem Problem zu begegnen wurde durch Dr. Norbert Siegmund „SPL Conqueror“[1] entwickelt, ein Softwaretool zur Messung und Optimierung nichtfunktionaler Eigenschaften von Softwareproduktlinien und variablen Softwaresystemen [Sie]. Unterstützt werden bis dato nur boolesche Konfigurationsoptionen, um beispielsweise eine Auswahl der gewünschten Bestandteile des fertigen Produkts zuzulassen. Ebenso denkbar sind boolesche Konfigurationsoptionen in Form von Kompilierungsanweisungen, die die Anpassung an bestimmte Hardware zulassen. Die Realität der konfigurierbaren Systeme sieht allerdings an vielen Stellen auch metrische Optionen vor, um das Verhalten eines beinhalteten Softwarebestandteils zu beeinflussen. Der Begriff „metrisch“ weist hier auf Konfigurationsoptionen hin, deren Wertebereiche nach Stevens [Ste46] intervall- oder verhältnisskaliert sind. Beispiel hierfür ist das Setzen der maximalen Cachegröße eines Webbrowsers. Ebendiese Möglichkeit zur Konfiguration mittels metrischer Konfigurationsoptionen führt schlussendlich zur Existenz unzähliger verschiedener Programmvarianten.

Man steht hier einem Problem gegenüber, welches in vielen Bereichen der Wissenschaft omnipräsent ist. Das Problem liegt dabei in der Tatsache, dass die Anzahl der Programmvariaten exponentiell wächst und eine vollumfängliche Messung derer zu zeit- und kostenintensiv ist. Nicht zuletzt stößt die Untersuchung der Auswirkungen der unzähligen Konfigurationsoptionen auf ein System daher oft an die Grenzen des technisch machbaren.
Zu Lösung dieses Problems wurden in der Statistik schon früh Verfahren entwickelt, mit deren Hilfe die Wahl weniger, zur Modellbestimmung geeigneter Punkte aus dem Varianteraum möglich ist. Basierend auf den Messungen dieser Punkte können anschließend, unter Zuhilfenahme verschiedener mathematischer Werkzeuge, Modelle abgeleitet werden, die den gesamten Variantenraum beschreiben.
Diese Techniken sollen nun auf den Raum der einem konfigurierbaren System entwachsenden Varianten angewendet werden, um auch hier mit möglichst wenigen

[1] SPL Conqueror - A holostic approach to optimize software product lines and variants for non-functional properties - http://wwwiti.cs.uni-magdeburg.de/~nsiegmun/SPLConqueror/ (05.04.2014)

Messungen eine möglichst gute Vorhersagegenauigkeit zu erzielen. Zu diesem Zweck werden im Rahmen dieser Arbeit experimentelle Designs zur Auswahl der zu testenden Parameterkonfigurationen implementiert und evaluiert. Hauptaugenmerk der Evaluierung werden dabei die Anzahl der benötigten Messungen und die Güte der daraus resultierenden Vorhersagen sein. Ziel ist es schlussendlich, das Design zu bestimmen, dass unter dem Gesichtspunkt der Modellgüte am effektivsten arbeitet.

Zur Erreichung dieses Ziel wurde vorab ein Tool entwickelt (Abschnitt 4.3), welches die Simulation verschiedener Modelle erlaubt. Die Modelle definieren sich hierbei in der auf die Werte der metrischen Konfigurationsoptionen angewandten mathematischen Funktion. Ein Simulator dieser Art ermöglicht nun die einfache und performante Bewertung der Güte der experimentellen Designs.

2. Grundlagen

Bevor mit der Evaluierung begonnen werden kann, werden in diesem Kapitel grundlegende Begrifflichkeiten und Einführungen in wesentliche Aspekte der Arbeit erläutert. Zuerst wird auf die Herkunft des Begriffs „metrisch“ Bezug genommen sowie auf konfigurierbare Systeme und deren Eigenschaften näher eingegangen. Anschließend wird die grundlegende Funktionsweise von „SPL Conqueror“ kurz vorgestellt und zuletzt die statistische Versuchsplanung im Allgemeinen besprochen.

2.1 Theorie der Messskalen

Seit jeher liegt es in der Natur des Menschen, seine Beobachtungen und Erkenntnisse festhalten und wiedergeben zu wollen. Das Problem der „Messung“ wurde daher bereits in den 1930er Jahren, vor allem im Hinblick auf die quantitative Erfassung von Sinneseindrücken, ausführlich diskutiert [Ste46]. Die Messverfahren und die Bedeutung der Ergebnisse wurden anfangs als inhaltslos abgetan. Es stellte sich jedoch bald heraus, dass es sich eher um ein Problem semantischer Natur handelte [Ste46].

Eine Messung lässt sich im weitesten Sinne als triviale, auf bestimmten Regeln basierende Zuweisung von Zahlwörtern zu Objekten beschreiben [Ste46]. Daraus entwachsen jedoch Probleme durch unterschiedliche Regelwerke, unterschiedliche mathematische Eigenschaften der daraus resultierenden Skalen und die Anwendbarkeit unterschiedlicher statistischer Verfahren [Ste46]. Ein Lösungsansatz fußt auf der Annahme, dass die Messskalen in unterschiedliche Klassen fallen, welche sich sowohl durch den Messvorgang selbst, als auch durch die mathematischen Eigenschaften der Skalen definieren [Ste46].

Zur Lösung dieser Problematik trug 1946 maßgeblich der Psychologe S. S. Stevens bei, dessen vorgeschlagene Klassifizierung der Messskalen bis heute allgemein akzeptiert und eingesetzt wird. Er schlug in seinem Artikel mit der Nominalskala, der Ordinalskala, der Intervallskala und der Verhältnisskala (Tabelle 2.1) die vier gebräuchlichen Skalentypen vor [Ste46].

Die Nominalskala ermöglicht die unbeschränkte Zuweisung von Zahlwörtern. Die Zahlwörter dienen im Grunde nur als Beschriftung oder Typbezeichnungen und er-

Typ	Mathematische Operationen			
	$= \neq$	$< >$	$+ -$	$\times \div$
Nominal	✓			
Ordinal	✓	✓		
Intervall	✓	✓	✓	
Verhältnis	✓	✓	✓	✓

Tabelle 2.1: Messskalen nach Stevens mit den jeweils unterstützten mathematischen Operationen - vgl. [Ste46, Table 1]

lauben damit die Identifizierung oder Klassifizierung von Entitäten [Ste46].
Die Ordinalskala entwächst der mathematischen Operation zur Darstellung von Größenverhältnissen. Die mathematische Struktur dieser Skala entspricht einer monotonen Funktion [Ste46].
Die Intervallskala erlaubt eine quantitative Bemessung von Größen im eigentlichen Sinn. Änderungen in Beobachtungen werden äquivalente Werte zugeordnet, ein Nullpunkt per Konvention festgelegt. Die Transformation zwischen den Werten x und x' zweier verschiedener Intervallskalen folgt dabei einer Gleichung der Form $x' = ax + b$ (z.B. °C nach °F) [Ste46].
Auf die Verhältnisskala kann zurückgegriffen werden, sofern alle vier mathematischen Operationen anwendbar sind. Dazu zählen die Gleichheit, das Größenverhältnis, die Intervallgleichheit und die Verhältnisgleichheit. Werte aus zwei verschiedenen Skalen diese Typs können durch einfache Multiplikation mit einer Konstante ineinander überführt werden (z.B. Kilo nach Pound). Zudem ist stets ein Nullpunkt impliziert [Ste46].

In die Klasse der Intervall- und Verhältnisskalen können nun auch die metrischen Konfigurationsoptionen eines Softwaresystems eingeordnet werden. Eine detailiertere Beleuchtung dieser Tatsache findet sich in Abschnitt 2.2.1.

2.2 Konfigurierbare Systeme

Prinzipiell lässt sich beinahe jedes Softwareprodukt konfigurieren. Im weitesten Sinn kann daher jede getätigte Eingabe, die den Programmablauf beeinflusst, als eine Art „Konfiguration“ betrachtet werden.[1]
Allgemein wird die Bezeichnung „konfigurierbares System“ jedoch im Speziellen für diejenigen Softwareprodukte verwendet, deren Konfigurationsoptionen eine Anpassung bezüglich des Funktionsgehalts oder der Bestimmung des Verhaltens zulassen. Weiterhin kann es möglich sein, ein Programm für verschiedene Hardwarearchitekturen oder Betriebssysteme zu optimieren oder anzupassen.

[1] vgl. Definition: Springer Gabler Verlag, Gabler Wirtschaftslexikon, Stichwort: Konfiguration - http://wirtschaftslexikon.gabler.de/Archiv/11989/konfiguration-v10.html (31.03.2014)

Als Beispiel für Anpassungen bezüglich des Funktionsgehalts der resultierenden Software kann der in Quelltext 2.1 abgebildete Aufruf dienen, der die Kompilierung der SQLite-Datenbank[2] zeigt. Die Kompilierungsoption `-DSQLITE_ENABLE_SQLLOG` macht zusätzlichen Code in dieser Variante verfügbar, der das Erzeugen umfangreicher Logginginformationen ermöglicht. Der Switch `-m64` hingegen ist Beispiel für eine hardwarespezifische Anpassung und weist den Compiler an, Code für eine 64-bit Umgebung zu generieren.[3]

Quelltext 2.1: Kompilieren einer Variante der SQLite-Datenbank

```
1 # gcc -DSQLITE_ENABLE_SQLLOG -m64 shell.c sqlite3.c -lpthread -ldl
```

Bekannte und umfangreiche Vertreter der konfigurierbaren Systeme sind Softwareproduktlinien. Softwareproduktlinien sind softwareintensive Systeme, die eine Menge von Eigenschaften teilen und den speziellen Bedürfnissen des jeweiligen Marktes genügen [dCMU]. Ihre variable Architektur erlaubt kosteneffektive Entwicklung und meist eine deutliche Reduzierung der Time-to-Market [BKPS04, Kapitel 1].

2.2.1 Konfigurationsoptionen

Die Basis für jedes konfigurierbare System bilden die jeweils angebotenen Konfigurationsoptionen. Die Übergabe der Parameter geschieht meist in Form von Kompilierungsflags, Konfigurationsdateien oder Kommandozeilenparametern [SRK+13]. Unterschieden werden nach A. Rabkin und R. Katz hauptsächlich drei Parameterkategorien: Parameter zur Wahl des Betriebsmodus, Parameter zur Definition externer Bezeichner sowie solche zur Übergabe numerischer Werte [RK11].
Zur Wahl des Betriebsmodus ist es möglich aus einer kleinen Menge von Optionen zu wählen. Als Vertreter dieser Kategorie sind vor allem die booleschen Konfigurationsoptionen zu nennen. Sie erlauben beispielsweise das komfortable An- und Abwählen von Funktionalitäten, wie es in Quelltext 2.1 ersichtlich ist. Die externen Bezeichner dienen zum Verweis auf Entitäten außerhalb des Programms. Beispiele hierfür sind Zeichenfolgen oder Zahlen, die auf Dateispeicherorte oder Netzwerkadressen verweisen. Die am häufigsten auftretende Variante ist allerdings der numerische Parameter zur Eingabe von Ganz- und Gleitkommazahlen. Das Festlegen einer maximalen Dateigröße würde dieser Kategorie zugerechnet [RK11].
Zur letzten Kategorie werden auch die metrischen Konfigurationsoptionen gezählt, deren Einflüsse auf die nichtfunktionalen Eigenschaften eines Programms im weiteren Verlauf dieser Arbeit näher betrachtet werden.

2.2.2 Funktionale und nichtfunktionale Eigenschaften

Die Anforderungen an die Eigenschaften eines Programms sind vielfältig und definieren sich laut IEEE Standard 610.12-1990 mit den „von einem Benutzer zur Lösung eines Problems benötigten Bedingungen und Leistungen“ [IEE90, „requirements“].
Auf oberster Ebene lassen sich die Anforderungen bezüglich der funktionalen und nichtfunktionalen Eigenschaften gliedern. Die sich ergebende Hierarchie ist in Abbildung 2.1 ausschnittsweise dargestellt.

[2] SQLite Home Page - https://www.sqlite.org/ (31.03.2014)
[3] i386 and x86-64 Options - Using the GNU Compiler Collection (GCC) - http://gcc.gnu.org/onlinedocs/gcc-4.8.2/gcc/i386-and-x86-64-Options.html (31.03.2014)

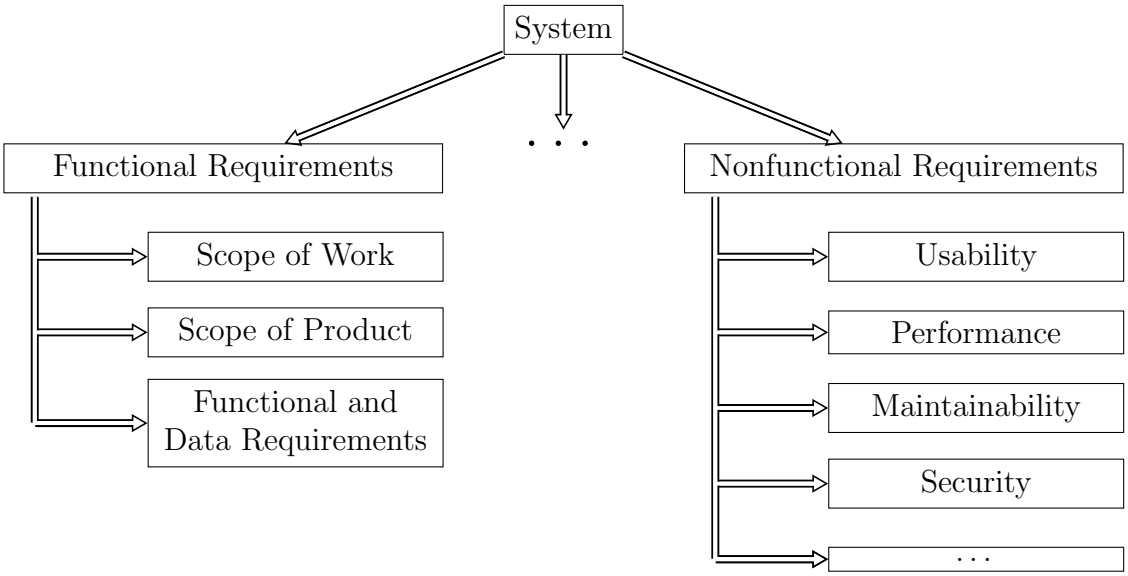


Abbildung 2.1: Gliederung der Anforderungen bezüglich der funktionalen und nichtfunktionalen Eigenschaften von Software - vgl. [RR06, Appendix B]

Funktionale Anforderungen beziehen sich auf den Funktionsumfang, den ein bestimmtes Programm bereitzustellen hat. Die nichtfunktionalen Anforderungen sind dagegen in erster Linie an die qualitativen Eigenschaften eines Softwaresystems gerichtet. Die Qualität ist dabei darin zu erkennen, wie gut das System seine jeweiligen Aufgaben erfüllt [RR06, Kapitel 1]. Ausgehend vom „Volere Requirements Specification Template“, einem „Leitfaden zum Schreiben exakter und vollständiger Lastenhefte“ [RR06, Appendix B], lassen sich die nichtfunktionalen Anforderungen klar kategorisieren. Die Optik und Haptik oder die Performanz hinsichtlich der Schnelligkeit oder Sicherheit eines Programms sind hier wohl die bekanntesten Vertreter [RR06, Kapitel 8].

Die Varianten eines konfigurierbaren Systems definieren sich hauptsächlich durch die gewünschten funktionalen Eigenschaften, wobei jede der möglichen Produktvarianten wiederum direkten Einfluss auf die nichtfunktionalen Eigenschaften dieser hat. Man erhält eine Vielzahl von Varianten, deren nichtfunktionale Eigenschaften aber - wie bereits in 1 erwähnt - nicht ohne Weiteres quantifizierbar sind. Gegeben sei hierzu beispielsweise ein fiktives Datenbanksystem, welches Optionen zur Konfiguration der Cache- und Seitengröße zur Verfügung stellt. Es scheint offensichtlich, dass die Einstellungen Einfluss auf den Arbeitsspeicherbedarf im Allgemeinen haben. Und auch wenn eine Erweiterung des Cache eine bessere Performanz bei Datenbankabfragen vermuten lässt, ist es nicht möglich, die Fragen nach der performantesten Variante pauschal zu beantworten.

2.2.3 Definitionen und Begrifflichkeiten

Zum besseren Verständnis der folgenden Abschnitte gelten zudem die hier aufgelisteten Definitionen und Begrifflichkeiten.

Konfigurationsoption Eine Konfigurationsoption (Abschnitt 2.2.1) erlaubt das festlegen einer bestimmten Eigenschaft eines konfigurierbaren Systems (Ab-

schnitt 2.2). Synonym wird im Kontext dieser Arbeit der Begriff „Parameter“ gebraucht.

Konfiguration Eine Konfiguration wird beschrieben durch eine Menge gesetzter Konfigurationsoptionen. Konfigurationen unterscheiden sich dabei in der jeweils gewählten Ausprägung der Optionen.

Variante Eine Variante stellt das aus einer Konfiguration abgeleitete System dar.

Featuremodell Ein Featuremodell repräsentiert ein konfigurierbares System in strukturierter Form. Es enthält unter anderem Beschreibungen der vorhandenen Konfigurationsoptionen und deren Ausprägungen.

2.3 SPL Conqueror

Die Variabilität und der daraus resultierende (extrem) große Variantenraum führt zu individuellen Ausprägungen der nichtfunktionalen Eigenschaften. Um diese zu bestimmen, müssen die Programmvarianten erzeugt und unter dem jeweiligen Aspekt gemessen werden. Zur Automatierung dieses Vorgangs wurde, im Zuge der durch Siegmund et al. angestellten Forschungen, „SPL Conqueror“ entwickelt, ein Softwaretool zur Messung und Optimierung nichtfunktionaler Eigenschaften von Softwareproduktlinien [Sie]. Das Tool stellt einen ganzheitlichen Ansatz zur Messung nichtfunktionaler Eigenschaften anhand von benutzerdefinierten Metriken dar [SRK+12].

Die Berechnung einer unter bestimmten Gesichtspunkten optimalen Variante gliedert sich dabei im Wesentlichen in vier Schritte. Zuerst werden die hinsichtlich der gewünschten funktionalen Eigenschaften benötigten Konfigurationsoptionen identifiziert und festgelegt. In Verbindung mit der anschließenden Beschränkung auf eine bestimmte nichtfunktionale Eigenschaft ermöglicht dies eine Verkleinerung des zu untersuchenden Variantenraums. Unter Betrachtung der verbleibenden Varianten wird eine Konfiguration errechnet, welche die zuvor festgelegte nichtfunktionale Eigenschaft optimiert. Zusätzlich kann nun im abschließenden Schritt eine weitere Optimierung der gefundenen Variante, beispielsweise auf Code-Ebene, durchgeführt werden [SRK+12].

Auf die im Rahmen dieser Arbeit entstandenen Erweiterungen für SPL Conqueror wird in Abschnitt 4.2 näher eingegangen.

2.4 Statistische Versuchsplanung

Die Grundsteine der statistischen Versuchsplanung legte bereits 1935 Sir Ronald A. Fisher mit seinem Buch „The Design of Experiments“ [Fis35]. Damals noch hauptsächlich für die landwirtschaftliche Versuchstechnik vorgesehen, wurden die Vorgehensweisen seitdem erheblich weiterentwickelt [Pet91, Kapitel 1]. Im Rahmen dieser Arbeit werden sie nun auch im Bereich der konfigurierbaren Systeme angewandt.

