# Weiterentwicklung und Validierung des GRS-Kernsimulators für DWR

GRS - 499

Gesellschaft für Anlagen-
und Reaktorsicherheit
(GRS) gGmbH

# Weiterentwicklung und Validierung des GRS-Kernsimulators für DWR

Matías Zilly
Alexander Aures
Jérémy Bousquet
Matthias Küntzel

Juni 2018

**Anmerkung:**

Das diesem Bericht zugrunde liegende FE-Vorhaben wurde mit Mitteln des Bundesministeriums für Umwelt, Naturschutz und nukleare Sicherheit (BMU) unter dem Kennzeichen 4715R01344 durchgeführt.

Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Auftragnehmer.

Der Bericht gibt die Auffassung und Meinung des Auftragnehmers wieder und muss nicht mit der Meinung des Auftraggebers übereinstimmen.

**GRS - 499**
**ISBN 978-3-946607-84-7**

**Deskriptoren**
Kernsimulator, KMACS, Zyklusberechnung

## Kurzfassung

Im Vorhaben 4715R01344 wurde der GRS-Kernsimulator KMACS für DWR weiterentwickelt und validiert. Das Ergebnis ist die Freigabeversion der Software KMACS 1.0. Dieser Bericht stellt die Weiterentwicklung des Programms im Vorhabensverlauf sowie die durchgeführten Arbeiten zur Validierung vor.

## Abstract

In the frame of project 4715R01344, the GRS core simulator KMACS for PWR was further developed and validated. The result is the release version of the KMACS 1.0 software. This report presents the further development of the program in the course of the project as well as the work carried out on the validation.

## Inhaltsverzeichnis

# 1 Einleitung

Der vorliegende Abschlussbericht zum Vorhaben 4715R01344 „Weiterentwicklung und Validierung des GRS-Kernsimulators für DWR“ beschreibt den Fortgang der Entwicklung und Validierung des GRS-Kernsimulators KMACS, der zum Berichtszeitpunkt als Freigabeversion KMACS 1.0 vorliegt.

Neben diesem Abschlussbericht sind mit dem „KMACS User Manual“ /BOU 18/ und dem „KMACS Validation Report“ /ZIL 18/ weitere Dokumente entstanden, die wesentliche Ergebnisse des Vorhabens enthalten, nämlich die Dokumentation der in KMACS implementierten Modelle und Methoden, eine Einführung in die Nutzung des Programms und der Analysewerkzeuge sowie die Beschreibung des Eingabedatensatzes einerseits und Ergebnisse der zur Validierung des Kernsimulators durchgeführten Rechnungen andererseits. Um unnötige Wiederholungen zu vermeiden, beschränkt sich daher dieser Bericht im Wesentlichen auf die Beschreibung der im Vorhabensverlauf durchgeführten Arbeiten sowie der daraus gewonnenen Erkenntnisse.

## 1.1 Ausgangslage

Das hier beschriebene Vorhaben mit dem Förderkennzeichen 4715R01344 ist das Nachfolgevorhaben des Vorhabens „Erstellung und Qualifizierung eines nuklearen Kernsimulators für die Sicherheitsbewertung aktueller Reaktorkernbeladungen“ (3612R01339), welches 2015 beendet wurde und in /ZIL 15/ beschrieben ist. Der darin entstandene Kernsimulator ist der Prototyp des jetzt vorliegenden Programmsystems.

Mit der Entwicklung eines Kernsimulators verfolgt die GRS, wie bereits in /ZIL 15/ genannt, das Ziel ein Werkzeug bereitzustellen um die Bundesaufsicht der Kernkraftwerke bezüglich aktueller Kernbeladungen unabhängig und mit einer diversitären Software beraten zu können.

Diese Kernbeladungen zeichnen sich in Deutschland angesichts des Endes der kommerziellen Stromerzeugung aus Kernenergie durch die Besonderheit aus, dass das festgelegte Laufzeitende bzw. die Reststrommenge aus ökonomischen Gründen Beladungen mit wenigen oder keinen frischen Brennelementen zur Folge hat und haben wird.

Diese sogenannten „Clean-Up“-Zyklen weisen eine höhere Heterogenität und einen höheren mittleren Abbrand auf. Deswegen müssen sie bereits von einem frühen Zeitpunkt im Zyklus bzw. über den gesamten Zyklus in Teillast („Stretch-out“- bzw. „Stretch-in“-Betrieb) gefahren werden.

Hinzu kommt ein häufigerer Lastwechselbetrieb, der sich wegen des zunehmenden Anteils von Strom aus regenerativen Quellen im Stromnetz ergibt und ebenfalls zu vermehrten Temperaturänderungen und Steuerstabbewegungen im Reaktorkern führt.

Damit haben sich die Bedingungen, die ein Kernsimulatorsystem nachvollziehen können muss, in den vergangenen Jahren verkompliziert. Gleichzeitig haben sich die Möglichkeiten zur Berechnung von Vorgängen im Reaktorkern fortentwickelt. So kommen, wenn auch noch nicht zur Zyklusauslegung, sogenannte direkte Ganzkern-Simulatoren zum Einsatz, mit denen die neutronenphysikalischen, thermohydraulischen und Brennstoffmechanischen Prozesse bis ins Detail der Brennstofftabletten genau berechnet werden können, s. z. B. /GOD 17/.

Neben der Begutachtung von Kernbeladungen soll KMACS auch der Bereitstellung von Reaktorkernmodellen für nachgelagerte Analysen dienen. Konkret geht es hier um die Anwendung des KMACS-Kernmodells im GRS-Analysesimulator, die Berechnung von Kerntransienten mit dem gekoppelten Codesystem ATHLET-QUABOX/CUBBOX, die Ermittlung des Nuklidinventars für Quelltermberechnungen sowie die Betrachtung lokaler Phänomene im Reaktorkern, wie z. B. Hüllrohroxidation oder Brennelement-Verbiegung. Letztere wird derzeit unter Verwendung von KMACS im Vorhaben „Leistungsverteilung bei Brennelement-Verbiegungen“ 4716R01355 untersucht.

Wegen dieser engen Verzahnung mit weiteren Aufgaben der GRS hat sich das modulare Konzept, d.h. die Verwendung in der GRS entwickelten und verwendeten Codes zur Wirkungsquerschnittserzeugung, 3D-Neutronenflussberechnung und zur Ermittlung des Abbrands als Module in KMACS bewährt.

## 1.2 Darstellung des Stands der Kernsimulator-Software vor Projektbeginn

Die wesentlichen Konzepte, die in KMACS 1.0 realisiert sind (s. /BOU 18/), waren schon im Prototyp angelegt: Die meisten physikalischen Gleichungen werden in Backend-Codes gelöst, die KMACS via Python-Schnittstellen ansteuert.

In der Vorausrechnung werden zunächst für jedes im zu untersuchenden Betriebszyklus eingesetzte Brennelement (BE) Wirkungsquerschnitte (WQ) erzeugt, die die neutronenphysikalischen Eigenschaften des BE bei verschiedenen a-priori angenommenen Betriebszuständen (gekennzeichnet durch Abbrand, Brennstofftemperatur, Kühlmitteldichte, Borkonzentration, kontrollierter/unkontrollierter Zustand) beschreiben. Die Vorausrechnung geschieht durch Lösung der Neutronentransportgleichung für einzelne BE in reflektierenden Randbedingungen mit einem Gittercode („Lattice Code“, auch Spektralcode genannt).

In der Zyklusrechnung werden sodann diese WQ entsprechend der Kernbeladung den Kontrollvolumina („Nodes“) des Flusslösers zugeordnet. Dieser ermittelt daraus die einem Zykluszeitpunkt entsprechende Neutronenfluss- und Leistungsverteilung. Diese werden anschließend im Abbrandcode verwendet, um die zeitliche Entwicklung von Nuklidinventar und Abbrand zu errechnen.