Die statistische Versuchsplanung, im Englischen auch mit „Design of Experiments“ (DoE) bezeichnet, stellt ein Verfahren zur effizienten Gestaltung von Experimenten dar. Die Effizienz eines experimentellen Designs ist dabei in dem Ziel begründet, aus

einer möglichst geringen Anzahl von Versuchen, das einem System zugrundeliegende Modell bestmöglich zu bestimmen. Erreicht wird dies durch die mathematische Abbildung des Systemverhaltens unter zuhilfenahme der aus den Versuchen erhaltenen Informationen [Pet91, Kapitel 1].

2.4.1 Grundbegriffe

Um den Ausführungen über die experimentellen Designs im Anschluss besser folgen zu können, werden zuerst einige grundlegende Begriffe der behandelten Domäne geklärt.

System

Mit „System“ wird im Kontext der statistischen Versuchsplanung das zu untersuchende Objekt bezeichnet. Es beinhaltet sämtliche kontrollierbaren und unkontrollierbaren Eingabeparameter sowie alle zu betrachtenden Ausgabeparameter. In Abbildung 2.2 wird ein solches System schematisch dargestellt.

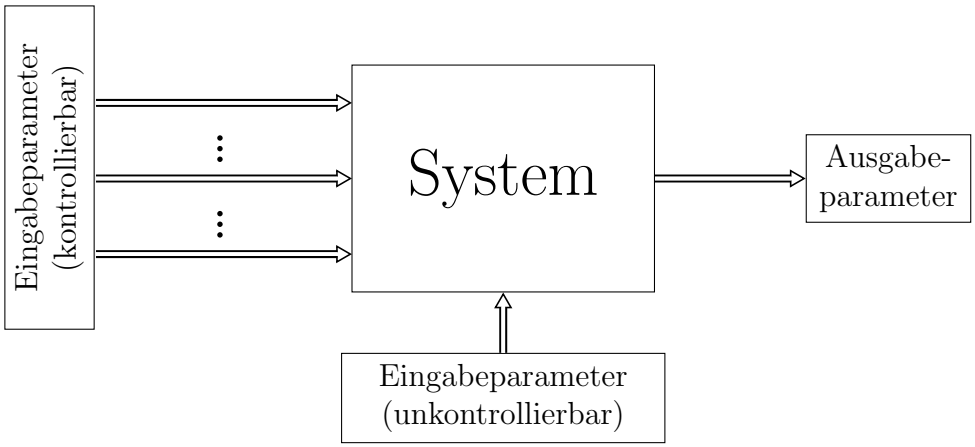


Abbildung 2.2: Schematische Darstellung eines zu untersuchenden Systems - vgl. [SvBH10, Kapitel 1.2.1]

Ein- und Ausgabeparameter

Die Eingabeparameter eines Systems sind überwiegend durch die vorhandenen Konfigurationsoptionen bestimmt. Sie werden vom Benutzer, wie in Abschnitt 2.2.1 beschrieben, definiert. Neben diesen existieren für gewöhnlich auch unkontrollierbare Parameter - im Englischen häufig mit „Noise“ bezeichnet. Sie stellen Eingaben dar, die nicht der Kontrolle des Benutzers unterliegen - zum Beispiel die Prozessorauslastung. Die kontrollierbaren Parameter in Verbindung mit den Unkontrollierbaren bilden systeminterne Prozesse auf einen oder mehrere Ausgabeparameter ab. Die zu untersuchende nichtfunktionale Eigenschaft eines Programms ist Beispiel für einen solchen Ausgabeparameter. Als Ausgabeparameter werden demnach nicht nur in Dateien geschriebene Ergebnisse, wie beispielsweise eine XML-Datei, oder Bildschirmausgaben bezeichnet. Vielmehr umfassen sie das gesamte beobachtbare Verhalten eines Systems.

Projiziert auf die SQLite-Datenbank definieren sich, wie in Abbildung 2.3 ersichtlich, die Ein- und Ausgabeparameter wie folgt:

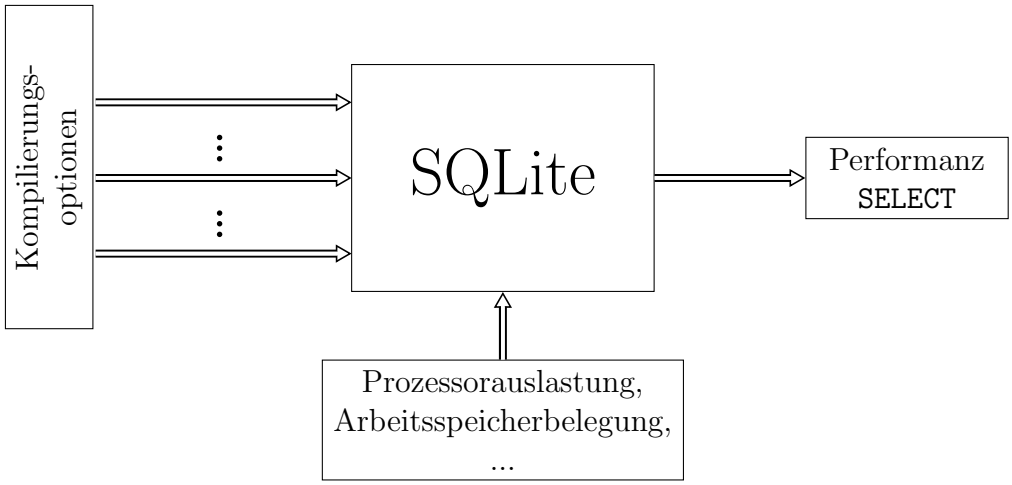


Abbildung 2.3: Übertragung der Darstellung auf die SQLite-Datenbank

- **Eingabeparameter (kontrollierbar)**:
  SQLite erlaubt bereits bei der Kompilierung weitreichende Anpassungen des Systemverhaltens mittels einer Vielzahl verschiedener Optionen. Ein Beispiel für einen booleschen Parameter ist `SQLITE_CASE_SENSITIVE_LIKE`, welcher festlegt `LIKE`-Statements standardmäßig unter Beachtung der Groß- und Kleinschreibung auszuführen.[4]
  Auch die metrische Option `SQLITE_DEFAULT_PAGE_SIZE=<bytes>` erlaubt eine Anpassung des Laufzeitverhaltens, indem sie die Standardseitengröße im Arbeitsspeicher definiert.[4]
  Zudem stellt natürlich auch jede Anfrage an die kompilierte, betriebsbereite Datenbank eine kontrollierbare Eingabe dar.

- **Eingabeparameter (unkontrollierbar)**:
  Die unkontrollierbaren Eingabeparameter, deren Ausprägungen durchaus signifikanten Einfluss auf das Systemverhalten haben können, sind weitestgehend generischer Natur. So spielen meist die Prozessorauslastung oder aber die Belegung des Arbeitsspeichers bei Softwaresystemen eine wichtige Rolle.

- **Ausgabeparameter**:
  Es wird hier die Performanz im Bezug auf die Schnelligkeit der Ausführung eines `SELECT`-Statements betrachtet. Sie ist also ein Vertreter nichtfunktionaler Eigenschaften.

Die jeweiligen Ein- und Ausgabeparameter sind es also, die eine Variante definieren. Folglich wächst die Anzahl der möglichen Varianten mit der Anzahl der verfügbaren Konfigurationsoptionen.

Stufen der Eingabeparameter

Zur effizienten Generierung der zu untersuchenden Varianten wird in der Regel vom (teils) großen Wertebereich der metrischen Konfigurationsoptionen abstrahiert. Dabei werden jeweils mindestens zwei unterschiedliche Stufen festgelegt und bei Erzeugung der Varianten auf den tatsächlichen Wertebereich abgebildet. Die Kodierung

[4] Compilation Options For SQLite - https://sqlite.org/compile.html (05.04.2014)

der diskreten Stufen folgt verschiedenen Konventionen, gängig ist die Repräsentation $-/+$ oder im Falle von drei Stufen $-1/0/+1$ [SvBH10, Kapitel 1.2.4].

Tabelle 2.2 und Tabelle 2.3 zeigen beispielhaft die Abbildung von zwei oder drei Stufen auf zugehörige Parameterwerte. Die Wertebereiche der beiden Eingabeparameter a und b definieren sich dabei wie folgt:

$a \in A$ mit $A = \{1, 2, ..., 9\}$ und $b \in B$ mit $B = \{64, 128, 256, 512, 1024\}$

	-	+
a	1	9
b	64	1024

Tabelle 2.2: Abbildung der Parameterwerte durch ein zweistufiges Modell

	-1	0	+1
a	1	5	9
b	64	256	1024

Tabelle 2.3: Abbildung der Parameterwerte durch ein dreistufiges Modell

3. Ausgewählte experimentelle Designs

In diesem Kapitel wird ein Überblick über die verfolgten Ansätze und Eigenschaften ausgewählter experimenteller Designs gegeben.

Ein experimentelles Design beschreibt die einer Versuchsfolge zugrundeliegende Vorgehensweise. Es legt sowohl die zu verwendenden Stufen, als auch die zu untersuchenden Varianten fest [fQCSD83, 6.1 - „Design of Experiments“].
Gut entworfene Experimente erlauben oft die Ableitung eines Modells für das Systemverhalten. Auf diese Weise bestimmte Modelle werden „empirische Modelle“ genannt [Mon05, Kapitel 1-1]. „SPL Conqueror“ übernimmt dazu die Generierung der Varianten anhand der Eingabeparameter und quantisiert mittels numerischer Mathematik die zugehörige nichtfunktionale Eigenschaft. Anschließend benutzt er die gefundenen Wertpaare, um mit Hilfe linearer Regression eine Funktion zu finden, die die Eingabeparameter auf einen gewählten Ausgabeparameter möglichst exakt abbildet.

Für den folgenden Abschnitt soll das Beispiel aus Abschnitt 2.4.1 um den Eingabeparameter c erweitert werden.

$$a \in A \text{ mit } A = \{1, 2, ..., 9\}$$
$$b \in B \text{ mit } B = \{64, 128, 256, 512, 1024\}$$
$$c \in C \text{ mit } C = \{1, 3, 5, 7, 9\}$$

Quelltext 3.1 zeigt einen Applikationsaufruf in der UNIX-Shell, bei welchem die Parameter über Kommandozeilenargumente übermittelt werden. Die Parameter definieren sich im diesem Fall mit $a = 1$, $b = 64$ und $c = 3$.

Quelltext 3.1: Aufruf einer Beispielapplikation mit Kommandozeilenargumenten

```
1 # run_example -a 1 -b 64 -c 3
```

#	a	b	#	a	b	#	a	b
0	1	64	15	4	64	30	7	64
1	1	128	16	4	128	31	7	128
2	1	256	17	4	256	32	7	256
3	1	512	18	4	512	33	7	512
4	1	1024	19	4	1024	34	7	1024
5	2	64	20	5	64	35	8	64
6	2	128	21	5	128	36	8	128
7	2	256	22	5	256	37	8	256
8	2	512	23	5	512	38	8	512
9	2	1024	24	5	1024	39	8	1024
10	3	64	25	6	64	40	9	64
11	3	128	26	6	128	41	9	128
12	3	256	27	6	256	42	9	256
13	3	512	28	6	512	43	9	512
14	3	1024	29	6	1024	44	9	1024

Tabelle 3.1: Vollständiger Versuchsplan auf Basis der Parameter a und b

Die Parameter a und b sind obligatorisch, c wird als optional betrachtet und nimmt per Default den Wert „1“ an. Für die Abbildung der Eingabe- auf den Ausgabeparameter wird eine Funktion f der Form $f(a, b, c) = a + bc$ angenommen.

3.1 Vollfaktorielles Design

Das naheliegendste Vorgehen zur Findung eines zugrundeliegenden empirischen Modells ist die Betrachtung aller möglichen Parameterkombinationen. Man spricht von einem vollständigen Versuchsplan oder auch einem vollfaktoriellen Design. Es stellt das leistungsfähigste Design im Bezug auf die Untersuchung der Effekte von zwei oder mehr Faktoren dar. Die Leistungsfähigkeit besteht hier darin, sowohl Hauptwirkungen als auch Wechselwirkungen identifizieren zu können [Pet91, Kapitel 5]. Mathematisch betrachtet handelt sich sich bei der resultierenden Matrix um das kartesische Produkt der Wertemengen. Im Falle der Betrachtung der Parametern a und b erhalten wir die 45 in Tabelle 3.1 abgebildeten Versuchsläufe, bestehend aus allen möglichen Kombinationen der beiden Parameter. Abbildung 3.1 visualisiert die Abdeckung des, durch die Parameter a und b aufgespannten, Variantenraums. Ein mit einem Punkt versehenes Quadrat markiert dabei eine zu testende Parameterkombination.

Die Anzahl der Versuchsläufe r in einem vollfaktoriellen Design ergibt sich aus dem Produkt der Mächtigkeit der Wertemengen $x_1, x_2, ..., x_i$:

$$r = \prod_{n=1}^{i} |x_n| \tag{3.1}$$

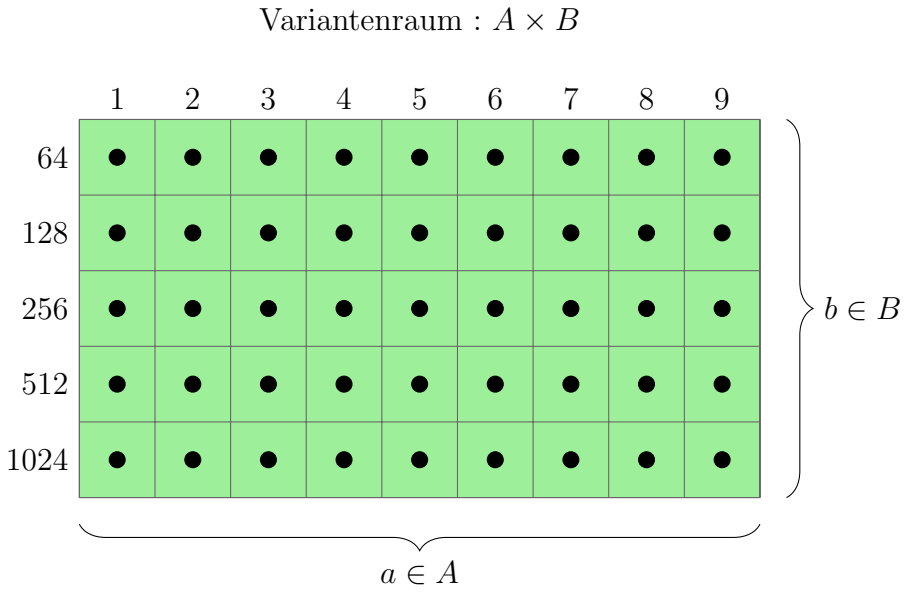


Abbildung 3.1: Abdeckung des Variantenraums durch ein vollständiges faktorielles Design

Die Mächtigkeit einer Wertemenge ist dabei gleichzusetzen mit der Anzahl der möglichen Stufen des zugehörigen Eingabeparameters.
Im Fall der Beispielapplikation unter Berücksichtigung des Parameters c sind bereits 225 Messungen durchzuführen:

$$r = |A| \cdot |B| \cdot |C| = 9 \cdot 5 \cdot 5 = 225 \tag{3.2}$$

Wie zu erkennen ist, hängt die Zahl der benötigten Versuchsläufe sowohl von den Stufen als auch von der Anzahl der Eingabeparameter ab. Aufgrund dessen führen bereits kleine Erhöhungen dieser beiden Variablen zu einem erheblichen Mehraufwand bei der Durchführung des Experiments. Um diesem Umstand zu begegnen, werden stattdessen häufig zweistufige faktorielle Designs verwendet, auf welche nun genauer eingegangen wird.

3.2 Zweistufiges faktorielles Design - 2^k

Beim zweistufigen faktoriellen Design - auch als 2^k-Design bezeichnet - handelt es sich um ein Design mit k Eingabeparametern auf jeweils zwei Stufen. Zu diesem Zweck wird von den eigentlichen Stufen abstrahiert und nur der jeweils minimale und maximale Wert einer metrischen Konfigurationsoption betrachtet. Kodiert werden die beiden Ausprägungen für gewöhnlich mit $-$ und $+$ [SvBH10, Kapitel 1.2.4]. Das 2^k-Design resultiert daher in der für ein vollfaktorielles Design kleinstmöglichen Anzahl benötigter Versuchsläufe $r = 2^k$.

Die Tabelle 3.2 zeigt die im Bezug auf die Beispielapplikation generierte Designmatrix und ihre Abbildung auf die tatsächlichen Wertemengen der Parameter. Im Beispielfall bilden a und b die Spalten x_1 und x_2 und die Werte -1 und $+1$ werden mit $-$ und $+$ abgekürzt. Verglichen mit dem vollfaktoriellen Versuchsplan kann die

#	a	b
0	−	−
1	+	−
2	−	+
3	+	+

$\Rightarrow$

#	a	b
0	1	64
1	9	64
2	1	1024
3	9	1024

Tabelle 3.2: Abbildung der kodierten Designmatrix auf die Wertemengen A und B

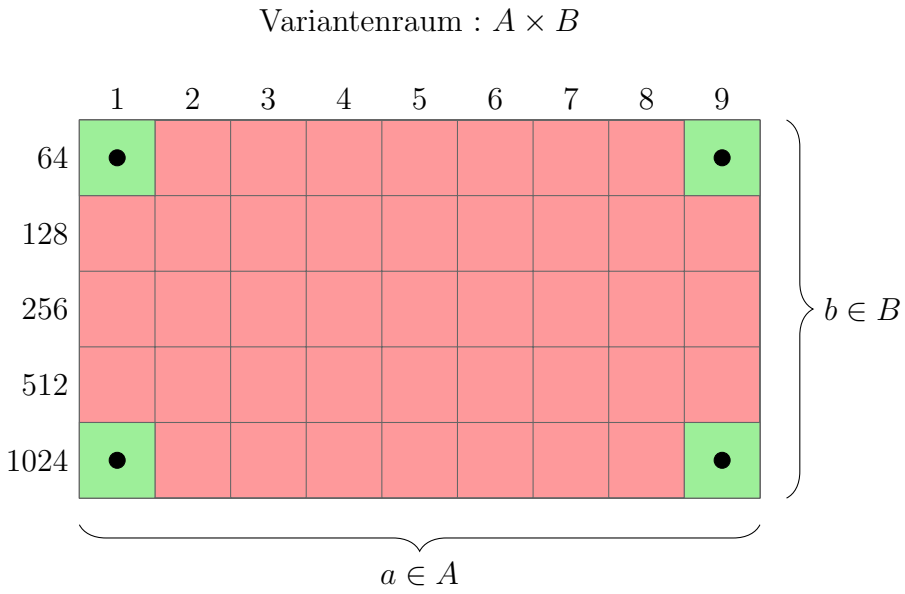


Abbildung 3.2: Abdeckung des Variantenraums durch ein zweistufiges faktorielles Design

Anzahl der zu untersuchenden Konfigurationen von 45 auf nur vier - also um etwa den Faktor 11 - reduziert werden. Abbildung 3.2 zeigt die Visualisierung der daraus resultierenden Abdeckung des Variantenraums. Die selektierten Kombinationen bilden in diesem Design die Randpunkte des jeweiligen k-dimensionalen Raums.