Im Prototypstadium beschränkten jedoch folgende Punkte die Anwendbarkeit des Kernsimulators:

1. Rechenzeitaufwand der Wirkungsquerschnittserzeugung. Mit dem im Prototyp verwendeten Gittercode NEWT aus dem SCALE 6.1-Paket waren, je nach verfügbaren Kapazitäten auf dem GRS-Rechencluster, mehrere Wochen bis Monate für die Erzeugung der WQ für einen Betriebszyklus erforderlich. Des Weiteren nahmen die Rechnungen auf den Rechenknoten einen immensen Plattenspeicherplatz in Anspruch, was immer wieder zu Programmabstürzen führte, wenn mehrere Rechnungen gleichzeitig auf einem Rechenknoten liefen.
2. Speicherplatzbedarf und Zugriffszeiten der Datenbank. Der Zugriff auf die Datenbank beim Lesen und Schreiben nahm, insbesondere bei den Zyklusrech-

nungen, mehrere Minuten in Anspruch und somit etwa so viel Zeit wie die Rechnungen von Flusslöser und Abbrandcode selbst.

3. Hauptspeicherplatzanforderung des Moduls zur Ansteuerung des Abbrandcodes. Die in der Vorausrechnung erzeugten nuklidspezifischen Wirkungsquerschnitte müssen für den Abbrandcode in jedem Kontrollvolumen zum jeweiligen thermohydraulischen Zustand hin interpoliert werden. Die frühere Implementierung nahm dafür große Hauptspeicherkapazitäten in Anspruch, die bei feiner räumlicher Diskretisierung des Kerns oder einer großen Anzahl betrachteter Nuklide zum Programmabsturz führten.
4. Geringe Benutzerfreundlichkeit des Eingabedatensatzes. Das im Prototyp verwendete `json`-Eingabeformat erwartet in geschweifte Klammern `{}` gruppierte Datenblöcke, die voneinander durch Kommata getrennt sind. Da eine Syntax-Überprüfung des Eingabedatensatzes nicht implementiert war, war die Erstellung eines Eingabedatensatzes fehleranfällig und zeitaufwendig.
5. Unzureichende Validierungsbasis mit deutlichen Abweichungen von der Referenz. Im Vorgängervorhaben wurden nur zwei Zyklen umfangreich untersucht. Dabei wurden insbesondere bei dem untersuchten Zyklus mit MOX-Einsatz große Abweichungen von bis zu 9 % in der Brennelementleistung beobachtet. Auch zyklusübergreifende Rechnungen waren im Vorgängervorhaben noch nicht untersucht worden.
6. Fehlende Funktionalität. Einige während eines normalen Betriebszyklus übliche Vorgänge konnte der Prototyp noch nicht abbilden. Dazu gehören das Verfahren von Steuerstabbänken, das Absenken der mittleren Kühlmitteltemperatur und die damit verbundene Zunahme des Massenstroms durch den Kern im Streckbetrieb.

## 1.3 Zielsetzung

Die vorgenannten Mängel des Kernsimulator-Prototyps haben zur Formulierung der Ziele dieses Vorhabens geführt, nämlich der Weiterentwicklung und der Validierung des Kernsimulators für DWR. Dazu werden in der Vorhabensbeschreibung die folgenden Einzelziele genannt:

1. Weitere Validierung an DWR-Anlagenzyklen. Dabei soll ein möglichst repräsentatives Spektrum typischer Kernbeladungen von mindestens zwei Anlagen abgedeckt werden, um zu vermeiden, dass Anlagen-Spezifika Einfluss auf die Validierung haben.
2. Einbau eines alternativen Gittercodes. Dabei ist das Ziel, die notwendige Rechenzeit für die WQ-Erzeugung deutlich zu reduzieren.
3. Bereitstellung einer Freigabeversion. Dabei ist das Ziel, ein für den externen Anwender handhabbares Werkzeug zu entwickeln und diese Entwicklung entsprechend den Anforderungen der GRS-QM-Richtlinie Codeentwicklung /SMG 17/ zu dokumentieren.

Der vorliegende Bericht dokumentiert, wie diese Ziele umgesetzt wurden.

# 2 Kurzdarstellung der Kernsimulator-Implementierung

Um die nachfolgend in Kapitel 0 (siehe Seite 11 ff.) ausgeführten Arbeiten am Kernsimulator KMACS gut erläutern zu können, seien hier einige im Kontext von KMACS spezifisch verwendete Begriffe erläutert und die Grundzüge der Implementierung dargestellt. Eine ausführlichere Darstellung findet sich im Benutzerhandbuch /BOU 18/.

**Nomenklatur**

**Composition**: Chemische Zusammensetzung, gekennzeichnet durch die Dichten der darin vorkommenden Nuklide.

**Material**: Zweidimensionale Anordnung verschiedener Compositions, z. B. ein Brennelement-Querschnitt. Da diese Materialien im Kernsimulator durch homogenisierte Wirkungsquerschnitte gekennzeichnet sind, bilden sie die Bausteine des dreidimensionalen Kernmodells.

**BE-Typ**: Ein durch eine axiale Abfolge von Materialien gekennzeichneter Typ eines Brennelements.

**Programmpakete**

KMACS ist ein in Python 2 /PYT 18/ geschriebenes Softwarepaket. Die verschiedenen Aufgaben, die in KMACS durchgeführt werden, verteilen sich auf die folgenden Python-Pakete:

- `kcommon`. Paket mit gemeinsamen, d.h. in verschiedenen Paketen genutzten Funktionen. Die Klasse `Configuration` dient zum Einlesen und Übersetzen des KMACS-Eingabedatensatzes sowie zu dessen Bereitstellung in anderen Paketen. Die Klasse `Core` stellt die geometrischen und Beladeinformationen des Reaktorkerns bereit.
- `klib`. Datenbank-Paket. Enthält die Funktionen für den Datenbankzugriff in der Klasse `Database`.
- `kspect`. Paket zur Wirkungsquerschnittserzeugung. Enthält die Klassen `Composition` zur Handhabung chemischer Zusammensetzungen (Übersetzen ver-

schiedener möglicher Eingabedaten in die für Gittercodes übliche Angabe Nuklide je barn-cm, d.h. je $10^{-24}$ $cm^3$, Generierung der Zusammensetzung des Moderators als Funktion von Moderatordichte und Borkonzentration), `Material` zur Gittercode-unabhängigen Prozessierung von Material-spezifischen Größen (Geometrien, Zustandspunkte und zugehörige Compositions, Zonennummerierung, Behandlung von Abstandhaltern und Steuerstäben), sowie `XsGenerator` zum Erstellen der Eingabedatensätze für den Gittercode, Gittercode-Ausführung und Auslesen der Gittercode-Ergebnisse.

- `kflux`. Paket zur Ansteuerung von Flusslöser und Thermohydraulikcode in der Klasse `FluxSolver`.
- `kburn`. Paket zum Ansteuern des Abbrandcodes in der Klasse `BurnupSolver`.
- `kvip`. Paket zur Visualisierung und Nachverarbeitung der in der Datenbank gespeicherten Ergebnisdaten.

Für die Steuerung des Simulationsablaufs werden mit KMACS 1.0 die Python-Skripte `generateXS.py` zur WQ-Erzeugung und `runCycle.py` zur Zyklusrechnung bereitgestellt. Die Interaktion der KMACS-Pakete ist in Abb. 2.1 dargestellt.

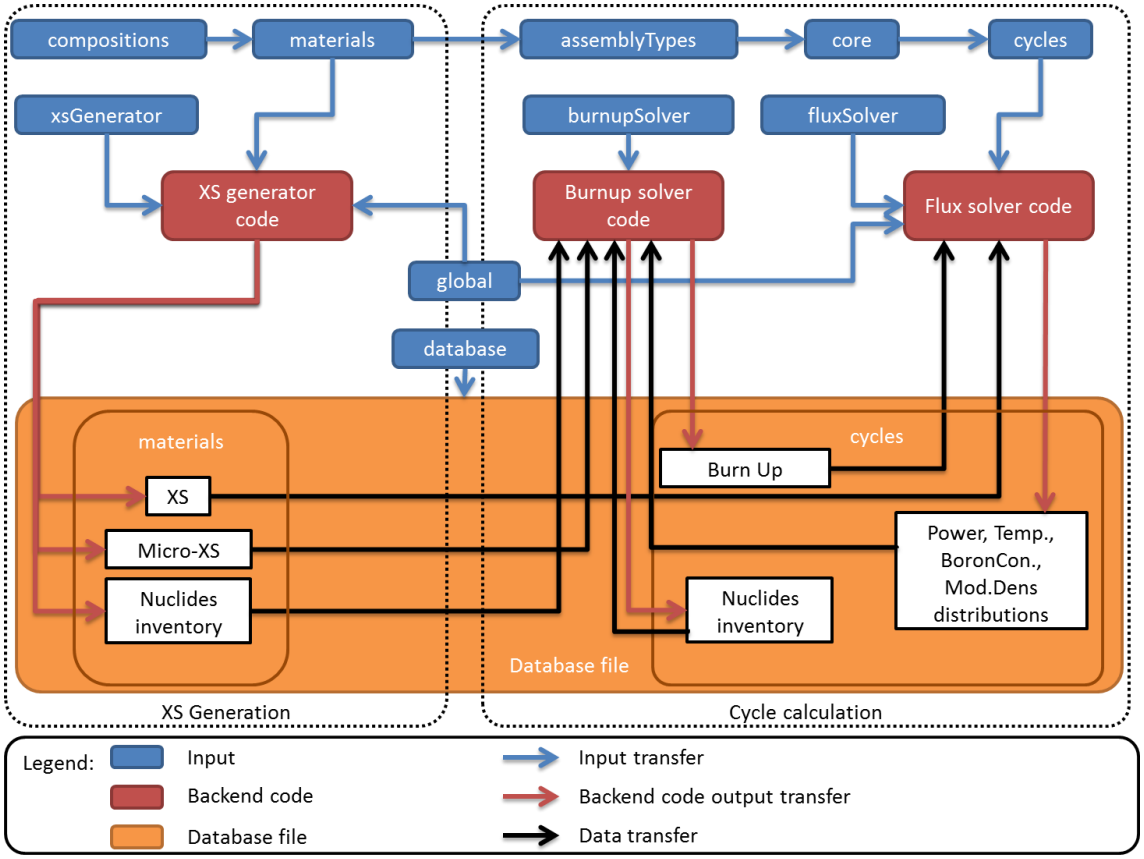


**Abb. 2.1** Interaktion der KMACS-Pakete

### Ablauf der Vorausrechnung

Die im KMACS-Eingabedatensatz unter den Abschnitten `compositions`, `materials`, und `xsGenerator` bereitgestellten Informationen werden von der Klasse `XsGenerator` herangezogen, um den Gittercode anzusteuern. Dieser erzeugt für jedes Material die Weniggruppenkonstanten (WQ, englisch: XS), die Assembly discontinuity factors (ADF) und nuklidspezifischen Wirkungsquerschnitte (micro-XS) in Abhängigkeit von Abbrand (*burnup*) und den antizipierten thermohydraulischen Rückwirkungszuständen (*branching*). Die Ergebnisse werden im `materials`-Abschnitt der Datenbank abgelegt. Falls das Nuklidinventar in der KMACS-Rechnung mitberücksichtigt werden soll, wird dieses von der Klasse `BurnupSolver` durch Abbrand des frischen Nuklidinventars jedes Materials unter nominellen Reaktorbedingungen bereitgestellt.

**Ablauf der Zyklusrechnung**

Die Zyklusrechnung findet durch abwechselnden Aufruf von Flusslöser und Abbrandcode statt.

**Ansteuerung des Flusslösers**. Die im KMACS-Datensatz unter den Abschnitten `assemblyTypes`, `core` und `cycles` bereitgestellten Informationen werden von der `Core`-Klasse in ein Modell des Reaktorkerns mit der aktuellen Kernbeladung übersetzt. Diese verwendet die Klasse `FluxSolver` zusammen mit den Informationen unter `fluxSolver`, der aktuellen Abbrandinformation im Eintrag „burnup“ aus der Datenbank und den XS aller im Kern eingeladenen Materialien, um den Flusslöser (wenn gewünscht mit externer Thermohydraulik) anzusteuern. Das Ergebnis der Flusslöser-Rechnung ist der aktuelle Kernzustand, gekennzeichnet durch die dreidimensionale Verteilung von Leistung, Neutronenfluss, Brennstoff- und Moderatortemperatur, Moderatordichte und Borkonzentration. Diese Verteilungen werden in der Datenbank im Abschnitt „cycles“ beim aktuellen Zeitpunkt des Betriebszyklus abgelegt.

**Ansteuerung des Abbrandcodes**. Die aktuelle Abbrandverteilung und zugehörige Nukliddichten sowie der aktuelle Kernzustand werden zusammen mit den Informationen unter `burnupSolver` von der Klasse `BurnupSolver` verwendet. Falls es sich um den ersten mit KMACS betrachteten Abbrandpunkt der im Zyklus eingesetzten Brennelemente handelt, wird das Nuklidinventar aus dem unter „`materials` in der Datenbank gespeicherten zum aktuellen Abbrandwert hin linear interpoliert. D.h. es wird angenommen, dass die betreffenden Brennelemente während ihrer bisherigen Einsatzzeit nominalen Reaktorbetriebsbedingungen ausgesetzt waren. Mit den vorgenannten Informationen ermittelt der Abbrandcode das für den nächsten Zeitpunkt gültige Nuklidinventar sowie die Abbrandverteilung, die im Abschnitt „cycles“ der Datenbank gespeichert werden.

# 3 Durchgeführte Arbeiten

Dieses Kapitel beschreibt die im Vorhaben durchgeführten Arbeiten. Da die allermeiste Arbeit Programmiertätigkeiten zu den in Kapitel 2 benannten Python-Paketen umfasst, ist dieses Kapitel entsprechend diesen Paketen untergliedert.

Die Aufschlüsselung nach den in der Vorhabensbeschreibung benannten Arbeitspaketen AP 1 „Weitere Validierung des Kernsimulators an DWR-Anlagendaten“, AP 2 „Einbindung eines alternativen Spektralcodes in den Kernsimulator“ und AP 3 „Bereitstellung einer Release-Version des Kernsimulators“ kann anhand der Quartalsberichte nachvollzogen werden.

## 3.1 Software-Engineering

Als ein auf Dauer angelegtes Software-Entwicklungsprojekt erfüllt KMACS die Standards der Qualitätsmanagement-Richtlinie zur Programmentwicklung in der GRS /SMG 17/. Es wird mit dem SVN Versionskontrollsystem verwaltet und verfügt über einen programmspezifischen QS-Plan /ZIL 17/. Gemäß einer der Maßgaben des QS-Plans für den GRS-Kernsimulator KMACS verfügt jedes Python-Paket über eine Reihe von Unit-Tests, die einzelne Funktionen auf ihre Korrektheit überprüfen, und Sample Test Cases, die das Paket insgesamt testen. Für die routinemäßige Überprüfung des Testerfolgs wurde im Laufe des Vorhabens eine Reihe von Konventionen bezüglich der zu verwendenden Tests eingeführt, die es ermöglichen mit der Software `pytest` die Pakete zu testen. Damit ist es auf einfache Art und Weise möglich, die im Laufe des Vorhabens eingeführten Tests vor jedem Commit (d.h. dem Einstellen einer Coderevision ins Versionskontrollsystem) auf Erfolg zu prüfen.

Um einen Eingabedatensatz schon vor der Laufzeit zu überprüfen, wurde zunächst exemplarisch in den Modulen `kcommon` und `klib` die Eingabedatenvalidierung nach JSONSchema eingeführt. Diese Eingabedatenvalidierung überprüft den Eingabedatensatz auf Vollständigkeit (sind alle notwendigen Parameter gesetzt, auch inklusive Wenn-dann-Abhängigkeiten) und auf korrekten Datentyp (ein Parameter muss ein Zahlwert/eine Liste von Zahlen/ein Wert aus einer vorgegebenen Liste/ein boolescher Parameter sein). Bei nichterfüllter Validierung wird der Anwender auf die Stelle hinge-

wiesen, an der sein Eingabedatensatz zu verbessern ist. Diese Validierung nach JSONSchema soll in Zukunft auf alle Pakete ausgedehnt werden. Das JSONSchema kann in geeigneten Editoren auch dazu verwendet werden, schon während der Eingabe den Datensatz auf Korrektheit zu überprüfen bzw. dem Nutzer die mögliche Parameterwahl eines Eintrags anzuzeigen.