Die Betrachtung nur zweier Stufen setzt dabei jedoch einen linearen Zusammenhang der Eingabeparameter voraus. Die dem abzubildende Systemprozess zugrundeliegende Funktion darf folglich keine Interaktionen beinhalten [Mon05, Kapitel 6-1].
Um eine Möglichkeit zur Erkennung einer Nichtlinearität oder Krümmung in der zugrundliegenden Funktion zu schaffen, kann das Design mit einer oder mehreren Mittelpunktsmessungen („Center Points Runs“) ergänzt werden [Mon05, Kapitel 6-6]. Diese setzen mindestens drei Stufen je Parameter voraus und sind vor allem für die Box-Behnken Designs und die zentral zusammengesetzten Versuchspläne, welche in Abschnitt 3.4 und Abschnitt 3.5 noch näher betrachtet werden, von Relevanz.

Darüber hinaus existieren zur Erkennung quadratischer Zusammenhänge weitere Designformen. Dreistufige faktorielle Designs erweitern im Grunde die eben beschriebenen Versuchspläne um die Untersuchung einer dritten Stufe und können auch in neue Versuchspläne eingebettet werden. Das Ergebnis sind gemischt-stufige faktori-

elle Designs oder fraktionelle Designs [Mon05, Kapitel 9], deren Funktionsweise in Abschnitt 3.3 genauer betrachtet wird.

Wechselwirkungen

Wirft man einen abschließenden Blick auf Tabelle 3.2 erkennt man die Faktorstufenkombinationen (1), a, b und ab. Das Symbol (1) bezeichnet dabei die Konfiguration bei der alle Parameter auf der Stufe „−“ untersucht werden. Ein Buchstabe weist auf die Untersuchung des zugehörigen Parameters auf der Stufe „+“ hin [Pet91, Kapitel 6.1].
Durch ab wird also die Konfiguration mit den beiden Parametern a und b auf der Stufe „+“ bezeichnet. Es besteht hier eine Wechselwirkung - oder auch Interaktion - zwischen den Parametern [fQCSD83, 6.15 - „Interaction“]. Der Begriff beschreibt die Tatsache, dass der Einfluss eines Eingabeparameters auf einen Ausgabeparameter von der Stufe eines anderen Parameters abhängig ist [Pet91, Kapitel 6.1].

3.3 2^{k-p} - teilfaktorielle Designs

Ungeachtet der Beschränkung auf nur zwei Parameterstufen, steigt die Anzahl der Versuchsläufe auch beim zweistufigen faktoriellen Design mit Zahl der Parameter k quadratisch an. Zur weiteren Reduzierung des Versuchsaufwands wird auf teilfaktorielle Versuchspläne zurückgegriffen. Als teilfaktorielle Versuchspläne werden laut „Amerikanischer Gesellschaft für Qualitätskontrolle“ diejenigen Versuchspläne bezeichnet, welche nur einen ausgewählten Bruchteil der Kombinationen eines vollständigen Versuchsplans beeinhalten [fQCSD83, 6.25.7 - „Fractional Factorial Design“].
Allgemein wird ein teilfaktorielles 2^k-Design mit 2^{k-p} Versuchsläufen als 2^{k-p}-teilfaktorielles Design bezeichnet [Mon05, Kapitel 8-4]. Die Variable p entspricht der Anzahl der Spalten, um die das 2^k-Design reduziert wird. Die Zahl der Versuchsläufe beträgt folglich nur $\frac{1}{2^p}$ der in einem zweistufigen faktoriellen Design vorgesehenen Kombinationen. Im Bezug auf die Beispielapplikation mit den drei Parametern a, b und c wird nachfolgend ein 2^{3-1}-teilfaktorielles Design Verwendung finden, anhand dessen die Funktionsweise der Reduktion genauer erklärt wird.

#	a	b
0	−	−
1	+	−
2	−	+
3	+	+

Tabelle 3.3: Designmatrix des 2^2-Designs

Die Basis für ein 2^{3-1}-teilfaktorielles Design bilden die Konfigurationen eines vollständigen 2^2 Versuchsplans (Tabelle 3.3) [NIS, Kapitel 5.3.3.4.2]. Wie zu erkennen ist, existiert jedoch keine Spalte für die Belegung des Parameters c.

Im Allgemeinen wird der vollständige 2^{k-1} Versuchsplan um eine Spalte für den k-ten Parameter ergänzt. Die jeweils zu testende Stufe wird von der bestehenden

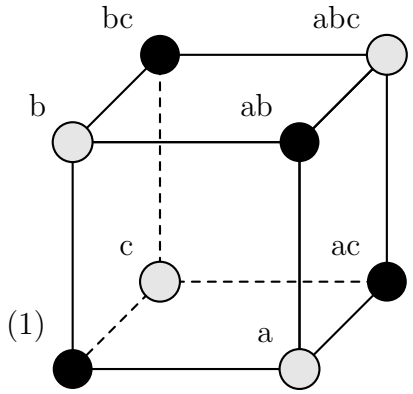


Abbildung 3.3: Darstellung des Variantenraums eines 2^3-Designs. a, b und c markieren jeweils die Untersuchung auf der höchsten Stufe des zugehörigen Parameters. abc bezeichnet somit den Versuch mit allen drei Parametern auf der Stufe +, (1) die Situation in der alle drei Parameter die Stufe − annehmen. Die schwarzen und grauen Punkte visualisieren die Zugehörigkeit zu den beiden definierenden Relationen.

höchstwertigen Interaktion $ABC \cdots (K-1)$ abgeleitet. Im Falle des vorliegenden 2^{3-1}-teilfaktoriellen Desgins leitet man - mit $I = ABC$ (bzw. $I = -ABC$) als definierende Relation - den Parameter c aus dem Produkt $C = AB$ (bzw. $C = -AB$) her. Das Ergebnis ist ein Design der Auflösung III, dargestellt in Tabelle 3.4 [Mon05, Kapitel 8-2.3].

$I = ABC$

#	a	b	$c = a \cdot b$
0	−	−	+
1	+	−	−
2	−	+	−
3	+	+	+

$I = -ABC$

#	a	b	$c = -a \cdot b$
0	−	−	−
1	+	−	+
2	−	+	+
3	+	+	−

Tabelle 3.4: Designmatrix des 2^{3-1}-Designs

Grafisch betrachtet, lässt sich das 2^3 zweistufige faktorielle Design als Würfel darstellen. Der Würfel in Abbildung 3.3 visualisiert die Menge aller Konfigurationen im zweistufigen faktoriellen Versuchsplan (Tabelle 3.2). Der 2^{3-1}-teilfaktorielle Versuchsplan resultiert in zwei distinkten Konfigurationsmengen, deren zugehörige-Punkte mit den schwarzen und grauen Kreisen kenntlich gemacht sind. Betrachtet man Tabelle 3.5, wird deutlich woraus die jeweilige Konfigurationsmengen entstehen. Die grau gefüllten Punkte repräsentieren die auf $I = ABC$, die schwarzen Punkte die auf $I = -ABC$ basierenden Konfigurationen. Also einmal der Konfigurationsblock, dessen Zeilen in der Spalte ABC ein Pluszeichen besitzen und der zweite Block dessen Zeilen mit dem Minuszeichen versehen sind.

Die Spalten A, B und C in Tabelle 3.5 stellen die Haupteffekte [fQCSD83, 6.14 - „Main Effect“], also die Parameter mit dem voraussichtlich größten Einfluss auf

	Faktorielle Effekte							
Konfiguration	I	A	B	C	AB	AC	BC	ABC
a	+	+	−	−	−	−	+	+
b	+	−	+	−	−	+	−	+
c	+	−	−	+	+	−	−	+
abc	+	+	+	+	+	+	+	+
ab	+	+	+	−	+	−	−	−
ac	+	+	−	+	−	+	−	−
bc	+	−	+	+	−	−	+	−
(1)	+	−	−	−	+	+	+	−

Tabelle 3.5: Designmatrix eines 2^3-Designs [Mon05, Kapitel 8-2.1]

den oder die Ausgabeparameter, dar. Daneben sind zudem mit AB, AC und BC noch alle möglichen Interaktionen angegeben, die ebenfalls einen - wenn auch unter Umständen geringeren - Einfluss haben. Im Fall des vorliegenden 2^{3-1} Designs wurde die Belegung des Parameters c aus der Belegung von a und b abgeleitet - die Effekte sind konfundiert [fQCSD83, 6.10 - „Confounding“] und nicht mehr unterscheidbar. Werden mehrere diese Effekte kombiniert, spricht man von „Aliasing“. Der Grad des „Aliasings“ bestimmt nun auch die Auflösung eines Designs, die im Falle des 2^{k-1} Designs III beträgt. Formal definieren sich die häufigsten Auflösungen wie folgt:

Auflösung III

> Ein Design hat die Auflösung III, sofern kein Haupteffekt mit einem anderen Haupteffekt „aliased“ ist. Haupteffekte dürfen jedoch mit Zweifaktorinteraktionen und diese ebenfalls untereinander „aliased“ sein.

Auflösung IV

> Ein Design hat die Auflösung IV, sofern kein Haupteffekt weder mit einem anderen Haupteffekt noch mit einer Zweifaktorinteraktion „aliased“ ist. „Aliasing“ zwischen Zweifaktorinteraktionen selbst ist jedoch zulässig.

Auflösung V

> Ein Design hat die Auflösung V, sofern kein Haupteffekt oder Zweifaktorinteraktion mit einem anderen Haupteffekt oder Zweifaktorinteraktion „aliased“ ist. „Aliasing“ zwischen Zwei- und Dreifaktorinteraktionen ist jedoch zulässig.

[Mon05, Kapitel 8-2.2]

Es ist weiterhin möglich das Design durch ein oder mehrere Mittelpunktsmessungen zu ergänzen, um anhand der Versuchsergebnisse auf quadratische Modelle schließen

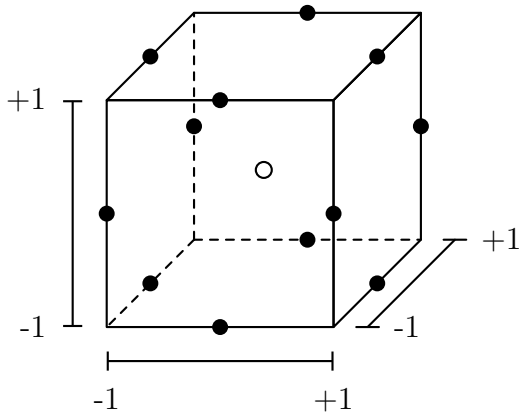


Abbildung 3.4: Darstellung eines Box-Behnken Designs für drei Parameter. Die schwarzen Punkte markieren die zu untersuchenden Konfigurationen - vgl. [Mon05, Abbildung 11-22]

zu können. Diese dritte Parameterstufe machen sich unter anderem auch die Box-Behnken Designs im Folgenden zunutze.

3.4 Box-Behnken Design

Das Box-Behnken Design stellt ein unabhängiges, quadratisches Design dar. Es beinhaltet kein eingebettetes faktorielles oder fraktionelles Design und setzt mindestens drei Stufen je Parameter voraus [NIS, Kapitel 5.3.3.6.2]. Wenngleich auf die Integration der Versuchsläufe eines fraktionellen Designs verzichtet wurde, wird sich dennoch zur Blockgenerierung den Designmatrizen von zweistufigen faktoriellen Designs bedient [BB60].
Box und Behnken entwickelten diese Versuchspläne 1960 auf Basis dreistufiger faktorieller Designs. Die Versuchspläne entstehen durch Reduktion der 3^k faktoriellen Versuchspläne, aber unter Erhaltung der Orthogonalität. Ein Design wird dann als „orthogonales Design“ bezeichnet, wenn alle Parameterpaare mit einer bestimmten Stufe gemeinsam und gleich oft auftreten [fQCSD83, 6.25.9]. Die angesprochene Reduktion besteht in der Eliminierung kompletter Sphären und/oder der Eliminierung von Teilsphären der zugrundeliegenden Designs [Pet91, Kapitel 7.7].
Eine Sphäre kann als Kugel betrachtet werden, deren Mittelpunkt der Zentralpunkt des Versuchsraums ist. Die in einem Box-Behnken Design enhaltenen Versuchspunkte liegen auf den Schnittpunkten der Kugeloberflächen mit den Rändern des Versuchsraums.

Die von Box-Behnken entworfenen Designmatrizen für 4, 6 und 9 Faktoren sind in Tabelle 3.6 einsehbar. Das für 9 Faktoren vorgesehene Design wurde hier bereits auf den speziellen Anwendungsfall und im Hinblick auf die Implementierung abgeändert. Möglich wird dies aufgrund des durch den Simulator gegebenen Determinismus, der wiederholte Messungen überflüssig macht. Konkret wurde dazu die Anzahl der Mittelpunktsversuche von 10 auf nur einen Versuch reduziert sowie die ursprünglich replizierten Zeilen 1-3 vernachlässigt.

In Abbildung 3.4 ist die geometrische Darstellung eines Box-Behnken Designs für drei Parameter ersichtlich. Sämtliche Punkte dieses sphärischen Designs liegen auf

Design #	Faktoren	Designmatrix	Versuche
2	4	$\begin{bmatrix} \pm1 & \pm1 & 0 & 0 \\ 0 & 0 & \pm1 & \pm1 \\ \pm1 & 0 & 0 & \pm1 \\ 0 & \pm1 & \pm1 & 0 \\ \pm1 & 0 & \pm1 & 0 \\ 0 & \pm1 & 0 & \pm1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	25
4	6	$\begin{bmatrix} \pm1 & \pm1 & 0 & \pm1 & 0 & 0 \\ 0 & \pm1 & \pm1 & 0 & \pm1 & 0 \\ 0 & 0 & \pm1 & \pm1 & 0 & \pm1 \\ \pm1 & 0 & 0 & \pm1 & \pm1 & 0 \\ 0 & \pm1 & 0 & 0 & \pm1 & \pm1 \\ \pm1 & 0 & \pm1 & 0 & 0 & \pm1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	49
6	9	$\begin{bmatrix} \pm1 & 0 & 0 & \pm1 & 0 & 0 & \pm1 & 0 & 0 \\ 0 & \pm1 & 0 & 0 & \pm1 & 0 & 0 & \pm1 & 0 \\ 0 & 0 & \pm1 & 0 & 0 & \pm1 & 0 & 0 & \pm1 \\ \pm1 & \pm1 & \pm1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \pm1 & \pm1 & \pm1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \pm1 & \pm1 & \pm1 \\ \pm1 & 0 & 0 & 0 & \pm1 & 0 & 0 & 0 & \pm1 \\ 0 & 0 & \pm1 & \pm1 & 0 & 0 & 0 & \pm1 & 0 \\ 0 & \pm1 & 0 & 0 & 0 & \pm1 & \pm1 & 0 & 0 \\ \pm1 & 0 & 0 & 0 & 0 & \pm1 & 0 & \pm1 & 0 \\ 0 & \pm1 & 0 & \pm1 & 0 & 0 & 0 & 0 & \pm1 \\ 0 & 0 & \pm1 & 0 & \pm1 & 0 & \pm1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	97

Tabelle 3.6: Ausgewählte, von duplizierten Konfigurationen befreite Designmatrizen nach Box-Behnken - vgl. [BB60]

einer Kugel mit Radius $\sqrt{2}$ [Mon05, Kapitel 11-4.2]. Der Generierungsvorgang kann also auch in der Weise betrachtet werden, dass um den Zentralpunkt ringförmig jeweils zwei Parameter auf zwei Stufen in allen Kombinationen untersucht werden [SvBH10, Kapitel 2.3.2].
Die dadurch fehlenden Versuche am Randbereich des Variantenraums bringen allerdings sowohl Vorteile, als auch Nachteile mit sich. Der Vorteil äußert sich darin, dass die Messung der Extrempunkte unter Umständen mit höherem Aufwand verbunden ist, diese Punkte aber ausgespart bleiben. Als nachteilig kann sich jedoch die Tatsache erweisen, dass das aus dem Messwerten abgeleitete Modell nicht zwingend für Punkte außerhalb des Messbereichs gelten - eine Extrapolation ist für gewöhnlich unzulässig [SvBH10, Kapitel 2.3.2].

3.5 Central Composite Design

Die Bezeichnung „zentrale zusammengesetzte Versuchspläne“ - deutsch für “Central Composite Designs“ (CCD) - lässt direkt auf deren Konstruktion schließen. Für gewöhnlich besteht das CCD aus den Konfigurationen eines 2^k faktoriellen Designs ergänzt um $2k$ Sternpunktsversuche und einen oder mehrere Mittelpunktsversuche. Abbildung 3.5 und Abbildung 3.6 zeigen die Konstruktion eines CCD mit zwei und drei Parametern [NIS, Kapitel 5.3.3.6.1]. Die schwarzen Punkte stellen dabei die Konfigurationen eines 2^k faktoriellen Designs dar, die grauen Punkte die $2k$ Sternpunktversuche und den Mittelpunktsversuch.

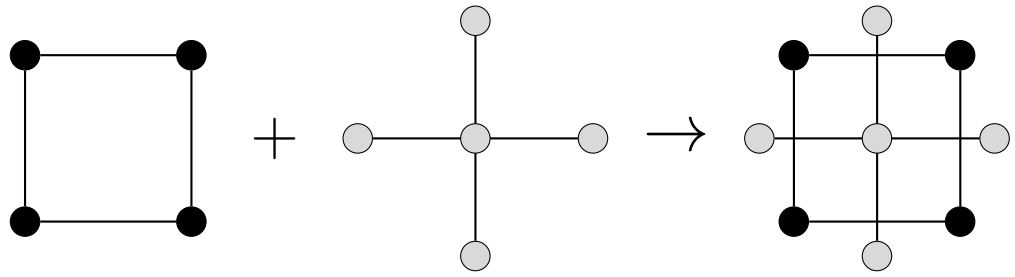


Abbildung 3.5: Konstruktion eines CCD mit zwei Parametern aus den Punkten eines 2^2 faktoriellen Designs (schwarz) und zusätzlichen Sternpunkts- und Mittelpunktsversuchen (grau) - vgl. [Pet91, Abbildung 216]

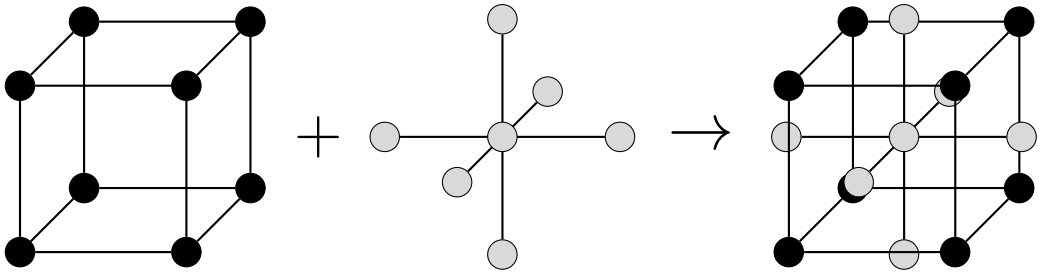


Abbildung 3.6: Konstruktion eines CCD mit drei Parametern aus den Punkten eines 2^3 faktoriellen Designs (schwarz) und zusätzlichen Sternpunkts- und Mittelpunktsversuchen (grau) - vgl. [Pet91, Abbildung 217]

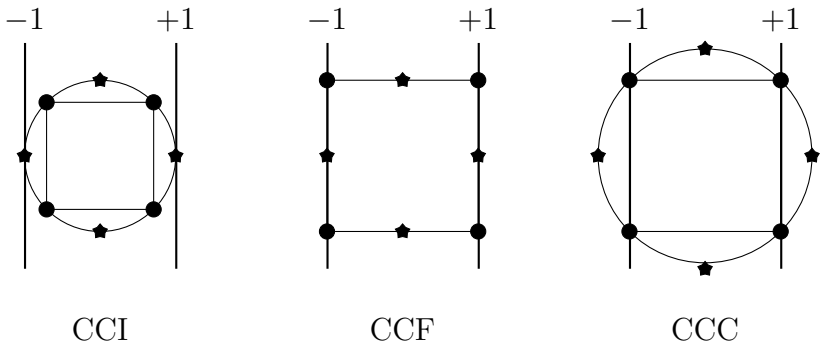


Abbildung 3.7: Die drei Typen des Central Composite Designs (vgl. [NIS, Abbildung 3.21]): „Central Composite Inscribed“, „Central Composite Face Centered“ und „Central Composite Circumscribed“

Mittels des CCD erzeugte Versuchspläne sind die wohl gängigste Vorgehensweise, wenn es um die Funktionsfindung zu Modellen zweiter Ordnung geht [Mon05, Kapitel 11-4.2]. Da sie in den verschiedensten Gebieten Verwendung finden, deren Einschränkungen bezüglich des Wertebereichs der Parameter unterschiedlicher Natur sind, wird in der Regel zwischen drei Typen unterschieden.
Die ursprüngliche Form des CCD wird dabei auch als „Central Composite Circumscribed“ (CCC) bezeichnet, dessen Sternpunkte allerdings neue Extremwerte annehmen. Das Setzen eines Parameters auf eine Stufe außerhalb seines zulässigen Wertebereichs ist jedoch nicht immer möglich. Daher existiert die Variante des „Central Composite Inscribed“ (CCI), welche das CCC auf den zulässigen Wertebereich umskaliert und die Sternpunkte auf die Grenzen der Wertebereiche (Innkreis) legt. Beide Typen setzen mindestens 5 Stufen je Parameter voraus. Ist dies nicht gewährleistet kann noch auf das „Central Composite Face Centered“ (CCF) Design zurückgegriffen werden. Es legt die Sternpunkte auf die Mittelpunkte der Dimensionen des Variantenraums [NIS, Kapitel 5.3.3.6.1]. Abbildung 3.7 macht die Unterschiede zwischen den drei Typen deutlich.

Abbildung 3.8 verdeutlicht die Abdeckung des Variantenraums im Bezug auf das Beispiel aus Abschnitt 3. Es wurde auf die Variante des Innkreises (CCI) zurückgegriffen. Die eingezeichneten Linien weisen auf die Konstruktion des Versuchsplans hin.

3.6 Plackett-Burman Design

Wie bereits in Abschnitt 3.1 aufgezeigt, steigt die Anzahl der benötigten Versuchsläufe in einem zweistufigen vollfaktoriellen Design mit der Zahl der Parameter schnell an. Erlaubt man zusätzlich drei oder mehr Stufen je Parameter, ist eine Durchführung aufgrund des Aufwands oft gar nicht mehr möglich. Im Wissen darüber haben Plackett und Burman multifaktorielle Versuchspläne erarbeitet und 1946 vorgestellt [PB46]. Die Pläne kommen mit nur einer begrenzten Auswahl der Versuchsläufe eines vollfaktoriellen Designs aus. Ziel ist es zudem, trotz der Reduzierung des vollständigen Versuchsplans, den Einfluss und die Interaktion der Parameter mit der selben Genauigkeit zu bestimmen, wie es bei Einzelbetrachtung der Parameter möglich ist [PB46].

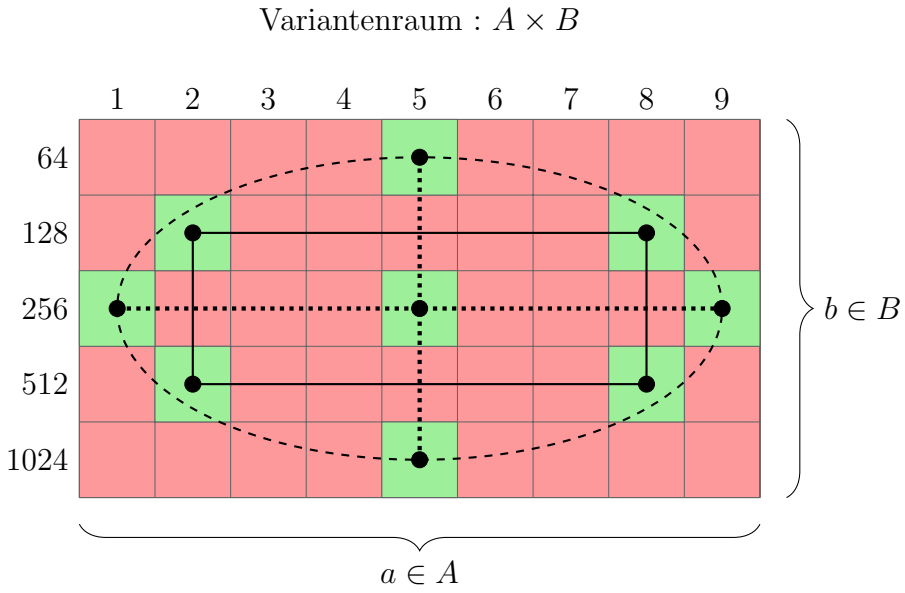


Abbildung 3.8: Abdeckung des Variantenraumes durch das CCI Design

Die in diesen Versuchsplänen verfolgte Idee besteht hauptsächlich darin, zur Erkennung der Zusammenhänge nur auf die minimalste Anzahl von Messungen zurückzugreifen. Betrachtet man die Eingabeparameter und die zugehörige Ausgabe als lineares Gleichungssystem, so erfordert die eindeutige Bestimmung von k unbekannten Koeffizienten nur genau $k+1$ Versuche [PB46]. Zusätzliche Versuche lassen keine eindeutige Bestimmung der Koeffizienten mehr zu, es muss auf die Methode der kleinsten Quadrate zurückgegriffen werden [PB46].

Der im Kontext dieses Designs verwendete Bezeichner N steht für die Anzahl der Versuche im jeweiligen Versuchsplan. Ein Versuchsplan nach Plackett und Burman erlaubt dabei die Untersuchung von $N-1$ Parametern. Der Bezeichner L steht für die Anzahl der untersuchten Stufen in einem Versuchsplan. Mittels des Seeds mit $N = 27$ und $L = 3$ (Tabelle 3.7) lässt sich also ein Design für bis zu 26 Parameter auf je drei Stufen realisieren. Die Ziffern von 0 bis $L-1$ der Seeds beschreiben abstrakt die zu untersuchenden Stufen des jeweiligen Parameters.

Zur Konstruktion eines Versuchsplans nach Plackett und Burman, wird zuerst ein passender Seed gewählt. Die verfügbaren Varianten finden sich in Tabelle 3.7. Der Seed bildet die erste Spalte der (vorerst) quadratischen Designmatrix. Die erste Spalte wird anschließend spaltenweise zyklisch permutiert, bis eine $(N-1) \times (N-1)$ Matrix entsteht. Zuletzt wird der daraus resultierenden Matrix eine Zeile mit Nullen hinzugefügt [PB46].
Generiert man ein Design basierend auf dem Seed für $N = 9$ und $L = 3$, erhält man die Designmatrix aus Tabelle 3.8. Es lässt die Untersuchung von bis zu 8 Parametern zu.

Sollen weniger als $N-1$ Parameter untersucht werden, wird nur auf eine Teilmenge (im Bezug auf die Spalten) der Designmatrix zurückgegriffen. Wendet man dies im Falle der Beispielapplikation auf die Parameter a und b an, ergibt sich der in

N	L	Seed
9	3	01220211
27	3	00101 21120 11100 20212 21022 2
81	3	01111 20121 12120 20221 10201 10012 22021 00200 02222 10212 21210 10112 20102 20021 11012 00100
25	5	04112 10322 42014 43402 3313
125	5	02221 04114 13134 12021 10244 31402 00444 20322 32121 32404 22043 31230 40033 34014 41424 21430 34403 11241 03001 11302 33234 34231 01330 12243 2010
49	7	01262 21605 32335 20413 11430 65155 61024 54425 03646 634

Tabelle 3.7: Seeds zur Generierung eines Plackett-Burman Designs für jeweils $N-1$ Parameter auf L Stufen - vgl. [PB46]

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
0	0	1	1	2	0	2	2	1
1	1	0	1	1	2	0	2	2
2	2	1	0	1	1	2	0	2
3	2	2	1	0	1	1	2	0
4	0	2	2	1	0	1	1	2
5	2	0	2	2	1	0	1	1
6	1	2	0	2	2	1	0	1
7	1	1	2	0	2	2	1	0
8	0	0	0	0	0	0	0	0

Tabelle 3.8: Versuchsplan für (bis zu) 8 Parameter unter Verwendung des Seeds mit $N = 9$ und $L = 3$

Abbildung 3.9 visualisierte Variantenraum. Gewählt wurden die Spalten x_1 und x_2 entsprechend für die Parameter a und b.

3.7 Optimale Designs

Die bisher vorgestellten experimentellen Desgins haben viele Vorteile im Vergleich zu vollständigen Versuchsplänen. Dennoch bergen auch diese Designs unterschiedliche Nachteile [Pet91, Kapitel 8]. So müssen bei zweistufigen faktoriellen Designs

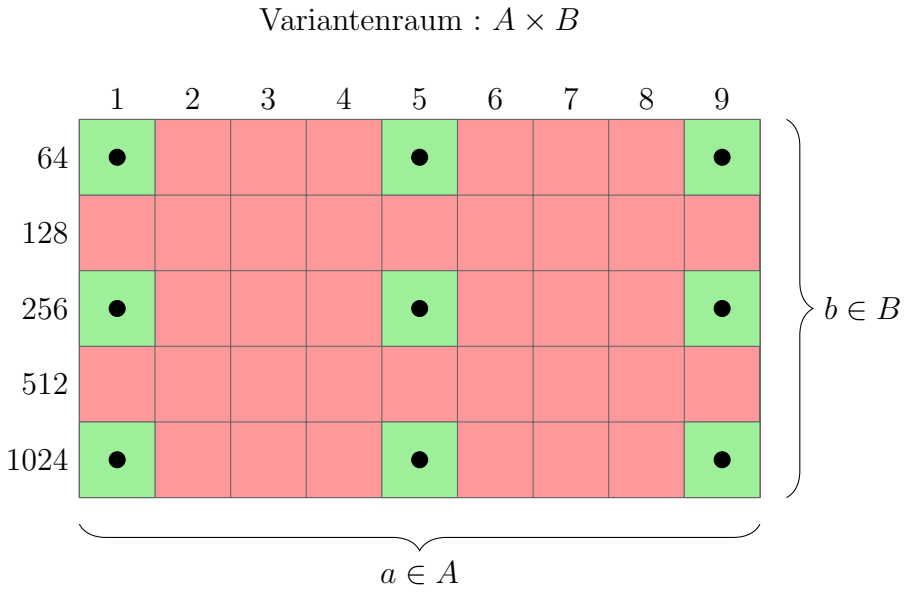


Abbildung 3.9: Abdeckung des Variantenraumes - Plackett-Burman Design

mit k Faktoren alle 2^k Versuche durchgeführt werden. Ein solches Vorgehen wird also mit wachsender Faktorenzahl k sehr schnell unpraktikabel. Zudem erlaubt es die Abstraktion auf zwei Stufen nur Rückschlüsse auf lineare Zusammenhänge zu ziehen. Um Krümmungen indentifizieren zu können, werden Designs auf drei Stufen erweitert, was aber auch hier die Zahl der nötigen Versuche rasch steigen lässt. Versuchspläne nach Box-Behnken und zentral zusammengesetzte Designs setzen zwar den Messaufwand und die Güte des errechneten Modells in ein vertretbares Verhältnis, geben aber Parameterstufen und Versuchszahl fest vor [Pet91, Kapitel 8].

Diesen Nachteilen begegnen optimale Designs, indem sie beispielweise bei der Festlegung der Versuchsanzahl, der Stufen oder sogar der Stufenabstände freie Wahl lassen. Zudem können gezielt Parameterkonfigurationen und bereits existierende Ergebnisse implementiert sowie der Versuchsraum eingeschränkt (irreguläre Versuchsfelder) werden [Pet91, Kapitel 8].
Optimal bedeutet in diesem Zusammenhang, das „beste“ Design im Hinblick auf ein bestimmtes Kriterium zu sein [Mon05, Kapitel 11-4.4]. Zur Generierung der Designs wird in der Regel auf die Hilfe von Programme zurückgegriffen. Grundsätzlich existieren eine Vielzahl von Optimalitätskriterien, bei denen sich aber im Folgenden mit A-, D-, G- und V-Optimalität auf die häufigsten Vertreter beschränkt wird.

Kandidatenmatrix und Modellmatrix

Zum Verständnis der Kriterien sollten zudem die Begriffe Kandidatenmatrix, Informationsmatrix und Dispersionsmatrix geläufig sein.

Die Kandidatenmatrix ξ_N beinhaltet alle Kandidatenpunkte, also die Menge der zu untersuchenden Konfigurationen. In dieser Matrix repräsentiert jede Zeile eine Konfiguration, wobei jede Spalte einem Parameter entspricht. Die Länge der Matrix beträgt N [dABK+95].
Im Bezug auf das zu Beginn des Kapitels eingeführte Beispiel mit den Parametern

a, b und c und der Betrachtung von jeweils zwei Stufen ergibt sich für die Kandidatenmatrix ξ_N und $N = 2^3 = 8$:

$$\xi_8 = \begin{pmatrix} - & - & - \\ + & - & - \\ - & + & - \\ + & + & - \\ - & - & + \\ + & - & + \\ - & + & + \\ + & + & + \end{pmatrix} \tag{3.3}$$

Ausgehend von dieser Kandidatenmatrix können Untermengen ξ_n mit n Punkten ($n \leq N$) gebildet werden [dABK+95]. Beispiel für eine Untermenge ξ_4 der oben abgebildeten Kandidatenmatrix ist:

$$\xi_4 = \begin{pmatrix} - & - & - \\ + & - & - \\ - & + & - \\ + & + & - \end{pmatrix} \tag{3.4}$$

Basierend auf einem Kandidatenset ξ_n kann die $(n \times p)$-Modellmatrix X konstruiert werden [dABK+95].

Informations- und Dispersionsmatrix

Die Informationsmatrix ist das Produkt aus der Modellmatrix X und ihrer Transponierten[1] X'. Für den nun folgenden Abschnitt sei die Informationsmatrix $(X'X)$ und deren Inverses mit $(X'X)^{-1} = [d_ij]$ (Dispersionsmatrix) gegeben [dABK+95].

A-Optimalität

Ein Design ist A-optimal, wenn es die Summe der Elemente auf der Hauptdiagonalen der Dispersionsmatrix $(X'X)^{-1}$ minimiert. Die Summe der Elemente der Hauptdiagonalen wird in der Mathematik als „Spur“ der Matrix $(X'X)^{-1}$ bezeichnet und für gewöhnlich mit $tr(X'X)^{-1}$ formalisiert:[2]

$$tr(X'X)^{-1} = \sum_{i=0}^{n} d_{ii} \tag{3.5}$$

Ein A-optimales Design minimiert also die Summe der Varianzen der Regressionskoeffizienten [Kie59] [Mon05, Kapitel 11-4.4].
Formal definiert sich ein A-optimales Design X^* durch [dABK+95]:

$$tr(X^{*\prime}X^*)^{-1} = \min_{\xi_n \Xi_N} tr(X'X)^{-1} \tag{3.6}$$

[1] Transposed matrix. Encyclopedia of Mathematics. http://www.encyclopediaofmath.org/index.php?title=Transposed_matrix (06.04.2014)

[2] Trace of a square matrix. Encyclopedia of Mathematics. http://www.encyclopediaofmath.org/index.php?title=Trace_of_a_square_matrix (06.04.2014)

D-Optimalität

Ein Design ist D-optimal, wenn es die Determinante der Dispersionsmatrix $|(X'X)^{-1}|$ minimiert [Mon05, Kapitel 11-4.4]. Analog dazu ist die Maximierung der Determinante der Informationsmatrix [dABK+95]:

$$|(X'X)| = \frac{1}{|X'X|^{-1}} \tag{3.7}$$

Formal definiert sich ein D-optimales Design X^* demnach durch [dABK+95]:

$$|(X^{*\prime}X^*)^{-1}| = \min_{\xi_n \Xi_N} |(X'X)^{-1}| \tag{3.8}$$

bzw.

$$|(X^{*\prime}X^*)| = \max_{\xi_n \Xi_N} |(X'X)| \tag{3.9}$$

Einfach gesagt, ist mit einer maximalen Determinante der Informationsmatrix auch die enthaltene Information maximal [SvBH10, Kapitel 2.6.1]. Das D-Optimalitätskriterium ist das wohl am häufigsten verwendete Kriterium. Der in Abschnitt 3.8 näher beschriebene „k-Exchange Algorithmus“ bedient sich ihm ebenfalls.

G-Optimalität

Ein Design ist G-optimal, wenn es die maximale Varianz des Designs minimiert [Mon05, Kapitel 11-4.4].
Die Varianz $d(x)$ einer Konfiguration x wird definiert durch [dABK+95]:

$$d(x) = x'(X'X)^{-1}x \tag{3.10}$$

Formal definiert sich ein G-optimales Design X^* demnach durch [Mon05, Kapitel 11-4.4]:

$$\max(x'(X^{*\prime}X^*)^{-1}x) = \min_{\xi_n \Xi_N}(\max(x'(X'X)^{-1}x)) \tag{3.11}$$

Sind bei der Suche nach einem D-optimalen Design zwei Modelle ähnlich, können andere Optimalitätskriterien die Auswahl erleichtern [NIS, Kapitel 5.5.2.1]. Häufig geschieht dies unter Betrachtung der G-Effizienz eines Designs [dABK+95]:

$$G_{eff} = \frac{p}{d_{max}(x) * n} \tag{3.12}$$

Wobei $d_{max}(x)$ die maximale Varianzfunktion, p die Anzahl der Parameter und n die Anzahl der Versuche im Design ist.