Das mit dem QS-Plan entstandene KMACS Developer Manual /KÜN17/ schreibt einheitliche Quellcode-Dokumentation vor. Damit kann durch das Software-Dokumentationswerkzeug Sphinx automatisiert eine stets aktuelle Dokumentation der Programmschnittstellen erzeugt werden.

Zum Vorhabensabschluss wurde gemäß QS-Plan ein Code-Review durchgeführt, bei dem die Einhaltung der Programmierrichtlinien und Code-Konventionen überprüft wurde.

Die aktuelle SVN-Revision wurde als Release-Version eingefroren. Mit der Software Cython wurden die KMACS-Pakete in binäre Form überführt.

Am Paket `kcommon` wurden folgende Arbeiten vorgenommen.

**Neues Eingabeformat**

Das im Prototyp verwendete JSON-Eingabeformat hatte sich wie bereits oben erwähnt als fehleranfällig herausgestellt. Zwar weist jenes Format die gewünschte Flexibilität (mögliche Freitexteingabe, Verwendung von beliebigen Zahlenformaten, mögliche Verwendung von booleschen Variablen) auf und kann daher leicht als verallgemeinerte Eingabe der verschiedenen Backend-Code-Eingaben realisiert werden. Jedoch verwendet das JSON-Format eine spezifische Syntax der Gruppierung in geschweifte Klammern und Trennung verschiedener Gruppen durch Kommata sowie verschiedener Hierarchieebenen durch strikte Einrückung, sodass leicht Formatfehler unterlaufen können, die nicht einfach per Programm abzufangen sind.

Daher wurde im Laufe des Vorhabens ein neues Eingabedatenformat, welches auf YAML /YAM09/ basiert, eingeführt. Dieses Format ist einfacher lesbar und editierbar und verfügt des Weiteren über die folgenden Vorzüge:

- Möglichkeit von Referenzen. Teile des Inputs können mit Markern versehen werden, auf die dann an anderen Stellen per Referenz zurückgegriffen werden kann. Dies verhindert Falscheingaben, da gleichartige Eingabe referenziert werden kann und nicht wiederholt werden muss.

    - Möglichkeit programmspezifischer Syntax. Spezielle Zeichen(kombinationen) können mit einer Bedeutung versehen werden. Diese wird dann im Moment des Einlesens des Datensatzes angewandt.

Beide Vorzüge wurden für KMACS folgendermaßen benutzt:

- Neben den Referenzen wurde in `kcommon` die Möglichkeit implementiert, den Eingabedatensatz auf verschiedene Dateien zu verteilen. Die Einleseroutine verkettet die Dateien dann dergestalt, dass alle Referenzen ihrer Abhängigkeit entsprechend aufgelöst werden können. Dies erleichtert die Verwendung von Referenzen und die Vermeidung wiederholter Eingabe noch zusätzlich, da der Benutzer somit nicht darauf achten muss, ob der referenzierte Teil der Eingabe vor der Referenz in der Datei definiert wurde.

- Die Möglichkeit der erweiterten Syntax wurde dazu genutzt, es dem Benutzer freizustellen, physikalische Größen mit der Einheit seiner Wahl zu versehen. Ordnet er mittels `!unit <Wert> <Einheit>` eine Einheit zu, so wird diese automatisch beim Einlesen in die KMACS-Basiseinheit umgerechnet. Dies kann besonders nützlich sein, wenn etwa die Angaben für Drücke in psi und Temperaturen in °F gegeben sind, was v.a. in den USA übliche Größeneinheiten sind. Das für die Einheitenumwandlung zuständige Modul `units.py` kann auch für die Umwandlung von KMACS-Basiseinheit in die Einheit eines Backend-Codes verwendet.

- Des Weiteren wurde der `!range` Syntax implementiert. Damit kann beispielsweise ein BE-Typ einer Reihe von Brennelementen zugewiesen werden, deren Name nicht nur aus einer Zahl besteht, etwa wird `!range 1501:1541 A%` in die Liste A1501, A1502, …, A1540 übersetzt.

Schließlich wurde die Routine zum Einlesen des Eingabedatensatzes mit einer Routine zum Überprüfen der Syntax ausgestattet. Bei syntaktisch falscher Eingabe wird der

Nutzer nun mit einer Fehlermeldung darauf hingewiesen, in welcher Zeile der Eingabe ein Eingabefehler vorliegt.

### Flexiblere `Core`-Klasse

Im Vorhabensverlauf wurde auch die Core-Klasse, die die geometrischen und Belade-Informationen für die anderen Klassen bereitstellt, überarbeitet. Dies war aus folgenden Gründen geboten.

1. Die `Core`-Klasse im Prototyp ordnete jedem Reflektor-Node ein Material abhängig vom nächsten Brennelement zu, genauso wie sie für die Brennelemente am Kernrand spezielle Materialien vorsah. Im Lauf der Validierung im Vorgängervorhaben stellte sich jedoch heraus, dass die WQ der Reflektormaterialien nur in geringem Maße von den Eigenschaften des benachbarten BEs abhängen und eine spezielle Behandlung der Reflektornachbar-BE nicht konsistent mit den für die nodale Kernrechnung benutzten ADF ist. Spezielle Reflektornachbar-Materialien hätten die KMACS-Rechnung also unnötig verkompliziert. Diese Annahmen über die Reflektoren in der `Core`-Klasse sollten entfallen.
2. Zwecks möglichen Rechenzeitgewinns sollten auch ¼-Kernmodelle möglich werden, sofern der Kern eine entsprechende Symmetrie aufweist. Dies ermöglicht eine deutliche Reduktion der Rechenzeit von Flusslöser und Abbrandcode.
3. Für den Fall einer Erweiterung auf hexagonale Brennelemente sollte die `Core`-Klasse ohne fundamentale Änderungen weiter benutzbar bleiben.

Alle drei Punkte wurden umgesetzt in der neuen Implementation. Da nun ein Reflektormaterial nur noch dort gesetzt wird, wo der Nutzer explizit einen Reflektor definiert hat, sind nun auch die Simulationen von Modellen ohne Reflektor (z. B. Schachbrettanordnungen) bzw. ohne Top- und Bottom-Reflektor (2D-Simulation) möglich.

### Einheitliche Logging-Funktion

Im Paket `kcommon` wurde die `Logit`-Metaklasse für ein Paket-übergreifendes, einheitliches Logging implementiert. Mit Logging ist hier das Schreiben von Log-Dateien gemeint, die über den Programmfortschritt bzw. Fehler im Programmablauf während der Laufzeit buchführen. Wird ein erwarteter Punkt im Programmablauf erreicht, kann über

die von Logit bereitgestellten Routinen eine Meldung geschrieben werden. Diese Meldungen („log messages“) sind einheitlich im Format und enthalten die Information darüber, von welcher Klasse sie aus welchem Grund geschrieben wird. Je nach antizipiertem Programmverlauf kann die Meldung die Kategorien `Debug`, `Info`, `Warning`, `Error` oder `Fatal` annehmen und entsprechend weiterverarbeitet werden. Beispielsweise kann eine Log-Message der Kategorie `Error` zum geordneten Programmende genutzt werden.

Die log messages sind für die Nutzerinformation, aber auch für die Code-Weiterentwicklung wichtig, da sie einen Hinweis auf gemäß QS-Plan zu implementierende Testfälle geben können.

## 3.2 Arbeiten am Paket `klib`

Wie oben unter 1.2 genannt, wies die Datenbank des Prototyps das Problem eines sehr langsamen Lese- und Schreibzugriffs auf. Auch der Speicherplatzverbrauch sollte verringert werden. Ein weiteres Problem war, dass beim Prototyp ein Programmabsturz häufig zu einem vollständigen Datenverlust führte.

Eine Problemanalyse zu Vorhabensbeginn ergab folgende Schwachstellen in der bisherigen Implementation:

1. Die Zyklusdaten (Datenbankabschnitt „cycles“) wurden in Unterordnern für jeden Reaktornode abgelegt. Damit war es möglich, ohne Kenntnis der Eingabedatei für KMACS den Reaktorzustand in jedem Zeitpunkt eines Betriebszyklus allein mit der Datenbank zu analysieren. Der Nachteil dieser Anordnung der Daten in Unterordnern war jedoch, dass für jeden dieser (typischerweise mehr als 3.000) Unterordner unabhängige Variablen für den Reaktorzustand mit ihren jeweiligen Metadaten angelegt werden mussten. Dies führte zu großem Speicherplatzbedarf.
2. Die Anordnung der Daten in Unterordnern hatte außerdem zur Folge, dass der Reaktorzustand nicht in einzelnen Arrays für Brennstofftemperatur, Moderatordichte etc. an die Datenbank übergeben werden konnte, sondern in einer

Schleife über die einzelnen Nodes des Reaktors je nur ein Wert geschrieben werden musste.

3. Beim Lesen und Schreiben wurde jede einzelne Anfrage an die Datenbank direkt in eine Anfrage an die Dateiinstanz übersetzt. Damit wurde insbesondere beim Schreiben des kleinteilig übergebenen Reaktorzustands eine Vielzahl an Schreibvorgängen ausgelöst.

Die Analyse führte zu folgendem Datenbank-Konzept, das in KMACS 1.0 umgesetzt ist:

1. Anordnung der Zyklusdaten in großen Arrays. Die den Reaktorzustand kennzeichnenden Größen werden je in Arrays der Größe Zeitschritte x Reaktornodes abgelegt und können so nach jedem Flusslöser- bzw. Abbrandcode-Schritt auf einmal in die Datenbank übertragen werden. Ein Zugriff auf die Daten eines einzelnen Nodes ist über die neue `Core`-Klasse immer noch möglich.
2. Puffern von Lese und Schreibvorgängen. Lese- und Schreibvorgänge auf die Festplatte werden durch das Verwenden von Puffern im Hauptspeicher verringert. Erst wenn beispielsweise alle Einträge eines Zeitschritts an die Datenbank-Klasse übergeben wurden, wird ein Festplatten-Schreibvorgang ausgelöst. Damit werden auch unvollständige Schreibvorgänge vermieden, die zu Datenverlust führen können.
3. Möglichkeit des Verlinkens mehrerer Datenbanken. Um Festplattenspeicherplatz zu reduzieren wurde auch das Verlinken mehrerer Datenbanken ermöglicht. So können zum Beispiel die WQ von BE-Typen, die in mehreren Zyklen eingesetzt wurden in einer Datei gespeichert werden. Wenn dann auf diese WQ zwecks Simulation eines Betriebszyklus zugegriffen werden muss, müssen diese nicht mehr Teil derselben Datei wie die Zyklusdaten sein, sondern können durch einen Link hinzugefügt werden.

Außer den genannten technisch bedingten Änderungen an der Datenbank-Konzeption wurde auch die Aufteilung von Abbrand und Nuklidinventar nach verschiedenen Brennstab-Typen innerhalb eines Nodes bzw. Materials aufgegeben. Diese Unterteilung, die im Prototyp vorhanden war, erwies sich als für einen nodalen Kernsimulator nicht sinnvoll, da aus Stabzell-aufgelösten Informationen nur mit erneuten Gittercode-

Rechnungen während des Zyklusabbrands neue homogenisierte WQ erzeugt werden könnten, siehe auch Abschnitt 3.5.

Mit diesen Änderungen wurden die Speicherplatzanforderungen um bis zu 75 % reduziert. Die Zugriffszeiten auf die Datenbank spielen in KMACS 1.0 im Vergleich mit den Laufzeiten der Backend-Codes keine Rolle mehr.

## 3.3 Arbeiten am Paket `kspect`

Ein wesentlicher Teil der Arbeiten am Paket `kspect`, nämlich die Einbindung eines alternativen Spektralcodes, war mit dem Arbeitsprogramm dieses Vorhabens vorgegeben.

### Einbau eines alternativen Spektralcodes

Der alternative Spektralcode sollte einerseits zu einer Reduktion der Rechenzeit führen, andererseits das Prinzip der Modularität (d.h. Austauschbarkeit von Backend-Codes) in KMACS unterstreichen.

Zunächst wurden mit SCALE-POLARIS /SCA 18/, HELIOS /STU 12/, nTRACER /JUN 13/ und Serpent /LEP 15/ vier Gittercodes miteinander hinsichtlich ihrer Eignung in KMACS verglichen, s. Abschnitt 4.2. Die Wahl fiel auf den Code HELIOS:

Um diesen Code als modulare Alternative zu SCALE-NEWT in KMACS einzubeziehen, wurden diejenigen Aufgaben des `kspect`-Pakets, welche unabhängig vom gewählten Gittercode zu erfüllen sind, in die Klassen `Composition` und `Material` überführt. In diesen werden die jeweiligen Compositions und Geometrie-Details der Materialien Backendcode-unabhängig verwaltet, s.u. in Implementierung der Klassen `Composition` und `Material`.

Anschließend wurden sowohl für den bisherigen Gittercode SCALE-NEWT wie für HELIOS die Module `scaleBackend.py` bzw. `heliosBackend.py` implementiert, die die in der Klasse `Material` vorgehaltenen Informationen in NEWT- bzw. HELIOS-

Eingabedatensätze überführen und die Codeausführung triggern sowie die Ausgabedateien lesen und relevante Daten in die Datenbank übertragen.

Da der HELIOS-Code über keine eingebaute Funktion zur Errechnung der ADF besitzt, wurde diese in `heliosBackend.py` implementiert. Wird dieselbe Funktion auf NEWT-Ergebnisse angewandt, so ergeben sich speziell für Reflektor-ADF Werte, die sich von denjenigen der NEWT-Ausgabe unterscheiden. Um dem Nutzer für den Gittercode NEWT die Wahl zu überlassen, ob die von NEWT oder von KMACS ermittelten Reflektor-ADF verwendet werden sollen, wird diese Funktion auch für `scaleBackend.py` bereitgestellt.

**Implementierung der Klassen `Composition` und `Material`**

Die Klasse `Composition` stellt die Nukliddichten bereit. Dazu muss einerseits die Nutzereingabe (die verschiedene Möglichkeiten der Beschreibung chemischer Zusammensetzungen kennt, s. /BOU 18/) in die einheitliche Angabe Anzahl Nuklide/Volumen überführt werden und andererseits eine Routine zur Dichte-abhängigen Bereitstellung dieser Anzahl Nuklide/Volumen für den Moderator (inkl. Bor) bereitgestellt werden, um die Moderatordichte- und Bor-Branchzustände repräsentieren zu können.

Dafür ist im `kspect`-Paket eine auf den Angaben in /SCA 18/ basierende Liste der Nuklidmassen und der natürlichen relativen Häufigkeiten der einzelnen Isotope eines Elements hinterlegt. Anhand dieser Angaben kann sowohl aus der Nutzerangabe einer Composition in Gewichtsprozent wie einer Angabe in Volumenprozent die gewünschte Nuklidkonzentration errechnet werden.

Die Klasse `Material` stellt den verschiedenen Gitter-Backendcodes die Informationen über Geometrie und Composition eines Materials bereit. Gleichzeitig müssen auch die Informationen über die Temperatur aller Zonen im Branchzustand sowie, ob eine Zone abbrandabhängig betrachtet werden muss (Brennstoff/Neutronengift), vorgehalten werden. Diese Informationen werden sämtlich aus der Klasse `Composition` sowie der Benutzereingabe abgeleitet. Die Klasse `Material` stellt auch die Funktion zum Verschmieren der Abstandhalter im Moderatorvolumen des Brennelements bereit. Für die Zonen eines Materials, die vom Moderator eingenommen werden, stellt die `Material`-

Klasse eine effektive Mischung aus Abstandhalter und Moderator an das Gittercode-Backend bereit.

Die Volumenanteile der Zonen eines Materials werden auch benutzt um die nuklidweisen Zweigruppen-WQ über das Material zu homogenisieren.

Die `Material`-Klasse wird über das Paket `kspect` hinaus auch vom `QuaboxCubboxAthlet`-Backend des Pakets `kflux` für die Berechnung der durchströmten Fläche und des benetzten Umfangs des Parallelkanalmodells sowie von der `BurnupSolver`-Klasse für die Angabe der Schwermetallmasse jedes Nodes benutzt.