V-Optimalität

Ein Design ist V-optimal, wenn es die durchschnittliche Varianz über eine Menge von m Versuchsläufen minimiert [Mon05, Kapitel 11-4.4]. Die Punktmenge kann dabei den Kandidatenpunkten oder aber auch einer beliebigen anderen Punktmenge von Belang entsprechen [Mon05, Kapitel 11-4.4].
Formal definiert sich ein V-optimales Design X^* demnach durch [Mon05, Kapitel 11-4.4]:

$$\frac{\sum_{i=1}^{m} x_i'(X^{*\prime}X^*)^{-1}x_i}{m} = \min_{\xi_n \Xi_N} \left(\frac{\sum_{i=1}^{m} x_i'(X'X)^{-1}x_i}{m} \right) \tag{3.13}$$

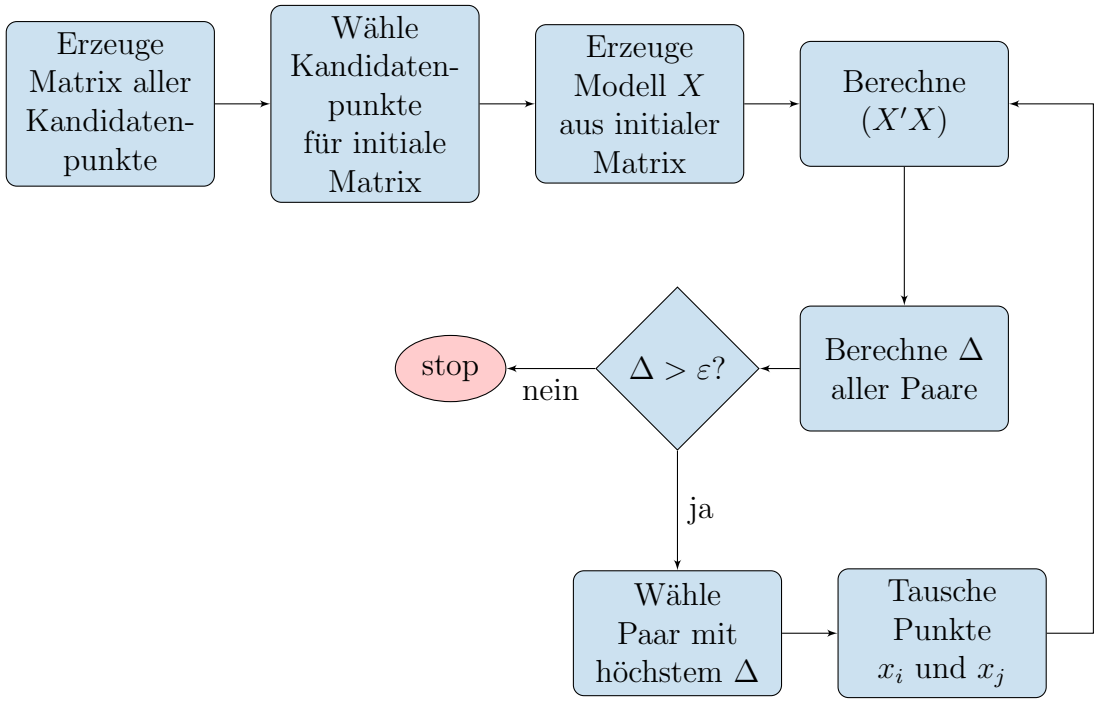


Abbildung 3.10: Ablaufdiagramm des Fedorov-Algorithmus - vgl. [dABK+95]

3.8 D-Optimales Design mittels des k-Exchange Algorithmus

Das D-Optimalitätskriterium für sich erzeugt noch kein eigenständiges Design. Vielmehr sind Computer nötig, die die aufwendige Erzeugung eines D-optimalen Versuchsplans übernehmen. Zur softwaretechnischen Realisierung wird hier auf den "k-Exchange Algorithmus" zurückgegriffen. Der k-Exchange Algorithmus orientiert sich stark am Fedorov-Algorithmus. Dies wird deutlich, wenn man die Spezialfälle des k-Exchange Algorithmus mit $k = 1$ („Van Schalkwyk") und den (modifizierten) „Fedorov-Algorithmus" mit $k = N$ betrachtet [JN83].

Der Ablauf des Fedorov-Algorithmus gestaltet sich wie in Abbildung 3.10 dargestellt. Man beginnt mit der Erzeugung einer Matrix die alle Kandidatenpunkte enthält. Dieser Vorgang ist gleichzusetzen mit der Erstellung eines vollständigen Versuchsplans, was in hochkonfigurierbaren Systemen schnell zum begrenzenden Faktor wird (Abschnitt 4.5). Anschließend sind daraus N Punkte beliebig zu wählen, welche die (initiale) Designmatrix bilden. Daraus wird die für die weiteren Berechnungen verwendete Modellmatrix X erzeugt. Der eigentlichen Algorithmus beginnt nun mit der Berechnung der Informationsmatrix $(X'X)$. Es folgt die Berechnung des Δ aller Paare aus dem vollständigen Design und der Designmatrix. Überschreitet kein Δ eines Paares einen bestimmten Schwellenwert ε (Abbruchkriterium) endet der Algorithmus. Ist dies nicht der Fall, wird das Paar gewählt, welches das größte Δ aufweist und fortgefahren. Der Kandidatenpunkt x_i aus dem vollständigen Versuchsplan wird in die Designmatrix an die Stelle x_j getauscht und der Algorithmus beginnt wieder bei der (Neu-)Berechnung der Informationsmatrix [dABK+95].
Der k-Exchange Algorithmus adaptiert dieses Vorgehen grundlegend, unterscheidet

sich aber in einem Punkt erheblich. Statt nur das Paar mit dem höchsten Δ zu betrachten, werden k Paare mit den höchsten Δ betrachtet [dABK+95].

Pseudocode 1 macht die Funktionsweise des k-Exchange Algorithmus deutlich und die Struktur des Fedorov-Algorithmus (Abbildung 3.10) erkennbar. Der Unterschied dieser Variante ist erstmals in Zeile 4 mit Selektion der k Punkte mit niedrigster Varianz ersichtlich. Diese k Elemente werden dann im weiteren Verlauf des Algorithmus jeweils als Tauschkandidaten betrachtet.

```
1  create random design with desired number of points;
2  while couples with positive delta are found do
3      calculate variance of prediction d(x_i) for all design points;
4      select k design points with lowest variance of prediction;
5      for design point x_1 to design point x_k do
6          calculate variance of prediction d(x_i) for this design point;
7          for support point x_1 to support point x_N do
8              calculate variance of prediction d(x_j) for this support point;
9              calculate variance function d(x_i, x_j) for this couple;
10             calculate delta function Δ(x_i, x_j) for this couple;
11             select couple with maximum delta;
12             if maximum delta is positive then
13                 if more then one couple with same maximum delta then
14                     select couple randomly;
15                 end
16                 exchange selected point x_i with x_j;
17                 update information and dispersion matrix;
18                 reset maximum delta;
19             end
20         end
21     end
22 end
```

Pseudocode 1 : k-Exchange Algorithmus [Tri08, Algorithm 3.4]

Anzahl der Versuchsläufe N und die Wahl des k

Die Anzahl der Versuchsläufe N ist nicht an spezielle Beschränkungen gebunden, dennoch empfiehlt es sich, sich über dessen Wahl ein paar Gedanken zu machen. Die einzige Empfehlung hinsichtlich der Wahl des N ist im Bezug auf eine untere Grenze auszusprechen. Denn mathematisch gesehen erfordert die Untersuchung von p Parametern mindestens p Versuchsläufe. Betrachtet man die Eingabeparameter und den draus resultierenden Ausgabeparameter als Gleichungssystem, erhält man mit weniger als p Experimenten ein unterbestimmtes Gleichungssystem.

Zur Wahl des k existieren empirische Daten, die in der Regel

$$k \leq \frac{N}{4} \tag{3.14}$$

als ausreichend groß betrachtet, um ein mit dem modifizierten Fedorov-Algorithmus vergleichbares Design zu liefern [MN95].

3.9 Zusammenfassung der experimentellen Designs

Tabelle 3.9 zeigt eine Zusammenfassung der vorgestellten experimentellen Designs und deren Eigenschaften hinsichtlich unterstützter Parameterzahl und -stufen sowie den enthaltenen Versuchsläufen. Zu erkennen ist die Beschränkung des Box-Behnken und der Plackett-Burman Designs im Bezug auf die Parameterzahl, mit maximal 16 bzw. 124 unterstützten Parametern. Von einer Abstraktion der vorhandenen Stufen kann oder wird nur beim vollfaktoriellen Design und dem dem durch den k-Exchange Algorithmus generierten D-optimalen Design abgesehen. Die Untersuchung nichtlinearer Zusammenhänge ist aber, abgesehen von den zweistufen Versuchsplänen, überall möglich. Im Hinblick auf die Anzahl der Versuchsläufe sind bis auf die Box-Behnken, die Plackett-Burman und die mittels des k-Exchange Algorithmus erzeugten Design alle direkt abhängig von der Anzahl der zu untersuchenden Parametern.

	Vollfaktoriell	2^k	2^{k-1}	Box-Behnken	CCI	Plackett-Burman	k-Exchange
Parameter	k	k	k	3 bis 16	k	$k \leq 124$	k
Stufen	l_i, $i \in [1, ..., k]$	2	2	3	5	3, 5 oder 7	l_i, $i \in [1, ..., k]$
Versuchsläufe	$\prod_{i=1}^{k} l_i$	2^k	2^{k-1}	13 bis 385	$2^k + 2k + 1$	9 bis 125	beliebig

Tabelle 3.9: Vergleich ausgewählter experimenteller Design unter den Gesichtspunkten der unterstützten Parameteranzahl, der Parameterstufen und der Anzahl der enthaltenen Versuchsläufe

4. Realisierung

In diesem Kapitel wird auf die implementierten Versuchspläne, das für die Simulation verwendete Tool und den Gegenstand der Experimente eingangen. Es dient zur Erläuterung der Rahmenbedingungen, unter denen anschließend die Experimente durchgeführt und die experimentellen Designs evaluiert werden.

4.1 Implementierung der Versuchspläne

Die Programmierung der Generatoren für die experimentellen Designs wurden zwar für den „SPL Conqueror“ vorgenommen, ist aber im Kern als universell zu betrachten. Sie implementieren dazu ein für diesen Zweck geschaffenes Interface, welches Methoden zur Designgenerierung und zur anschließenden Rückgabe eines auf den metrischen Konfigurationsoptionen basierenden Versuchsplans vorsieht.
Bei allen Umsetzungen wurde dabei auf niedrigen Hauptspeicher- und Rechenleistungsbedarf geachtet. Erreicht wird dies sowohl durch Abstraktion von den Parameterwerten bei der eigentlichen Generierung der Designs, als auch durch Optimierungen im Bereich der Speicherzugriffe und der verwendeten Speicherstrukturen. Der Pseudocode beschreibt jeweils nur sehr grob die Vorgehensweise, für einen detailierten Einblick wird auf die in `C#` realisierten Implementierungen im Anhang verwiesen.
Berücksichtigt wurden im Folgenden das 2^{k-1}-teilfaktorielle Design, das CCI, das Box-Behnken Design, das Plackett-Burman Design, sowie ein D-Optimales Design. Letzteres wird mit Hilfe des k-Exchange Algorithmus erzeugt.

4.1.1 2^{k-1}-teilfaktorielles Design

Zur effizienten Erzeugung der zweistufigen Designmatrix wird auf die Standardreihenfolge zurückgegriffen. Die Standardreihenfolge definiert sich dabei im Falle eines allgemeinen 2^k Designs wie folgt:
Die erste Spalte (x_1) beginnt mit einer -1 und wechselt mit jedem der $2k$ Versuchsläufe das Vorzeichen. Die zweite Spalte (x_2) beginnt mit der zweimaligen Wiederholung von -1 und ändert dann jeweils alle zwei Versuchsläufe das Vorzeichen. Die

dritte Spalte (x_3) beginnt mit viermaliger Wiederholung von -1, gefolgt von vier Wiederholungen der $+1$ und so weiter. Allgemein gesprochen beginnt also die i-te Spalte (x_i) mit $2i-1$ Wiederholungen der -1, gefolgt von $2i-1$ Wiederholungen von $+1$. [NIS, Kapitel 5.3.3.3.1]
Ein Beispiel für das Ergebnis dieses Vorgehens kann in Tabelle 3.3 eingesehen werden. Die Spalten x_i entsprechenden hier je einem der Parameter a und b und damit den vorhandenen metrischen Konfigurationsoptionen.

Zur Generierung des Designs (Pseudocode 2) wird zunächst eine Matrix erzeugt (Zeile 1), deren Spaltenanzahl durch die k Parameter definiert ist. Die Zeilenanzahl entspricht den enthaltenen Konfigurationen und beträgt 2^{k-1}. Die Designmatrix wird nun spaltenweise durchlaufen (Zeile 2) und die Werte entsprechend der Standardreihenfolge gesetzt (Zeile 7). Die Werte der letzten Spalte werden abschließend durch das Produkt der vorhergehenden Spalten bestimmt (Zeile 5).

```
   Input : num_vf
   Output : designmatrix
 1 lege designmatrix[2^(num_vf-1), num_vf] an;
 2 foreach Spalte der designmatrix do
 3 |   foreach Zeile der designmatrix do
 4 |   |   if letzte Spalte then
 5 |   |   |   setzte den Wert auf Produkt der vorhergehenden Spalten;
 6 |   |   else
 7 |   |   |   setzte den Wert entsprechend der Standardreihenfolge;
 8 |   |   end
 9 |   end
10 end
11 return designmatrix
```

Pseudocode 2 : Funktionsweise des Generators für ein 2^{k-1}-teilfaktorielles Design

Anschließend ermöglicht es eine Funktion den auf der generierten Designmatrix basierenden Versuchsplan zu erhalten (Pseudocode 3). Es werden die Stufen + und − auf das Maximum beziehungsweise Minimum des zur jeweiligen metrischen Konfigurationsoption gehörigen Wertebereichs gemappt.
Dazu wird die Designmatrix zeilenweise durchlaufen (Zeile 1). Jede dieser Zeilen repräsentiert eine zu testende Konfiguration (Zeile 2). Da jede Spalte der Designmatrix einem Parameter entspricht, werden deren Werte auf die Werte der Parameter abgebildet (Zeile 3-9). Findet sich im Design an der Stelle eine -1 wird der minimale Wert des Parameters zum Versuchslauf hinzugefügt (Zeile 4-5), andernfalls der ma-

ximale Wert (Zeile 7). Jeder der Versuchsläufe wird zum Versuchsplan hinzugefügt (Zeile 10), welcher abschließend von der Methode zurückgegeben wird (Zeile 12).

```
   Input : designmatrix, parameters
   Output : runs
 1 foreach Zeile der designmatrix do
 2 |   erzeuge run;
 3 |   foreach Parameter in parameters do
 4 |   |   if Wert in der designmatrix == -1 then
 5 |   |   |   füge mininmalen Wert des Parameters zu run hinzu;
 6 |   |   else
 7 |   |   |   füge maximalen Wert des Parameters zu run hinzu;
 8 |   |   end
 9 |   end
10 |   füge run zu runs hinzu;
11 end
12 return runs
```

Pseudocode 3 : Mapping des 2^{k-1}-teilfaktoriellen Designs auf die Parameter

4.1.2 Box-Behnken Design

Zur Generierung eines Box-Behnken Designs (Pseudocode 4) wird zuerst die für die Zahl der vorliegenden Parameter passende Seedmatrix ausgewählt (Zeile 1). Anschließend wird eine Submatrix generiert (Zeile 2), die einen 2^i zweistufigen faktoriellen Versuchsplan darstellt - wobei i die Anzahl der mit „1“ belegten Felder in den Zeilen der Seedmatrix beschreibt. Es wird eine Designmatrix angelegt, deren Zeilenzahl den Zeilen der Seedmatrix (ohne Mittelpunktsversuch) mal der Länge der Submatrix entspricht - zusätzlich ist ein Mittelpunktsversuch vorgesehen (Zeile 3). Die Seedmatrix wird nun zeilenweise durchlaufen und für die mit „1“ belegten Felder die entsprechenden Spalte der Submatrix in die Designmatrix geschrieben.

```
   Input : num_vf
   Output : designmatrix
 1 wähle die passende Seedmatrix in Abhängigkeit von num_vf;
 2 erzeuge die zur Seedmatrix gehörige Submatrix;
 3 lege designmatrix[(Seedmatrix.Length - 1) * Submatrix.Length + 1, num_vf]
   an;
 4 foreach Zeile der Seedmatrix do
 5 |   foreach Spalte der Seedmatrix do
 6 |   |   if Wert in Seedmatrix == 1 then
 7 |   |   |   hänge entsprechende Spalte der Submatrix an designmatrix an;
 8 |   |   end
 9 |   end
10 end
11 return designmatrix
```

Pseudocode 4 : Funktionsweise des Generators für ein Box-Behnken Design

Die drei Stufen der Designmatrix werden abschließend jeweils auf den minimalen, maximalen oder Mittelpunktswert der Parameter gemappt.

```
   Input : designmatrix, parameters
   Output : runs
 1 foreach Zeile der designmatrix do
 2 |   erzeuge run;
 3 |   foreach Parameter in parameters do
 4 |   |   if Wert in der designmatrix == 0 then
 5 |   |   |   füge Mittelwert des Parameters zu run hinzu;
 6 |   |   else if Wert in der designmatrix == -1 then
 7 |   |   |   füge mininmalen Wert des Parameters zu run hinzu;
 8 |   |   else
 9 |   |   |   füge maximalen Wert des Parameters zu run hinzu;
10 |   |   end
11 |   |   füge run zu runs hinzu;
12 |   end
13 end
14 return runs
```

Pseudocode 5 : Mapping des Box-Behnken Designs auf die Parameter

4.1.3 Central Composite Design (Inscribed)

Zu Beginn der Generierung eines Central Composite Designs (Pseudocode 6) muss mit α der Radius des Kreises festgelegt werden (Zeile 1), auf dem die Sterpunktsversuche liegen. Der Wert wird dabei auf

$$\alpha = 2^{\frac{k}{4}} \tag{4.1}$$

festgelegt [NIS, Kapitel 5.3.3.6.1], also die vierte Wurzel aus der Anzahl der Versuche in einem zweistufigen Design. Anschließend wird die Designmatrix für die Sternpunktsversuche angelegt, die je zwei Versuche je Parameter vorsieht (Zeile 2). Es wird nun in jede Zeile der Matrix ein Sternpunktsversuch geschrieben (Zeile 4), welcher im Bezug auf den Zentralpunkt entweder auf $+\alpha$ oder $-\alpha$ liegt - die anderen Parameter verbleiben jeweils in ihrem Mittelwert. Abschließend wird die Designmatrix noch um den Zentralpunktsversuch erweitert (Zeile 6) und zurückgegeben.
Wie zu erkennen ist, ist diese Vorgehensweise universell anwendbar für die Generierung der verschiedenen Typen des Central Composite Designs. Die Festlegung der

tatsächlichen Punkte im Variantenraum erfolgt erst im anschließenden Mapping der Werte der Designmatrix auf die Werte der Parameter.

```
   Input : num_vf
   Output : designmatrix
1  berechne α;
2  lege designmatrix[2 ∗ num_vf, num_vf] an;
3  foreach Zeile der designmatrix do
4  |  setze Sternpunktsversuche in Abhängigkeit von α;
5  end
6  füge Zeile mit Mittelpunktsversuch zur designmatrix hinzu;
7  return designmatrix
```

Pseudocode 6 : Funktionsweise des Generators für ein „Central Composite Inscribed"-Design

Die Umwandlung in den Typ des „Central Composite Inscribed" erfolgt erst unmittelbar vor Rückgabe der Werte (Pseudocode 7). Zu Beginn werden dem Versuchsplan die Versuche eines 2^k Designs hinzugefügt (Zeile 1). Da die Sternpunkte - wie in Abschnitt 3.5 beschrieben - außerhalb des Wertebereichs liegen, werden die Punkte des Versuchsplans in Abhängigkeit von α umskaliert (Zeile 2). Anschließend werden die Sternpunktsversuche aus der Designmatrix an den Versuchsplan angehängt. Die drei Stufen der Designmatrix $-\alpha$, $+\alpha$ und 0 werden jeweils auf den minimalen, maximalen oder den Mittelpunktswert der Parameter gemappt.
Abgesehen vom hier beschriebenen Verfahren, kann ebenso auf einfache Weise die Erstellung der anderen Typen implementiert werden.

```
   Input : designmatrix, parameters
   Output : runs
1  füge Konfigurationen eines 2^k-Designs zu runs hinzu;
2  skaliere bestehende Konfigurationen in Abhängigkeit von α um;
3  foreach Zeile der designmatrix do
4  |  erzeuge run;
5  |  foreach Parameter in parameters do
6  |  |  if Wert in der designmatrix == 0 then
7  |  |  |  füge Mittelwert des Parameters zu run hinzu;
8  |  |  else if Wert in der designmatrix < 0 then
9  |  |  |  füge mininmalen Wert des Parameters zu run hinzu;
10 |  |  else
11 |  |  |  füge maximalen Wert des Parameters zu run hinzu;
12 |  |  end
13 |  end
14 |  füge run zu runs hinzu;
15 end
16 return runs
```

Pseudocode 7 : Mapping des „Central Composite Inscribed"-Designs auf die Parameter

4.1.4 Plackett-Burman Design

Zu Beginn der Generierung eines Plackett-Burman Designs (Pseudocode 8) wird der zu verwendende Seed (siehe Tabelle 3.7) in Abhängigkeit von der Anzahl der Parameter und deren Stufen gewählt (Zeile 1). Anschließend wird die Designmatrix angelegt (Zeile 2), deren Spaltenanzahl durch die Anzahl k der Parameter festgelegt ist. Die Zeilenzahl entspricht der Länge des Seeds plus eine Zeile für den Versuch auf den jeweils niedrigsten Stufen der Parameter. Es wird nun der Seed in die erste Spalte der Designmatrix eingetragen (Zeile 3) und die restlichen Spalten der Matrix durch zyklische Permuation der jeweils vorhergehenden Spalten gefüllt (Zeile 5). Die letzte Zeile der Designmatrix wird abschließend auf „0“ gesetzt (Zeile 7) und das Resultat zurückgegeben.

```
   Input : num_vf
   Output : designmatrix
1  wähle den passenden Seed in Abhängigkeit von num_vf und den Stufen;
2  lege designmatrix[Seed.Length + 1, num_vf] an;
3  trage Seed in erste Spalte der designmatrix ein;
4  foreach Spalte der designmatrix do
5  |  fülle Spalte mit zyklischer Permutation der vorhergehenden Spalte;
6  end
7  setzte die letzt Zeile der Designmatrix auf 0;
8  return designmatrix
```

Pseudocode 8 : Funktionsweise des Generators für ein Plackett-Burman Design

Vor Erstellung des Versuchsplans aus der Designmatrix (Pseudocode 9) werden die Stufen des verwendeten Seeds den tatsächlichen Stufen der Parameter zugeordnet (Zeile 1) und hinterlegt. Der niedrigste bzw. höchste Seedwert entspricht dem minimalen bzw. maximalen Wert des Parameters, die verbleibenden Seedwerte werden entsprechend dem tatsächlichen Wertebereich skaliert. Bei der Festlegung der Versuche wird anschließend auf das vorgehaltene Mapping zurückgegriffen.

```
   Input : designmatrix, parameters
   Output : runs
1  erstelle Zuordnung der Seedstufen auf die Stufen der Parameter;
2  foreach Zeile der designmatrix do
3  |  erzeuge run;
4  |  foreach Parameter in parameters do
5  |  |  füge Wert entsprechend der Zuordnung zu run hinzu;
6  |  end
7  |  füge run zu runs hinzu;
8  end
9  return runs
```

Pseudocode 9 : Mapping des Plackett-Burman Designs auf die Parameter

4.1.5 D-Optimales Design mittels des k-Exchange Algorithmus

Bei der Implementierung des k-Exchange Algorithmus wurde in erster Linie auf den in Abschnitt 3.8 abgebildeten Pseudocode zurückgegriffen sowie die von F. Triefenbach verwendete Programmstruktur adaptiert [Tri08, C++ Listing 4.3]. Da hier auf die redundante Abbildung verzichtet wird, sei für einen detailierten Einblick auf die im Anhang befindliche Implementierung verwiesen.
Das anschließende Mapping der Designmatrix auf die Parameterstufen gestaltet sich grundlegend wie in Pseudocode 9 beschrieben.

4.2 Integration der experimentellen Designs in SPL Conqueror

Die vorgestellten Implementierungen wurden in erster Linie für „SPL Conqueror“ vorgenommen. Anhand Abbildung 4.1 wird die Integration der experimentellen Designs veranschaulicht. Der Ablauf gestaltet sich wie folgt:

1. **Es wird ein Modell des konfigurierbaren Systems in „SPL Conqueror“ eingelesen.**
   Das Modell liegt in XML-Syntax vor und enthält eine Systembeschreibung mit Informationen zu den verfügbaren Parametern und deren Wertemengen, aber auch eventuelle Nebenbedingungen für ihr gemeinsames Auftreten.

2. **„SPL Conqueror“ gibt das interne Featuremodell an den Generator eines experimentellen Designs weiter.**
   Das übergebene Featuremodell ist die interne Repräsentation der eingelesenen Systembeschreibung und bündelt auch die Ergebnisse aus den späteren Untersuchungen.

3. **Der gewählte Generator erzeugt einen statistischen Versuchsplan und gibt diesen zurück an „SPL Conqueror“.**
   Das jeweilige experimentelle Design wird dabei unter Berücksichtigung der vorhandenen Parameter und deren Eigenschaften generiert. Der übergebene Versuchsplan ist damit eine Reihe von bereits vollständig konfigurierten Versuchsläufen.

4. **„SPL Conqueror“ testet das zu untersuchende System.**
   Dazu generiert er die im übermittelten Versuchsplan enthaltenen Varianten einer Software oder ruft das zu untersuchende Programm mit den entsprechenden Konfigurationsoptionen auf.

5. **„SPL Conqueror“ erfässt die Eigenschaft(en) der jeweiligen Variante.**
   Sofern er nicht selbst in der Lage ist die untersuchte(n) Eigenschaft(en) zu bemessen, delegiert er dies an eine externe Messapplikation. Das jeweils erhaltene Ergebnis wird abschließend im internen Featuremodell festgehalten und mit der Variante in Verbindung gesetzt.

6. **„SPL Conqueror“ lernt basierend auf den Varianteneigenschaften eine Funktion zur Abbildung des Systemverhaltens.**
   Basierend auf den Wertpaaren aus Schritt 5 ermittelt SPL Conqueror die Einflüsse einzelner Konfigurationsoptionen auf die nichtfunktionalen Eigenschaften. Es wird sich dabei verschiedener Lernverfahren, wie beispielsweise der Regressionsanalyse, bedient.

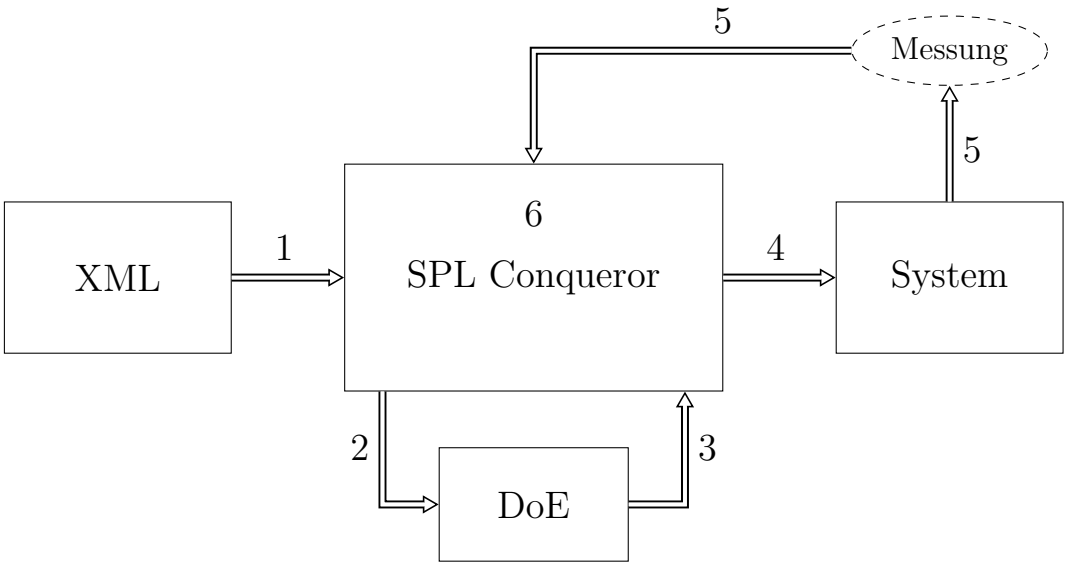


Abbildung 4.1: Veranschaulichung der Integration der experimentellen Designs in den SPL Conqueror

4.3 NFPMsimulator

Um eine performante und unverfälschte Analyse der ausgewählten Versuchspläne zu ermöglichen, wurde ein Hilfsprogramm implementiert. Der Begriff „unverfälscht“ soll hier darauf verweisen, dass von sämtlichen unkontrollierbaren Eingabeparametern abstrahiert wird. Ein Beispiel für einen solchen unkontrollierbaren Eingabeparameter ist beispielsweise die sich ändernde Grundauslastung eines Computers, während man versucht die Antwortzeiten einer Variante der SQLite-Datenbank zu testen. Das Akronym „NFPM“ steht für „non-functional property model“.
Der „NFPMsimulator“ stellt also einen „Simulator für nichtfunktionale Eigenschaften eines Programms“ dar. Das Tool ermöglicht die Simulation verschiedener Modelle auf Basis einer beliebigen Anzahl übergebener metrischer Optionen, die beschreibende Funktion ist bekannt.

Das Tool erlaubt die Wahl des zu simulierenden Modells aus den aktuell drei unterstützten Modellen (siehe Abschnitt 4.3.2). Die Modelle stellen hier einfache mathematische Funktionen dar und unterscheiden sich in ihrem Grad und ihren Parameterinteraktionen. Es ist modular aufgebaut und eine Erweiterung um beliebige Modelle grundsätzlich möglich. Um dies zu erreichen wurde ein zu implementierendes Interface definiert. Weiterhin wird die Übergabe einer beliebigen Anzahl von Parametern unterstützt.

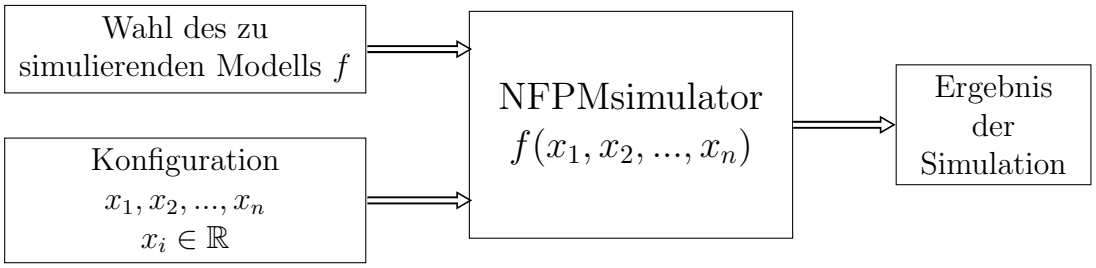


Abbildung 4.2: Schematische Darstellung des Testsystems mit dem NFPMsimulator

4.3.1 Gründe für die Verwendung des Simulators

Die Gründe für die Verwendung des Simulators liegen, wie bereits angesprochen, in erster Linie in der Kenntnis der zugrundeliegenden Funktion. Ohne dieses Wissen ist eine Bewertung der aus den ausgewählten Versuchsläufen abgeleiteten Funktion bezüglich der Genauigkeit nur schwer möglich, da das abzubildenden Verhaltensmodell realer Software selten bekannt ist. Abgebildet werden außerdem nur die übergebenen Parameter, dadurch werden unkontrollierbare Eingabeparamter vollständig ausgeblendet.
Zudem ist das Erzeugen der verschiedenen Programmvarianten eines konfigurierbaren Systems und das Messen der jeweiligen nichtfunktionalen Eigenschaft mit einem erheblichen Zeitaufwand und unter Umständen auch Messungenauigkeiten verbunden. Der Simulator stellt das Ergebnis hingegen unmittelbar zur Verfügung.
Zusätzlich liefert das Tool einen Generator für einen vollständigen Versuchsplan. Damit kann basierend auf dem Featuremodell, das auch dem „SPL Conqueror“ übergeben wurde, eine XML-Datei mit sämtlichen Ergebnissen erzeugt werden. Diese Ergebnisse sind vor allem für die spätere Beurteilung der errechneten Modelle von Bedeutung.

4.3.2 Unterstützte Modelle

Die vom NFPMsimulator unterstützten Funktionen orientieren sich in erster Linie an den gängigen Regressionsmodellen. Im Folgenden wird jeweils das zugrundeliegende Modell kurz beschrieben und der Pseudocode der Implementierung abgebildet. Zudem wird jedes Modell jeweils durch einen dreidimensionalen Plot visualisiert, dem die Parameter aus dem Beispiel in Abschnitt 3 als Datenquelle dienen.
Als Funktionsvariablen dienen ebenfalls die Parameter a, b und c der Beispielanwendung. Die übergebene Parameterliste entspricht also einer gewählten Konfiguration. Jedes Modell ist im Bezug auf die Parameterzahl beliebig erweiterbar. Die jeweils konstruierte Form ist aus dem zugehörigen Pseudocode abzuleiten.

Lineares Modell ohne Wechselwirkungen (LwoD)

Das „Lineare Modell ohne Wechselwirkungen“ simuliert eine Funktion der Form:

$$y = \beta_1 a + \beta_2 b + \beta_3 c \tag{4.2}$$

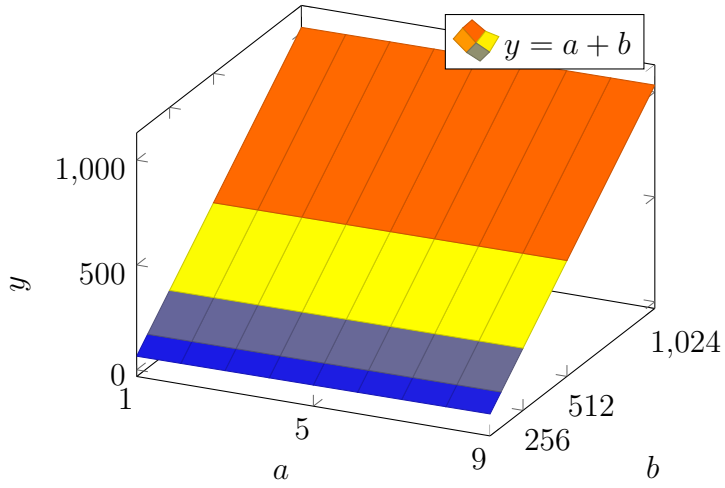


Abbildung 4.3: Dreidimensionaler Plot zur Veranschaulichung eines linearen Modells ohne Wechselwirkungen. Die Farben dienen lediglich zur Kenntlichmachung der Lage der Ebene.

Dazu wird jeder Parameter mit einem beliebigen Wert β_i multipliziert und die entstehenden Teilwerte aufsummiert (Pseudocode 10). Der Rückgabewert der Funktion ist das Ergebnis y der Gleichung.

```
   Data : Liste mit metrischen Parametern
   Result : result
1  foreach Parameter der Liste do
2  |  result += β_i * parameter;
3  end
4  return result
```

Pseudocode 10 : Lineare Funktion ohne Wechselwirkungen

Lineares Modell mit Wechselwirkungen (LwD)

Das „Lineare Modell mit Wechselwirkungen“ simuliert eine Funktion der Form:

$$y = \beta_1 ab + \beta_2 c \tag{4.3}$$

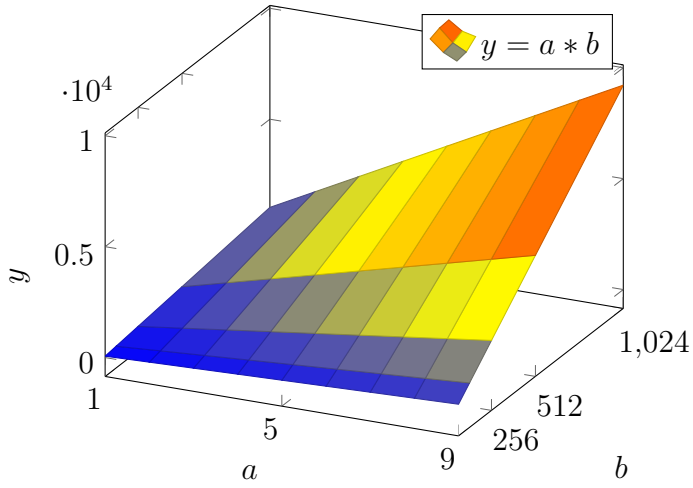


Abbildung 4.4: Dreidimensionaler Plot zur Veranschaulichung eines linearen Modells mit Wechselwirkungen. Die Farben dienen lediglich zur Kenntlichmachung der Lage der Ebene.

Dazu werden die Parameter paarweise multipliziert und die entstehenden Teilwerte aufsummiert (Pseudocode 11). Der Rückgabewert der Funktion ist das Ergebnis y der Gleichung.

```
   Data : Liste mit metrischen Parametern
   Result : result
1  foreach je zwei Parameter der Liste do
2  |  if nur ein Parameter übrig then
3  |  |  result += β_i parameter;
4  |  else
5  |  |  result += β_i parameter1 * parameter2
6  |  end
7  end
8  return result
```

Pseudocode 11 : Lineare Funktion mit Wechselwirkungen

Quadratisches Modell ohne Wechselwirkungen (QwoD)

Das „Quadratische Modell ohne Wechselwirkungen“ simuliert eine Funktion der Form:

$$y = \beta_1 a^2 + \beta_2 b + \beta_3 c \tag{4.4}$$

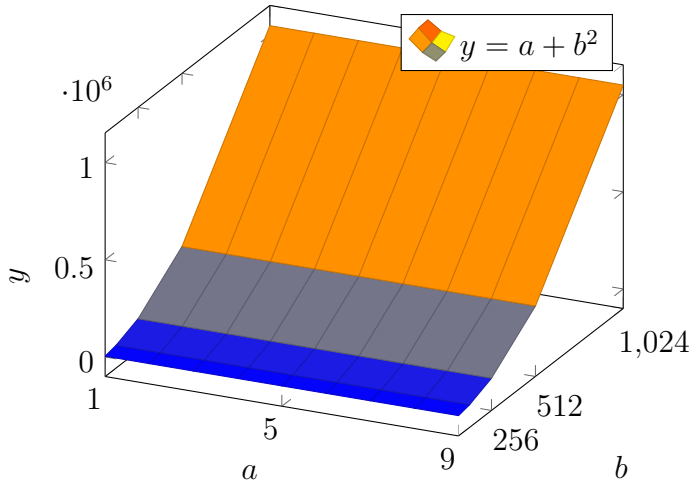


Abbildung 4.5: Dreidimensionaler Plot zur Veranschaulichung eines quadratischen Modells ohne Wechselwirkungen. Die Farben dienen lediglich zur Kenntlichmachung der Lage der Ebene

Dazu wird der erste Parameter der Liste quadriert und mit einem beliebigen Wert β_i multipliziert. Die daraus entstehenden Teilwerte werden anschließend aufsummiert (Pseudocode 12). Der Rückgabewert der Funktion ist das Ergebnis y der Gleichung.

```
   Data : Liste mit metrischen Parametern
   Result : result
1  foreach Parameter der Liste do
2  |  if erster Parameter then
3  |  |  result += β_i parameter^2;
4  |  else
5  |  |  result += β_i parameter;
6  |  end
7  end
8  return result
```

Pseudocode 12 : Quadratische Funktion ohne Wechselwirkungen

4.3.3 Bedienung des Simulators

Die Bedienung des NFPMsimulator erfolgt mittels Übergabe diverser Kommandozeilenargumenten, die jeweils eine Variante widerspiegeln. Der beispielhafte Aufruf des NFPMsimulators in Quelltext 4.1 zeigt die Wahl des zu simulierenden Modells und die Übergabe der zu verwendenden metrischen Konfigurationsoptionen.

Quelltext 4.1: Beispielhafter Aufruf des NFPMsimulators

```
1 # NFPMsimulator -m linearWithoutDependencies -p x1=2:x2=4:x3=8
```

-m erlaubt die Wahl aus den zu simulierenden Modellen:

- `linearWithoutDependencies`

- `linearWithDependencies`
- `quadraticWithoutDependencies`

-p erlaubt die Übergabe metrischer Parameter in der Form: $x_0 = y_0 : \ldots : x_i = y_i$

-f stößt die Erzeugung eines vollfaktoriellen Designs an, welches als XML-Datei gespeichert wird. Es orientiert sich dabei an dem bereits für SPL Conqueror erstellten Featuremodell in XML-Notation, welches mittels `-x` angegeben wird.