Um die Funktion der `Composition`- und `Material`-Klasse zu verifizieren, wurden entsprechende Tests implementiert.

**Anpassung an die neue Struktur der Datenbank**

Da sich im Laufe des Vorhabens die Struktur der Datenbank, und speziell auch die Ansteuerung derselben im Paket `klib` geändert haben, mussten auch die Zugriffe auf die Datenbank in `kspect` modifiziert werden.

Wesentliche Änderung dabei ist der bereits in Abschnitt 3.2 genannte Wegfall der Brennstab-bezogenen nuklidweisen Wirkungsquerschnitte. Diese nuklidspezifischen WQ werden in KMACS 1.0 über das Material homogenisiert ermittelt und gespeichert. Dazu wird die Ausgabe des Gittercodes bezüglich Stabzell-spezifischer WQ mit dem Stabzell-Neutronenfluss und dem aus der `Material`-Klasse vorhandenen Stabzell-Volumenanteil gewichtet. Das Ergebnis wird in der Datenbank gespeichert.

**Einbinden des `CoolProp`-Pakets**

In KMACS werden die antizipierten Rückwirkungszustände im Reaktorkern (Branch-Zustände) nach Brennstofftemperatur, Moderatordichte und Borkonzentration tabelliert. Im Druckwasserreaktor sind die Moderatordichte und –temperatur bei einem angenommenen thermodynamischen Gleichgewicht über den konstanten Systemdruck miteinander verknüpft. Werden als Branch-Zustände verschiedene Moderatordichten definiert, so weisen diese bei vorgegebenem Systemdruck auch verschiedene definierte

Moderatortemperaturen auf. Um diese für einen beliebigen Systemdruck bereitzustellen (und also eine flexible Benutzereingabe zu erlauben) wurde das externe Paket `CoolProp` /BEL 14/ in das `kspect`-Modul `waterData.py` eingebunden, welches entsprechende Daten der Wasser-Dampf-Tafel bereitstellt.

`CoolProp` wird mittels `waterData` ebenfalls dazu verwendet, um den konstanten Volumenstrom der Hauptkühlmittelpumpen in Abhängigkeit der Kerneintrittstemperatur in einen Kernmassenstrom zu übersetzen, s. Abschnitt 4.13.

## 3.4 Arbeiten im Paket `kflux`

Das Paket `kflux` dient der Ansteuerung des nodalen Flusslösers für die Zyklusrechnung.

### Eingliederung des ehemaligen Pakets `kthermo`

Die Aufgaben, die derzeit in diesem Paket durchgeführt werden, waren im Prototyp auf die Pakete `kflux` für die Erstellung der Eingabe des Diffusionscodes QUABOX/CUBBOX und `kthermo` für die Erstellung der Eingabe des Thermohydraulikcodes ATHLET sowie das Auslesen der Ergebnisse verteilt. Da jedoch in KMACS das gekoppelte Codesystem QUABOX/CUBBOX-ATHLET, das eine simultane Lösung von Diffusionsgleichung und Thermohydraulik ermittelt, eingebunden ist und ein alternativer Backend-Code ebenso simultan Neutronik und Thermohydraulik behandeln würde, wurde die Aufteilung in zwei Pakete als dem Kernsimulator nicht angemessen aufgehoben.

Daher wurden diejenigen Funktionen, die zuvor im Paket `kthermo` vorhanden waren in ein Modul `athlet.py` des Pakets `kflux` überführt. Je nach Eingabe wird sodann von der Klasse `FluxSolver` die Eingabe für Standalone-Neutronik oder gekoppelte Neutronik-Thermohydraulik erzeugt und das jeweilige Ergebnis ausgelesen.

### Anpassung an die neue Datenbank- und `Core`-Klasse

Mit den neuen Zugriffsroutinen aus `klib` und `Core` mussten die entsprechenden Aufrufe in `kflux` angepasst werden. Insbesondere für das Lesen und Schreiben des aktuellen Kernzustands ergeben sich deutliche Vereinfachungen, da nun nicht mehr Schleifen über sämtliche Nodes des Reaktorkerns erforderlich sind, sondern das Array der Rückwirkungsparameter bzw. des Abbrands auf einmal übertragen werden. Auch die Mittelung der früher stabzellweise abgespeicherten Abbrandwerte entfällt.

### Nutzung der `Material`-Klasse aus `kspect`

Die neu implementierte `Material`-Klasse aus `kspect` stellt Material-spezifisch die durchströmte Fläche und den benetzten Radius bereit. Um unnötige Code-Dopplungen zu vermeiden, wird diese Klasse nun auch in `athlet.py` aus `kflux` verwendet, um entsprechende Parameter im ATHLET-Input zu setzen.

### Implementierung eines Abbrand-abhängigen Gasspalt-Maßes

Im Zuge der Validierung (s. Abschnitt 4.3) wurde als wesentlicher Einflussfaktor auf die mittlere Brennstofftemperatur die Weite des Gasspalts zwischen Brennstofftablette und Hüllrohr erkannt. Neben der thermischen Ausdehnung, die in KMACS 1.0 in die geometrischen Eingabeparameter (und entsprechend in die Massendichten) eingerechnet werden muss, wirken sich auch der Abbrand und die momentane Leistungserzeugung auf die Abmessung des Gasspalts aus. Ein detailliertes Modell hierzu verwendet z. B. der Brennstabcode FRAPCON /USD 97/.

Da die Leistung der Nodes vor der Rechnung des Flusslösers nicht bekannt ist, wurde in `kflux` nur der Abbrand-abhängige Teil der Gasspalt-Weite implementiert, s. /BOU 18/.

### Implementierung von Routinen zur Bewegung von Steuerelementen

Für jeden Zeitpunkt des Zyklusabbrands können die Eintauchtiefen der Steuerbänke im Eingabedatensatz konfiguriert werden. Um diese Konfiguration auf die Eingaben des Flusslösers zu übertragen, wurden entsprechende Funktionen implementiert.

**Implementierung von Routinen zur Änderung der Kerneintrittstemperatur und des Massenstroms**

Wie die Steuerbankpositionen können auch die relative Leistung und die Kern-Eintrittstemperatur des Kühlmittels in der Eingabe definiert werden. Während die Leistung und die Kerneintrittstemperatur unmittelbar in die Eingabe des Flusslösers übertragen werden können, ändert sich auch der Massenstrom durch den Kern mit der Eintrittstemperatur, da die Hauptkühlmittelpumpen üblicherweise mit konstanter Drehzahl und somit konstantem Volumenstrom arbeiten. Der sich mit konstantem Volumenstrom bei Änderung der Eintrittstemperatur geänderte Massenstrom wird mit der entsprechenden Funktion aus `waterData.py` des Pakets `kspect` ermittelt.

**Erweiterung der Schnittstelle zum Flusslöser (hinsichtlich DYN3D)**

Um die Modularität des Kernsimulators auch im `kflux`-Paket nutzen zu können, wurde die Schnittstelle zum Flusslöser in `kflux` entsprechend vorbereitet.

Der Vollständigkeit halber sei hier erwähnt, das im Rahmen des Vorhabens 4716R01355 (BE-Verbiegungen) DYN3D /GRU 05/ als ein weiterer Flusslöser als Backend in KMACS implementiert wurde. DYN3D ist ein nodaler Diffusionscode mit eingebauter Kern-Thermohydraulik. Er kann für quadratische und hexagonale Brennelementgeometrie verwendet werden und bietet somit Potential für eine spätere Erweiterung von KMACS auf WWER-Reaktoren. Des Weiteren können in DYN3D Seiten-abhängige ADF verwendet werden, was insbesondere für eine realistischere Beschreibung asymmetrischer oder verbogener Brennelemente wichtig ist. Die Validierung des DYN3D-Backend in KMACS wird ebenfalls im Vorhaben 4716R01355 durchgeführt.

## 3.5 Arbeiten im Paket `kburn`

Die Arbeiten im Paket umfassten den Einbau eines trivialen Backends, die Anpassungen an die neue Datenbankstruktur, die Aktualisierung auf eine neue VENTINA-Version und die Verbesserung der Interpolationsfunktion für nuklidspezifische WQ.

### Einbau eines trivialen Burnup-Backends

In vielen DWR-Kernsimulatoren wird das Nuklidinventar nicht Node für Node errechnet, sondern nur über den Abbrandwert berücksichtigt. D.h. es wird von der Gültigkeit der Näherung ausgegangen, dass sich das lokale Inventar nicht wesentlich von demjenigen unterscheidet, das sich bei Abbrand unter nominellen Reaktorbedingungen ergibt.

Damit wird der Abbrand einfach als Integral der Leistung über die Zeit in Bezug auf die Schwermetallmasse eines Nodes ermittelt. Dies wurde im Modul `trivialBackend.py` des Pakets implementiert. Dabei wird die Leistungsverteilung zu Beginn des Zeitschritts für die Abbrandrechnung als konstant über den Zeitschritt angenommen, s. auch /BOU 18/. Die Schwermetallmasse der Nodes wird von der Klasse `Material` aus dem Paket `kspect` bereitgestellt.

Große Teile der KMACS Validierung wurden mit diesem Abbrandmodul durchgeführt, s. die Zyklusrechnungen zu Reaktor A in /ZIL 18/.

### Anpassungen des VENTINA-Backends an die neue Datenbank-Struktur und neue Interpolationsroutinen

Da die nuklidspezifischen WQ nicht mehr per Brennstab-Typ sondern homogenisiert über das Material vorgehalten werden, mussten auch die Routinen zum Lesen und Schreiben der Nukliddichten für VENTINA angepasst werden.

Da mit den Änderungen der Datenbankstruktur der Kernzustand nicht mehr Node-weise sondern in großen Arrays abgelegt ist, kann auch die Interpolation der nuklidweisen WQ vereinfacht und für alle Nodes auf einmal realisiert werden. Damit gehen auch die benötigte deutliche Reduktion des Hauptspeicherplatzbedarfs und der Rechenzeit einher.

## 3.6 Implementierung des Pakets `kvip`

Der Kernsimulator-Prototyp verfügte über die Möglichkeit einer dreidimensionalen Visualisierung der Kernbeladung sowie von Ergebnisgrößen. Für die KMACS-Validierung sind darüber hinaus auch Vergleiche von Simulationsergebnissen mit Messgrößen o-

der Referenzrechnungen in einem Plot von Bedeutung. Auch die statistische Auswertung von Simulationsergebnissen ist wesentlich.

Daher wurden die bisherigen Visualisierungsmöglichkeiten erweitert und mit weiteren Visualisierungs- und Postprocessing-Tools im Paket `kvip` (KMACS visualization and postprocessing) zusammengeführt. Viele Beispiele der Visualisierung sind in /ZIL 18/ enthalten.

**Einführung von ein- und zweidimensionalen Plots**

Unter Verwendung von Routinen des externen Pakets `matplotlib` wurden Routinen zur einfachen Visualisierung von Ergebnissen in Form von eindimensionalen (Ergebnisgröße entlang eines Brennelements oder radial gemittelt bzw. als Funktion von Zeit oder Abbrand) und zweidimensionalen Plots (Darstellung von axial gemittelten Größen bzw. Größen in einer axialen Ebene auf dem Querschnitt des Reaktorkerns).

**Mittelung physikalischer Größen mit Wichtungsfaktoren**

Um die Mittelung korrekt auszuführen, wurden Routinen zum Filtern (Beispiel: Gehört ein Node zu einer Ebene, handelt es sich um einen Brennelement- oder Reflektor-Node) und Wichten implementiert. So wird beispielsweise die jeweilige Schwermetallmasse bei der Mittelung des Abbrands über einen Node-Bereich als Gewicht verwendet.

# 4 Fortgang der Validierung

Für den Prototyp wurden im Vorgängervorhaben vereinzelte Validierungsrechnungen durchgeführt. Siehe /ZIL 15/.

Diese umfassten:

- Vergleiche zwischen den automatisiert mit SCALE-NEWT erzeugten WQ und den manuell mit HELIOS erzeugten WQ
- Vergleiche zwischen der Entwicklung des Nuklidinventars aus SCALE-NEWT mit demjenigen aus `kburn`-VENTINA
- Vergleiche der mit KMACS ermittelten kritischen Borkonzentration zweier Zyklen mit den Messdaten
- Vergleiche der radialen Leistungsverteilung aus KMACS mit Referenzrechnungen für einige Zykluszeitpunkte zweier Betriebszyklen
- Vergleiche der axialen Leistungsverteilung aus KMACS mit dem Kugelmessprotokoll für zwei Brennelemente zu zwei Zykluszeitpunkten eines Zyklus

Gerade die Zyklusrechnungen zeigten zwar qualitativ richtige Ergebnisse, wichen quantitativ im Einzelnen aber noch deutlich von der Referenz ab. Gleichzeitig basierten sämtliche Rechnungen auf demselben Reaktor.

## 4.1 Erweiterung der Datenbasis für die Validierung von Zyklusrechnungen

Angesichts der begrenzten Menge an Referenzdaten wurde zu Vorhabensbeginn eine Bestandsaufnahme an in der GRS vorhandenen geeigneten Daten für die Validierung durchgeführt. Es stellte sich heraus, dass neben den bereits im Vorgängervorhaben verwendeten Daten zu einem Vor-KONVOI-Reaktor nur Daten aus dem internationalen BEAVRS-Benchmark /HOR 13/ in für die Validierung geeigneter Qualität vorhanden waren. Dieses Benchmark enthält Daten zu den ersten zwei Betriebszyklen eines US-DWRs von Westinghouse.

Da für KMACS die Aussagekraft hinsichtlich Anlagen deutscher Bauart wesentlich ist, wurden mit der Revision 2a des Vorhabens von einem inländischen Betreiber Daten zu einem weiteren Vor-KONVOI-Reaktor erworben.

Die im Validierungsband /ZIL 18/ ausgeführten Ergebnisse basieren damit auf den Daten der zwei Vor-KONVOI-Reaktoren A und B[1] sowie dem BEAVRS-Benchmark.

Zu A sind die folgenden Daten vorhanden:

- Beladepläne von zehn aufeinanderfolgenden Beitriebszyklen
- Messdaten der Borkonzentration der letzten sechs Betriebszyklen
- Leistungsverteilung aus dem Prozessrechner für je 2-3 Zeitpunkte im Zyklus
- Daten eines kommerziellen Kernsimulators:
    - Eingabedatensätze für den Gittercode
    - Kern-Abbrandinformation in 2x2x30 (x-y-z) Nodes per Brennelement zu Beginn und Ende jedes Betriebszyklus
    - Gesamtleistungs-, Steuerelementfahr-, Eintrittstemperatur-Geschichte aus der Zyklusnachrechnung (vereinfachte, effektive Parameters)
    - Radiale Leistungsverteilung zu zwei oder drei Zeitpunkten je Betriebszyklus
    - Axiale Leistungsverteilung (radial gemittelt) je zu 7 Volllasttagen

Zu B sind die folgenden Daten vorhanden:

- Endgültige Einsatzpläne aller Betriebszyklen
- Brennelementdatenblätter für alle eingesetzten BE-Typen
- Messdaten der Borkonzentration der letzten acht Betriebszyklen
- Für die letzten elf Zyklen je ca. 10-20 Kugelmessprotokolle
- Daten eines kommerziellen Kernsimulators für die letzten 7 Zyklen:
    - Kern-Abbrandinformation in 1x1x15 (x-y-z) Nodes per Brennelement zu Zyklusbeginn

[1] Gemäß Vereinbarung mit den Eigentümern der Daten werden die Reaktoren hier anonymisiert beschrieben.

- 3D-Leistungs-, Moderatordichte- und -temperatur-, Brennstofftemperatur-Verteilung sowie kritische Borkonzentration für diverse Zeitschritte im Zyklus
- Leistungs-, und Eintrittstemperatur-Historie in vereinfachter Form (ohne Steuerstabbewegungen)

Zum BEAVRS-Benchmark sind folgende Daten vorhanden:

- Beladepläne der ersten beiden Reaktorzyklen
- zugehörige Brennelementdaten
- Detaillierter Leistungs-Zeit-Verlauf und zugehörige Steuerelementpositionen
- Gemessene kritische Borkonzentration
- Neutronenflussmessdaten zu einzelnen Zykluszeitpunkten
- Messungen der Bor- und Steuerelementwirksamkeiten sowie des isothermen Temperaturkoeffizienten je für Zyklusbeginn

## 4.2 Vergleich verschiedener Gittercodes

Im ersten Jahr der Vorhabenslaufzeit wurde der Gittercode HELIOS in KMACS integriert.