-x definiert den Pfad zum Featuremodell, welches zur Erzeugung des vollfaktoriellen Designs verwendet werden soll.

4.4 Gegenstand der Experimente

Ziel der Experimente ist es, die für die spätere Evaluierung benötigten Daten zu generieren. Die zu untersuchenden Eigenschaften belaufen sich in erster Linie auf die Performanz der jeweiligen experimentellen Design im Hinblick auf diverse Faktoren. Die Performanz definiert sich dabei in der Anzahl der benötigten Versuchsläufe, der Güte der auf Basis der enthaltenen Versuchspunkte gelernten Funktionen sowie dem Aufwand der Designerzeugung. Zudem soll die Skalierbarkeit der Implementierungen im Bezug auf die Parameteranzahl und deren Stufen untersucht werden.

4.5 Limitationen in der Durchführung der Experimente

Wenngleich versucht wurde, den Rechenaufwand gering zu halten und den Ressourcenbedarf zu minimieren, unterliegen die folgenden Experimente einigen Limitationen. Diese sind einerseits in der Leistungsfähigkeit der verfügbaren Hardware begründet, andererseits auf Beschränkungen in der Implementierungstechnik zurückzuführen. Ebenso stießen die gängigen Speicherstrukturen der verwendeten Programmiersprache beispielsweise beim Vorhalten von vollständigen Versuchsplänen bei großen Variantenräumen an ihre Grenzen.
Die Implementierung von SPL Conqueror wurde - soweit möglich - für diesen Zweck stellenweise leicht modifiziert. Umfassendere Anpassungen gehen zwar über den Umfang dieser Arbeit hinaus, bieten aber gleichzeitig Ansatzpunkte für zukünftige Arbeiten (Abschnitt 7).

Zur Implementierung (Abschnitt 4.1) und Anwendung der experimentellen Designs (Abschnitt 5.2.2 und Abschnitt 5.2.3) wurde auf einen Desktop-Rechner mit Intel Core2Duo auf 3.16Ghz und 4GB Arbeitsspeicher zurückgegriffen. Softwareseitig fand Windows 7 in Verbindung mit Visual Studio 2012 Verwendung, um die in `C#` geschriebene Anwendung komfortabel entwickeln und ausführen zu können. Obwohl die genannten Spezifikationen für gewöhnlich als ausreichend zu beschreiben sind, findet sich hier vorallem in Verbindung mit der Generierung eines D-optimalen Design unter Verwendung des k-Exchange Algorithmus ein Flaschenhals.

5. Experimente

In diesem Kapitel wird der Versuchsaufbau und die Durchführung der Experimente beschrieben. Die Ergebnisse dieses Kapitels dienen anschließend für die Evaluierung der verschiedenen Designs.

5.1 Versuchsaufbau

Die Experimente zur Datengewinnung werden in zwei Stufen durchgeführt. In der ersten Ebene werden die verschiedenen Seeds des Plackett-Burman Designs gegenübergestellt - interne Betrachtung. Die Ergebnisse dieser Untersuchungen fließen in die Versuche der nächsten Stufe mit ein.
Auf zweiter Ebene werden anschließend die experimentellen Versuchspläne untereinander verglichen - externe Betrachtung. Dazu werden drei Parametersets mit jedem der drei Simulationsmodelle kombiniert und weitere drei der implementierten experimentellen Designs auf diese neun Paare angewendet. Diese Kombination lässt den Versuchsblock aus insgesamt 27 Teilexperimenten bestehen.
Auf eine interne sowie externe Evaluierung der mittels des k-Exchange Algorithmus erzeugten D-optimalen Designs wird aus Gründen fehlender potenter Hardware an dieser Stelle verzichtet.

Betrachtet werden jeweils die absolute Anzahl von benötigten Versuchsläufen und der Anteil der Versuche am gesamten Versuchsraum. Zudem wird der relative Fehler bestimmt, mit dem die, auf Basis der gelernten Modelle getroffenen, Vorhersagen von den realen Ergebnissen abweichen.

5.1.1 Parametersets

Jedes der Parametersets stellt auf abstrakte Weise die verfügbaren Konfigurationsoptionen eines konfigurierbaren Systems dar. Ein Parameterset besteht aus einer bestimmten Anzahl metrischer Parameter mit unterschiedlichen Wertebereichen. In Tabelle 5.1 ist die genaue Zusammensetzung der Sets sowie die Anzahl der daraus resultierenden Varianten dargestellt. Die Sets mit 4 beziehungsweise 6 Elementen bestehen jeweils zur Hälfte aus Parametern mit 7 und 9 Stufen. Aufgrund der in

Abschnitt 4.5 beschriebenen hardware- und softwareseitiger Limitationen mussten die Stufen des Sets mit 8 Parametern entsprechend auf 5 und 7 Stufen herabgesetzt werden. Ohne diese Anpassungen hätte das Parameterset bereits knapp 16 Millionen Varianten beinhaltet.

Set	Parameter (gesamt)	Parameter (1)	Parameter (2)	Varianten
1	**4**	2 á 7 Stufen	2 á 9 Stufen	**3.969**
2	**6**	3 á 7 Stufen	3 á 9 Stufen	**250.047**
3	**8**	4 á 5 Stufen	4 á 7 Stufen	**1.500.625**

Tabelle 5.1: Zusammensetzung der Parametersets mit Anzahl der Parameter und deren Aufteilung in jeweils zwei Gruppen mit unterschiedlicher Stufenzahl sowie die Größe des zugehörigen Variantenraums

Alle Sets liegen als Featuremodell vor, welches sowohl von SPL Conqueror, als auch dem NFPMsimulator eingelesen und verarbeitet werden kann. Ein Ausschnitt einer solchen Datei ist in Quelltext 5.1 abgebildet. Ein `<feature>`-Block beschreibt die Eigenschaften eines Parameters. Darin zu erkennen sind die Definition der oberen und unteren Wertebereichsgrenze durch `<min>` und `<max>` sowie die Schrittgröße zwischen den Werten des Wertebereichs. Der festgelegte Prefix dient unter anderem zur späteren Zuordnung der Parameter und wird Teil der an den NFPMsimulator übergebenen Kommandozeilenargumente.

Quelltext 5.1: Ausschnitt eines Featuremodells in XML-Syntax

```
<plm name=”Evaluierung_1” ...>
        ...
        <variableFeatures>
                <feature id=”1”>
                        <name>a</name>
                        <prefix>x1=</prefix>
                        ...
                        <min>1</min>
                        <max>9</max>
                        <stepsize>1</stepsize>
                        ...
                </feature>
        ...
        </variableFeatures>
        ...
</plm>
```

5.2 Versuchsdurchführung

In diesem Abschnitt finden sich die Ergebnisse der Versuche. Zur Auswertung der Daten wurde auf die vorgenommene Implementierungen der experimentellen Design

und die Programmiersprache `R` unter Verwendung des „R Studio Server“ für Linux zurückgegriffen. Die dazu verwendeten Dateien und das R-Skript mit den verschiedenen Regressionsmodellen findet sich im Anhang.

5.2.1 Generierung der vollständigen Versuchspläne

Um abschließend die Güte der von „SPL Conqueror“ gelernten Funktionen beurteilen zu können, bedarf es der Generierung eines vollständigen Versuchsplans. Anschließend wird jedem Versuch das entsprechende Ergebnis des simulierten Modells zugewiesen. Das Vorgehen resultiert also in insgesamt neun Datensätzen - je einer pro Kombination aus Modell und Parameterset. Durch den Vergleich dieser Werte mit den Ergebnissen der experimentellen Designs kann die relative Abweichung vom Modell ermittelt werden.

Übernommen wurde dieser Teilschritt - hauptsächlich aus Effizienzgründen - durch den NFPMsimulator, da eine durchschnittliche Bearbeitungszeit von etwa 150 Millisekunden je Messung durch „SPL Conqueror“ schnell zu Laufzeiten im Bereich von Tagen oder Wochen führt. Der Generierungsvorgang selbst wurde ebenso einigen Optimierungen unterzogen, um den Arbeitsspeicherbedarf und den Rechenaufwand möglichst gering zu halten. So können, im Falle des Sets 2, in der selben Zeit knapp 11.000-mal mehr Konfigurationen erzeugt, gemessen und geschrieben werden. Der Vorgang benötigt im Falle des Parametersets mit 6 Parametern auf diese Weise nur ca. 3,5 Sekunden. Der Speicherbedarf pro Konfiguration beträgt anschließend etwa 180 Byte, was in diesem Fall eine Dateigröße von ∼44MB zu Folge hat.

5.2.2 Vergleich der Varianten eines experimentellen Designs

Zum Vergleich der Varianten eines experimentellen Designs wurden zunächst die von den experimentellen Designs generierten Versuchspläne für jedes der drei Parametersets exportiert. Da sich im Verlauf der Untersuchungen bezüglich der zur Funktionsfindung verwendeten Regression das Problem des „perfect fit“ einstellte, wurde bei der anschließenden Berechnung der Funktion ein randomisierter Fehler von 5% (Noise) auf die Funktionswerte angewandt. Ein „perfect fit“ ist dabei gleichzusetzen mit einem Restfehlerwert [fQCSD83, 6.21 - „Residual Error“] von Null und wird durch die Kenntnis der exakten simulierten Funktion verursacht. Das Hinzufügen des Fehlers verhindert das Auftreten dieser Problematik und erlaubt eine grundlegende Bewertung der Qualität der von den experimentellen Designs vorgeschlagenen Versuche. Die durch die gelernte Funktion $f_{noise}(x_1, \ldots, x_n)$ bestimmten Werte werden anschließend gegen die Werte des tatsächlichen Simulationsmodells $f_{real}(x_1, \ldots, x_n)$ in Abhängigkeit der in der Konfiguration enthaltenen Parameter $x_1, \ldots, x_n$ verglichen und die relative Abweichung Err festgehalten:

$$Err = \left| \frac{f_{noise}(x_1, \ldots, x_n) - f_{real}(x_1, \ldots, x_n)}{f_{real}(x_1, \ldots, x_n)} \right| \tag{5.1}$$

Tabelle 5.2, Tabelle 5.3 und Tabelle 5.4 zeigen den Vergleich der unterschiedlichen Seeds für Plackett-Burman Designs. Jeweils eine der drei Tabelle enthält die Betrachtung des linearen Modells mit und ohne Wechselwirkungen sowie die des quadratischen Modells ohne Wechselwirkungen. Die beiden Spalten N und L identifizieren den verwendeten Seed, die übrigen drei Spalten das jeweilige Parameterset. Die

N	L	Parameterset 1			Parameterset 2			Parameterset 3		
		#	%	Err	#	%	Err	#	%	Err
9	3	9	0,23	2,30	9	0,00	1,70	9	0,00	2,05
27	3	27	0,68	0,47	27	0,01	1,09	27	0,00	1,58
81	3	81	2,04	0,35	81	0,03	0,85	81	0,00	0,47
25	5	25	0,63	0,48	25	0,01	1,63	25	0,00	1,81
125	5	125	3,15	0,46	125	0,05	0,45	125	0,00	0,46
49	7	49	1,23	0,96	49	0,02	1,57	-	-	-

Tabelle 5.2: Ergebnisse des Vergleichs der verschiedenen Seeds für Plackett-Burman Designs unter Verwendung des linearen Modells ohne Wechselwirkungen

Spalte eines jeden Parametersets ist wiederum in drei Spalten unterteilt, wobei die Raute die Anzahl der durchgeführten Versuche, das Prozentzeichen den prozentualen Anteil derer am gesamten Variantenraum und „Err“ die durchschnittliche relative Abweichung des gelernten vom tatsächlichen Wert in Prozent repräsentiert.

Es folgt jeweils eine kurze Beschreibung der Tabellen und deren Ergebnissen, die detailierte Auswertung und Interpretation findet sich anschließend in Abschnitt 6. Die in den Tabellen bei Parameterset 3 ausgesparten Experimente auf Basis des Seeds mit $N = 49$ und $L = 7$, können aufgrund der zu niedrigen Anzahl an verfügbaren Parameterstufen nicht durchgeführt werden.

Tabelle 5.2 zeigt die Ergebnisse des internen Vergleichs der Plackett-Burman Designs unter Verwendung des linearen Modells ohne Wechselwirkungen. Unabhängig vom Parameterset tendiert der relative Fehler mit steigender Versuchszahl zu niedrigeren Werten. Grün markiert wurde jeweils der beste - da der niedrigste - relative Fehler, der zugehörige Seed ist also bei der Anwendung des Designs im jeweiligen Parameterset zu bevorzugen. Für den externen Vergleich im Kontext des LwoD werden folglich der Seed mit $N = 81$ und $L = 3$ für Parameterset und der Seed mit $N = 125$ und $L = 5$ für die Parametersets 2 und 3 verwendet.

Tabelle 5.3 zeigt die Ergebnisse des internen Vergleichs der Plackett-Burman Designs unter Verwendung des linearen Modells mit Wechselwirkungen. Eine eindeutige Korrelation zwischen einem niedrigen relativen Fehler und einer hohen Anzahl an Versuchen ist hier nicht gegeben, eine Tendenz ist jedoch zu erkennen. Die grün hinterlegten Felder identifizieren erneut den besten Wert bezüglich des relativen Fehlers, es wird also der Seed mit $N = 25$ und $L = 5$ beim internen Vergleich im Kontext des LwD für Parameterset 1 und der Seed mit $N = 125$ und $L = 5$ für die Parametersets 2 und 3 verwendet.

Tabelle 5.4 zeigt die Ergebnisse des internen Vergleichs der Plackett-Burman Designs unter Verwendung des quadratischen Modells mit Wechselwirkungen. Wieder ist die

N	L	Parameterset 1			Parameterset 2			Parameterset 3		
		#	%	Err	#	%	Err	#	%	Err
9	3	9	0,23	1,85	9	0,00	1,74	9	0,00	0,86
27	3	27	0,68	0,55	27	0,01	0,60	27	0,00	0,65
81	3	81	2,04	0,41	81	0,03	0,55	81	0,00	0,55
25	5	25	0,63	0,34	25	0,01	1,52	25	0,00	2,92
125	5	125	3,15	0,90	125	0,05	0,17	125	0,00	0,52
49	7	49	1,23	0,99	49	0,02	0,54	-	-	-

Tabelle 5.3: Ergebnisse des Vergleichs der verschiedenen Seeds für Plackett-Burman Designs unter Verwendung des linearen Modells mit Wechselwirkungen.

N	L	Parameterset 1			Parameterset 2			Parameterset 3		
		#	%	Err	#	%	Err	#	%	Err
9	3	9	0,23	1,17	9	0,00	2,42	9	0,00	2,00
27	3	27	0,68	0,85	27	0,01	0,91	27	0,00	1,22
81	3	81	2,04	0,27	81	0,03	0,82	81	0,00	0,50
25	5	25	0,63	1,70	25	0,01	1,14	25	0,00	1,72
125	5	125	3,15	0,10	125	0,05	0,56	125	0,00	0,57
49	7	49	1,23	1,44	49	0,02	0,42	-	-	-

Tabelle 5.4: Ergebnisse des Vergleichs der verschiedenen Seeds für Plackett-Burman Designs unter Verwendung des quadratischen Modells ohne Wechselwirkungen

Tendenz des mit steigender Versuchszahl sinkenden relativen Fehlers zu beobachten. Der Rückgriff auf eine Seed mit mehr unterstützten Stufen scheint sich ebenso positiv auszuwirken. Wieder markieren die grünen Felder den niedrigsten relativen Fehler innerhalb eines Parametersets. Für den späteren internen Vergleich im Kontext des QwoD wird also der Seed mit $N = 125$ und $L = 5$ für Parameterset 1, der Seed mit $N = 49$ und $L = 7$ für Parameterset 2 und der Seed mit $N = 81$ und $L = 3$ für Parameterset 3 verwendet.

5.2.3 Vergleich der experimentellen Designs

Tabelle 5.5, Tabelle 5.6 und Tabelle 5.7 zeigen den externen Vergleich der experimentellen Designs. Das grundlegende Vorgehen zur Datengewinnung entspricht bei dieser Versuchsreihe dem in Abschnitt 5.2.2 vorgestellten. Für jedes der Modelle und Parametersets wurde im Falle der Plackett-Burman Designs jeweils der Beste Seed im Bezug auf den durchschnittlichen relativen Fehler aus Abschnitt 5.2.2 gewählt. Die erste Spalte gibt das verwendete Design an, die übrigen drei Spalten werden von je einem Parameterset belegt. Wie bereits bei der internen Betrachtung der Plackett-Burman Designs sind die Spalten der Parametersets selbst nochmal in drei Spalten unterteilt, welche die Anzahl der Versuchsläufe, den prozentualen Anteil derer am gesamten Variantenraum und den durschnittlichen relativen Fehler angeben.
Da ein unvollständiger Vergleich im Falle der mittels des k-Exchange Algorithmus erstellten D-optimalen Designs auch hier als nicht sinnvoll erscheint, wird auf deren Anwendung verzichtet.

Es folgt jeweils eine kurze Beschreibung der Tabellen und deren Ergebnissen, die detailierte Auswertung und Interpretation findet sich anschließend in Abschnitt 6.

Tabelle 5.5 zeigt die Ergebnisse des externen Vergleichs der experimentellen Designs unter Verwendung des linearen Modells ohne Wechselwirkungen. Abgesehen von den Plackett-Burman Designs scheint keiner der Versuchspläne eine über die Parametersets hinweg konstante Leistung zu erbringen. Die Plackett-Burman Designs schneiden zudem jeweils mit dem niedrigsten relativen Fehler am besten ab.

Tabelle 5.6 zeigt die Ergebnisse des externen Vergleichs der experimentellen Designs unter Verwendung des linearen Modells mit Wechselwirkungen. Auch hier lässt sich keine Leistungstendez der verschiedenen Designs zwischen den Parametersets erkennen, die Plackett-Burman Design bilden hier jedoch die Ausnahme. Die niedrigsten relativen Fehler sind im Falle des Parametersets 1 und 2 auf die Plackett-Burman Designs zurückzuführen, den Vergleich innerhalb des Parametersets 3 gewinnt das zweistufige faktorielle Design.

Tabelle 5.7 zeigt die Ergebnisse des externen Vergleichs der experimentellen Designs unter Verwendung des quadratischen Modells ohne Wechselwirkungen. Auch hier liefern die Plackett-Burman Designs für alle Parametersets die besten Ergebnisse im Bezug auf die Abweichung der gelernten von der tatsächlichen Funktion.

	Parameterset 1			Parameterset 2			Parameterset 3		
	#	%	Err	#	%	Err	#	%	Err
2^{k-1}	8	0,20	1,11	32	0,01	1,10	128	0,01	0,58
Box-Behnken	25	0,63	1,13	49	0,02	0,53	97	0,01	1,32
CCI	25	0,63	0,81	77	0,03	1,14	273	0,02	2,11
Plackett-Burman	81	2,04	0,35	125	0,05	0,45	125	0,01	0,47

Tabelle 5.5: Ergebnisse des Vergleichs der verschiedenen experimentellen Designs unter Verwendung des linearen Modells ohne Wechselwirkungen

	Parameterset 1			Parameterset 2			Parameterset 3		
	#	%	Err	#	%	Err	#	%	Err
2^{k-1}	8	0,20	3,30	32	0,01	0,65	128	0,01	0,38
Box-Behnken	25	0,63	0,80	49	0,02	1,14	97	0,01	0,60
CCI	25	0,63	1,44	77	0,03	0,80	273	0,02	5,05
Plackett-Burman	25	0,63	0,34	125	0,05	0,17	125	0,01	0,52

Tabelle 5.6: Ergebnisse des Vergleichs der verschiedenen experimentellen Designs unter Verwendung des linearen Modells mit Wechselwirkungen

	Parameterset 1			Parameterset 2			Parameterset 3		
	#	%	Err	#	%	Err	#	%	Err
2^{k-1}	8	0,20	0,89	32	0,01	0,91	128	0,01	0,54
Box-Behnken	25	0,63	1,26	49	0,02	0,85	97	0,01	1,14
CCI	25	0,63	1,19	77	0,03	0,98	273	0,02	3,78
Plackett-Burman	125	3,15	0,10	49	0,02	0,42	81	0,01	0,50

Tabelle 5.7: Ergebnisse des Vergleichs der verschiedenen experimentellen Designs unter Verwendung des quadratischen Modells ohne Wechselwirkungen

6. Evaluierung

In diesem Kapitel wird die Auswertung der Experimente präsentiert und eventuelle Einschränkungen der Validität aufgedeckt.

6.1 Auswertung der Ergebnistabellen

Es folgt die Auswertung der Ergebnistabellen des internen sowie externen Vergleichs der experimentellen Designs. Bereits an dieser Stelle sei auch auf Abschnitt 6.2 verwiesen, in welchem die Aussagekraft der präsentierten Ergebnisse bewertet wird.

Über die interne Betrachtung der Plackett-Burman Designs hinweg, erwiesen sich grundlegend die Designmatrizen mit einer höheren Anzahl an Versuchsläufen als leistungsfähiger im Hinblick auf den relativen Fehler gegenüber des tatsächlichen Modells. Betrachtet man die Untersuchung jeweils eines Parametersets in Verbindung mit allen verfügbaren Seeds als Teilexperiment, so stellen sich in 7 von 9 Teilexperimenten die Designmatrizen mit 81 bzw. 125 Versuchsläufen als überlegen heraus. Beschränkt man diese Betrachtungsweise nur auf den Bereich der Teilexperimente, der auf 3 Stufen basiert, lässt sich diese Entwicklung sogar in 9 von 9 Fällen bestätigen. Vergleicht man die Ergebnisse je eines Seeds über die Parametersets und Modelle hinweg, so zeichnet sich in nur 8 von 18 Fällen die Tendenz eines steigenden relativen Fehlers bei sinkender prozentualer Abdeckung des Variantenraums ab. Da dies wider Erwarten in der Mehrheit der Fälle nicht festgestellt werden konnte, kann hier keine eindeutige Aussage bezüglich einer möglichen Korrelation getroffen werden.

Zur externen Bewertung der verschiedenen experimentellen Designs wurde im Fall des Plackett-Burman Designs jeweils die Version mit dem besten Ergebnis im internen Vergleich verwendet. Vorweg lassen sich bereits die guten Ergebnisse der Plackett-Burman Designs erwähnen, die in 8 von 9 Teilexperimenten das beste Ergebnis im Vergleich zu den anderen untersuchten Designs erreichen konnten. Als besonders leistungsschwach stellte sich das „Central Composite Inscribed“ Design heraus, dessen Versuchspunkte im Schnitt im höchsten relativen Fehler resultierten. Als Grund hierfür ist wohl die starke Konzentrierung der Versuche im mittleren

Bereich des Variantenraums anzuführen. Das Design mit den stabilsten Ergebnissen über die Modelle hinweg ist das Box-Behnken Design, dessen durchschnittlicher relativer Fehler bei unter 1% liegt.

Obwohl manche der Beobachtungen dazu verleiten, rudimentäre Rückschlüsse auf die Qualität der Wahl der Versuchspunkte zu ziehen, sind alle Aussagen diesbezüglich mit Vorsicht zu genießen. Zieht man nämlich in Betracht, dass sämtliche der Regression zugeführten Werte vorab mit dem fünfprozentigen Fehler versehen wurde, muss allen Designs eine hohe Güte ausgesprochen werden. Da der durchschnittliche relative Fehler der gelernten Funktionen - über alle Modelle und Parametersets hinweg - weniger als 1,1% beträgt, wird im folgenden versucht die eventuellen Einschränkungen bezüglich der Validität der Experimente darzulegen.

6.2 Validität der Experimente

Die durchgeführten Experimente und deren vorgestellten Ergebnisse unterliegen einigen Einschränkungen bezüglich der Validität, was deren Aussagekraft erheblich reduziert.
Allem voran gehen die durchweg niedrigen durchnittlichen Fehler der gelernten Funktionen, die in keiner Weise abhägig von der zur Wahl der Versuchspunkte verwendeten Methode scheint. Diese Annahme ist vorallem in der Tatsache begründet, dass trotz des Versehens der Werte mit einem zufälligen fünfprozentigen Fehler, die von den Regressionen ermittelten Koeffizienten - und damit die gelernten Funktionen - zuverlässige Näherungen lieferten.

Probleme allgemeinerer Natur sind in den diskreten Stufen der verwendeten metrischen Konfigurationsoptionen begründet. Die von den experimentellen Designs vorgeschlagenen Punkte liegen nicht zwingend auf erlaubten Stufen der Eingabeparameter. Es muss auf die nächste zulässige Parameterstufe zurückgegriffen werden, was sich insbesondere im Fall der Mittelpunktsversuche als problematisch erweisen kann. Nimmt man beispielweise für einen Parameter x die vier möglichen Stufen 1, 2, 3 und 4 an, so befindet sich dessen Mittelpunkt bei $\frac{1+2+3+4}{4} = 2,5$. Die Entfernung zur nächstgelegenen zulässigen Stufe beträgt jeweils 0,5 - eine Entscheidung für einen der beiden Nachbarpunkte auf Grundlage des mathematischen Rundens des Wertes ist nicht zielführend. Man sieht sich folglich mit einem nicht entscheidbaren Problem konfrontiert.
Ähnlich gelagert ist die Problematik des Zusammenfallens von Versuchspunkten durch Runden beim Central Composite Design in Verbindung mit kleinen Variantenräumen. Selbiges Problem kann ebenfalls bei Verwendung eines Plackett-Burman Designs eintreten. Es ist also in diesem Fällen von einer Verwendung dieser Designs abzusehen.

7. Zusammenfassung und zukünftige Arbeiten

Ziel der Arbeit war es, zur Bestimmung des nichtfunktionalen Einflusses metrischer Konfigurationsoptionen geeignete experimentelle Designs zu erfassen, zu implementieren und zu evaluieren. Die Implementierungen wurden als Erweiterungen zwar für „SPL Conqueror“ vorgenommen, sind aber grundsätzlich als universell zu betrachten.
Um eine einfache Möglichkeit zur Evaluation zu schaffen, wurde zudem der „NFPM-simulator“ entwickelt. Das Programm wurde modular gestaltet und lässt jederzeit eine Erweiterung um beliebige Modelle zu, um komfortabel bestehende oder zusätzliche experimentelle Designs hinsichtlich dieser Modelle bewerten zu können. Die angebotenen Konfigurationsoptionen (Abschnitt 4.3.3) machen den „NFPMsimulator“ selbst zu einem konfigurierbaren System.

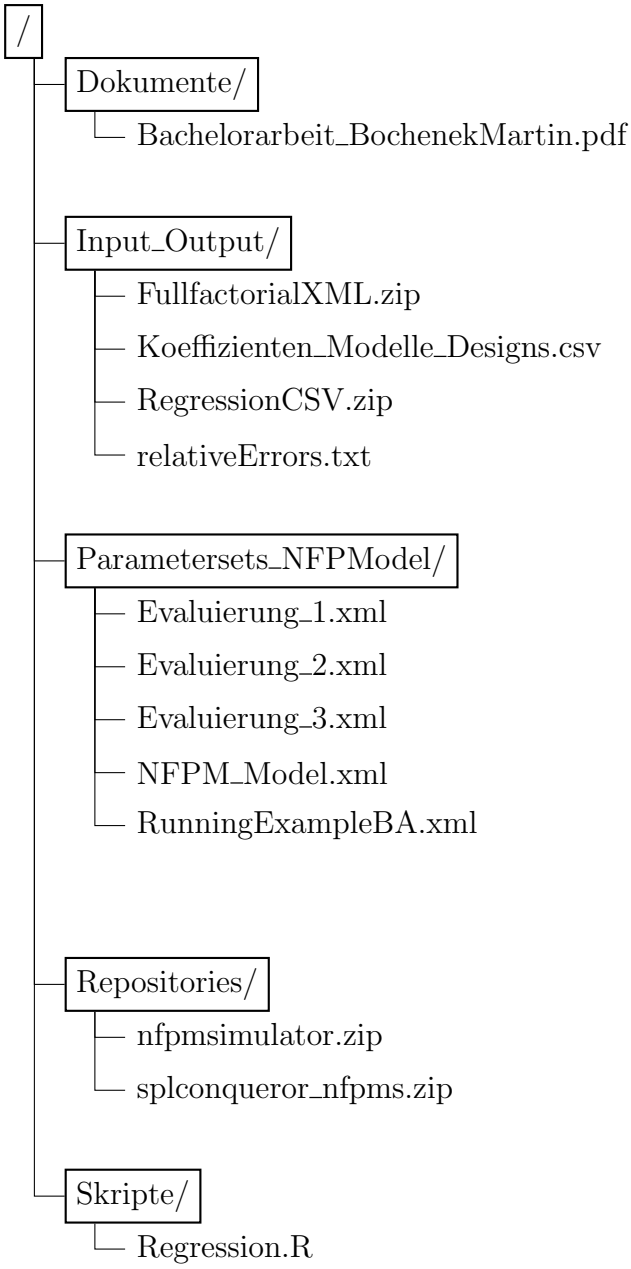
Die zu Beginn getroffene Entscheidung zur Simulation perfekter Modelle, erwies sich schlussendlich als Sackgasse und als ungeeignet für die angestrebte Evaluation. Die den Regressionsmodellen nachempfundenen simulierten Funktionen mit jeweils maximal k Koeffizienten für k Parameter, ließen durchgehend eine perfekte Funktion lernen - unabhängig von der verwendeten Methode zur Wahl der Versuchspunkte. Dieser Umstand liegt zusätzlich auch darin begründet, dass jedes Design bestrebt ist durch die Anzahl seiner enthaltenen Versuchspunkte ein quadratisches oder überbestimmtes Gleichungssystem zu generieren. Es ist folglich evident, dass die einem, mit dem Simulationsmodell übereinstimmenden, Regressionsmodell übergebenen Versuchspunkte und Werte zu einer perfekten Bestimmung der simulierten Koeffizienten führt.
Diese Erkenntnisse scheinen die Anwendung der vorgestellten experimentellen Designs auf den ersten Blick ad absurdum zu führen und die angestrebte Evaluierung als Misserfolg erscheinen. Nichtsdestotrotz konnten aber essentielle Erkenntnisse über die grundlegenden Eigenschaften der experimentellen Designs gewonnen werden. Beispielsweise Erkenntnisse über den mit den Versuchsplänen verbundenen Versuchsaufwand oder die im Falle des k-Exchange Algorithmus mangelhafte Skalierbarkeit.

Es bleibt jedoch wiederholt der modulare Aufbau des NFPMsimulators zu nennen, der die Integration eines verbesserten Simulationsmodells ohne großen Aufwand zulässt und somit eine erneute Evaluierung der ausgewählten experimentellen Designs ermöglicht.

Wenngleich also die vorliegenden Ergebnisse selbst leider keine qualitative Antwort für das eingangs vorgestellte Problem zulassen, zeigen die gewonnenen Erkenntnisse zahlreiche Ansatzpunkte für zukünftige Arbeiten auf.
Der wohl naheliegendste Ansatz ist die Anwendung der implementierten experimentellen Designs im Bereich realer, konfigurierbarer Systeme oder ganzer Softwareproduktlinien. Diese Untersuchungen lassen anschließend detailierte Rückschlüsse über das Systemverhalten zu, wodurch maßgeblich zur Lösung des zu Beginn angesprochene Problems der Findung einer optimalen Variante beigetragen werden kann.
Aufbauend auf diesem Punkt ist zudem eine tiefergehende Evaluierung von Systemen denkbar, welche durch die Einbringung von vorhandenem Domänenwissen angestoßen werden kann. Ein solches Wissen über das systeminterne Verhalten und/oder die Zusammenhänge und Wechselwirkungen bestimmter Konfigurationsoptionen, erlaubt die gesonderte Betrachtung von Untermengen der verfügbaren Konfigurationsoptionen. Dies wiederum gestattet die Reduzierung des Variantenraums und die gezielte Anwendung eines experimentellen Designs, auch im Hinblick auf das bekannte bzw. vermutete zugrundeliegende Modell.

Weiterhin finden sich Ansätze allgemeinerer Natur in der Erweiterung und Optimierung der Umsetzungen sowie der Implementierung zusätzlicher experimenteller Designs.
Denkbar sind hier beispielsweise der Ausbau des 2^{k-1} teilfaktoriellen Designs zu einem allgemeinen 2^{k-p} Design und dessen Erweiterung zum 3^{k-p} Design, mit je drei Stufen pro untersuchtem Parameter.
Ebenfalls überlegenswert ist die Verbesserung der vorliegenden Implementierung des k-Exchange Algorithmus im Bezug auf dessen Skalierbarkeit. Die aktuelle Vorgehensweise ist bereits bei kleineren Parametersets mit erheblichen Rechenaufwand und Speicherbedarf verbunden, was eine umfangreichere Anwendung dessen schnell als technisch nicht machbar ausweist. Ferner ist ebenso der Rückgriff auf andere Algorithmen zur Generierung D-optimaler Designs in Erwägung zu ziehen.
Zu guter Letzt bleibt die Umsetzung weiterer Versuchspläne wie beispielsweise Designs nach der Taguchi-Methode [NIS, Kapitel 5.5.6] oder aber sogenannte „Nested“- oder „Split-Pot“-Designs [Mon05, Kapitel 14] als weitere Anknüpfungspunkt zu nennen.

A. Anhang


/
Dokumente/
Bachelorarbeit_BochenekMartin.pdf
Input_Output/
FullfactorialXML.zip
Koeffizienten_Modelle_Designs.csv
RegressionCSV.zip
relativeErrors.txt
Parametersets_NFPModel/
Evaluierung_1.xml
Evaluierung_2.xml
Evaluierung_3.xml
NFPM_Model.xml
RunningExampleBA.xml
Repositories/
nfpmsimulator.zip
splconqueror_nfpms.zip
Skripte/
Regression.R

Literaturverzeichnis

[BB60] G. E. P. Box and D. W. Behnken. Some new three level designs for the study of quantitative variables. *Technometrics*, 2(4):pp. 455–475, 1960. (zitiert auf Seite 20 und 21)

[BKPS04] G. Boeckle, P. Knauber, K. Pohl, and K. Schmid. *Software-Produktlinien - Methoden, Einführung und Praxis*. Dpunkt-Verlag, 1. aufl. edition, 2004. (zitiert auf Seite 1 und 7)

[dABK+95] P. F. de Aguiar, B. Bourguignon, M. S. Khots, D. L. Massart, and R. Phan-Than-Luu. D-optimal designs. *Chemometrics and Intelligent Laboratory Systems*, 30(2):pp. 199–210, 1995. (zitiert auf Seite 26, 27, 28, 29 und 30)

[dCMU] Software Engineering Institute (SEI) der Carnegie Mellon University. Software product lines | overview (05.04.2014). http://www.sei.cmu.edu/productlines/. (zitiert auf Seite 7)

[Fis35] R. A. Fisher. *The Design of Experiments*. Oliver and Boyd, 1935. (zitiert auf Seite 9)

[fQCSD83] American Society for Quality Control. Statistics Division. *Glossary and Tables for Statistical Quality Control*. American Society for Quality Control, 1983. (zitiert auf Seite 13, 17, 18, 19, 20 und 49)

[IEE90] IEEE. Std 610.12-1990, ieee standard glossary of software engineering terminology, 1990. (zitiert auf Seite 7)

[JN83] M. E. Johnson and C. J. Nachtsheim. Some guidelines for constructing exact d-optimal designs on convex design spaces. *Technometrics*, 25(3):pp. 271–277, 1983. (zitiert auf Seite 29)

[Kie59] J. Kiefer. Optimum experimental designs. *Journal of the Royal Statistical Society. Series B (Methodological)*, 21(2):pp. 272–319, 1959. (zitiert auf Seite 27)

[MN95] R. K. Meyer and C. J. Nachtsheim. The coordinate-exchange algorithm for constructing exact optimal experimental designs. *Technometrics*, 37(1):pp. 60–69, 1995. (zitiert auf Seite 30)

[Mon05] D. C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons Australia, Limited, 6. aufl. edition, 2005. (zitiert auf Seite 13, 16, 17, 18, 19, 20, 22, 23, 26, 27, 28 und 60)

[NIS] NIST/SEMATECH. e-handbook of statistical methods (05.04.2014). http://www.itl.nist.gov/div898/handbook/index.htm. (zitiert auf Seite 17, 20, 22, 23, 28, 34, 36 und 60)

[PB46] R. L. Plackett and J. P. Burman. The design of optimum multifactorial experiments. *Biometrika*, 33(4):pp. 305–325, 1946. (zitiert auf Seite 23, 24 und 25)

[Pet91] H. Petersen. *Grundlagen der statistischen Versuchsplanung - Band 2*. Ecomed, Landsberg am Lech, 1991. (zitiert auf Seite 9, 10, 14, 17, 20, 22, 25 und 26)

[RK11] A. Rabkin and R. Katz. Static extraction of program configuration options. In *Proceedings of the 33rd International Conference on Software Engineering (ICSE)*, page 2011, 2011. (zitiert auf Seite 7)

[RR06] J. C. Robertson and S. Robertson. *Mastering the Requirements Process*. Pearson Education, 2. aufl. edition, 2006. (zitiert auf Seite 1 und 8)

[Sie] N. Siegmund. Spl conqueror - software product line configurator for non-functional properties (05.04.2014). http://wwwiti.cs.uni-magdeburg.de/~nsiegmun/SPLConqueror/. (zitiert auf Seite 2 und 9)

[SRK+12] N. Siegmund, M. Rosenmüller, M. Kuhlemann, C. Kästner, S. Apel, and G. Saake. Spl conqueror: Toward optimization of non-functional properties in software product lines. *Software Quality Journal*, 20(3-4):pp. 487–517, 2012. (zitiert auf Seite 9)

[SRK+13] N. Siegmund, M. Rosenmüller, C. Kästner, P. G. Giarrusso, S. Apel, and S. S. Kolesnikov. Scalable prediction of non-functional properties in software product lines: Footprint and memory consumption. *Information and Software Technology*, 55(3):pp. 491–507, 2013. Special Issue on Software Reuse and Product Lines. (zitiert auf Seite 7)

[Ste46] S. S. Stevens. On the theory of scales of measurement. 1946. (zitiert auf Seite 2, 5 und 6)

[SvBH10] K. Siebertz, D. van Bebber, and T. Hochkirchen. *Statistische Versuchsplanung: Design of Experiments (DoE)*. VDI-Buch. Springer, 2010. (zitiert auf Seite 10, 12, 15, 22 und 28)

[Tri08] F. Triefenbach. Design of experiments: The d-optimal approach and its implementation as a computer algorithm (bachelor's thesis in information and communication technology). 2008. (zitiert auf Seite 30 und 39)

Eidesstattliche Erklärung:

Hiermit versichere ich an Eides statt, dass ich diese Bachelorarbeit selbständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe und dass alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind, sowie dass ich die Bachelorarbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe.

Martin Bochenek

Passau, den 07. April 2014