In der Vorbereitung dazu wurden zunächst die in Frage kommenden Gittercodes SCALE-Polaris, HELIOS und Serpent miteinander und mit dem bisherigen Gittercode NEWT verglichen. Dazu wurden jeweils WQ erzeugt und in folgenden Modellen in der Wirkung im Diffusionscode QUABOX/CUBBOX miteinander verglichen

1. Schachbrettanordnung MOX/UOX (mit/ohne Steuerelemente)
2. Minikern MOX/UOX mit Reflektor (mit verschiedenen Steuerstabkonfigurationen)

Neben der jeweils erzielten Übereinstimmungsgenauigkeit mit einer Serpent-Monte-Carlo-Referenz wurden die Rechenzeiten der WQ-Erzeugung mit in die Evaluierung einbezogen. Ergebnisse der Vorstudie wurden in /GRS 16/ vorgestellt.

Aus dieser Vorstudie ergab sich zwar, dass HELIOS-WQ mit geringerem Rechenzeitaufwand zu erzeugen sind, jedoch wurde im Minikernmodell eine bessere Übereinstimmung mit der Referenz mit NEWT-WQ erreicht. Die Codes Polaris und Serpent führten zu weniger guter Übereinstimmung bei der Schachbrettanordnung und benötigen deutlich mehr Rechenzeit als HELIOS.

Die Anwendung von HELIOS- und NEWT-WQ auf die Reaktorstarttests im BEAVRS-Benchmark wird in /ZIL 17a/ dokumentiert und somit die Funktionalität von HELIOS in KMACS nachgewiesen.

Während des Vorhabens erscheint SCALE 6.2 und mithin eine neue Version des NEWT-Gittercodes. Da diese Version bei vergleichbaren Ergebnissen mit der vorigen Version eine wesentlich geringere Rechenzeit und geringeren Festplattenplatz benötigt, sind auch mit dem NEWT die in Abschnitt 1.3 genannten Anforderungen an den Gittercode in KMACS 1.0 erfüllt.

Wegen der besseren Vergleichbarkeit mit Ergebnissen aus dem Vorgängervohaben, und dem größeren Erfahrungsschatz mit SCALE-NEWT wurde in den meisten Rechnungen zur Validierung von KMACS der NEWT-Gittercode verwendet.

### 4.3 Abhängigkeit der Brennstofftemperatur vom Gasspalt

Mit Beginn des Vorhabens wurde untersucht, warum die von KMACS für den Erstzyklus von Reaktor B errechnete kritische Borkonzentration (s. /ZIL 15/) die gemessene deutlich unterschätzt. Beim Vergleich mit den Ergebnissen zum anderen in /ZIL 15/ betrachteten Zyklus mit den Ergebnissen der Referenzrechnung viel auf, dass KMACS eine deutlich höhere Brennstofftemperatur ermittelte. Da eine höhere Brennstofftemperatur ebenso wie eine höhere Borkonzentration zu einer Reduktion der Reaktivität führt, war diese zu hohe Brennstofftemperatur als mögliche Ursache der Unterschätzung der Borkonzentration gefunden (Kompensationseffekt).

Für die Überschätzung der Brennstofftemperatur wurde ein zu geringer Wärmeübertragungskoeffizient zwischen Brennstofftablette und Hüllrohr als ursächlich ausgemacht. Dieser wiederum hängt von der Breite und Materialzusammensetzung des Gasspalts zwischen Tablette und Hüllrohr ab.

Im KMACS-Prototyp war die herstellungsbedingte Breite des Gasspalts des Brennstabs im frischen Zustand über die Bestrahlung konstant gehalten. Tatsächlich jedoch verlagert sich die Tablette mit Leistungsaufnahme; außerdem kommt es zu einzelnen Rissen in der Tabelle, sodass bereits zu Betriebsbeginn der effektive Abstand zwischen Tablette und Hüllrohr reduziert ist.

Zur Beschreibung dieses Vorgangs wurde das phänomenologische Modell aus /USD 97/ in vereinfachter Weise implementiert.

Damit konnte eine sehr gute Übereinstimmung zwischen gemessener und errechneter Borkonzentration für den Erstkern des Reaktors B erzielt werden, s. /GRS 16/.

## 4.4 Abhängigkeit der Neutronenflussverteilung von der räumlichen Diskretisierung des Reaktorkerns

Im Prototyp entsprachen die radialen Abmessungen eines Nodes einem Brennelementquerschnitt. Da mit den Daten zu Reaktor A aber Abbrandinformationen auf einem Gitter von 2x2 Nodes im BE-Querschnitt vorliegen, wurden die Effekte dieser verfeinerten Nodalisierung untersucht.

Die Durchsicht der Anfangsabbrände zeigte vereinzelt Abbrandunterschiede in den Vierteln eines Brennelements von über 20 % bzw. 4 MWd/kg. Eine entsprechend feiner diskretisierte KMACS-Rechnung des Erstkerns von Reaktor B führte zu Leistungsunterschieden zwischen den Quadranten eines BEs von bis zu 60 % für Eck-BE. Hingegen ist der Leistungsgradient innerhalb eines BEs der zweiten Reihe schon wesentlich geringer und ab der dritten Reihe ergibt die feinere Diskretisierung nahezu keinen Unterschied von der gröberen, s. /GRS 16/.

Wegen des deutlichen Unterschieds der Leistung innerhalb von Rand-BE wurde für die Zyklusrechnungen im Validierungsband /ZIL 18/ stets eine Nodalisierung von 2x2-Nodes im BE-Querschnitt vorgenommen.

## 4.5 Verwendung unterschiedlicher Kühlmitteltemperaturen in Führungsrohren und zwischen den Brennstäben

Im Kernsimulator, dessen Ergebnisse neben der Anlagenbeschreibung zu Reaktor A im Vorhaben erworben wurden, wird ein anderes Gittercode-Modell als im Prototyp verwendet.

Dieses verwendet, im Gegensatz zu dem im Prototyp voreingestellten Modell, unterschiedliche Kühlmitteltemperaturen in den Führungsrohren und zwischen den Brennstäben. Die Annahme dahinter ist, dass sich das Kühlmittel, welches am Kerneintritt in die Führungsrohre einfließt, mangels Durchmischung im Kern nur unwesentlich aufheizt. Da das Kühlmittel in den Führungsrohren 7 - 8 % des Kühlmittels im BE-Querschnitt ausmacht, könnte ein signifikanter Einfluss auf die Moderation, und durch Bor auch auf die Absorption, vorhanden sein.

Die Möglichkeit, die Kerneintrittstemperatur anstelle der aus dem Branching definierten Moderatortemperatur für das Kühlmittel in den Führungsrohren zu setzten, wurde implementiert. Im Eingabedatensatz kann entsprechend die Temperatur gesetzt werden. Dies betrifft nur die WQ-Erzeugung.

Für die in den Projektgesprächen /GRS 17/, /GRS 18/ vorgestellten Rechnungen zu Reaktor A wurde die Kerneintrittstemperatur in den Führungsrohren für die WQ-Erzeugung gesetzt. In Validierungsbericht /ZIL 18/ wurde jedoch, um die Vergleichbarkeit zwischen den Reaktoren A und B zu verbessern, einheitlich für alle betrachteten Zyklen die Moderatortemperatur gemäß Branching gesetzt, ohne dass sich die Ergebnisse für Reaktor A wesentlich geändert hätten.

## 4.6 Modellierung von Reflektoren

Wie im Abschnitt 3.1 bereits erläutert, wurde die im Prototyp vorgesehene Modellierung je eines Reflektormaterials pro Randmaterial als unnötig verworfen. Auch im Referenzkernsimulator für den Reaktor A wurde zyklusübergreifend dasselbe Reflektormaterial verwendet, auch wenn der BE-Typ, der für die Erzeugung der WQ benutzt wurde, nicht mehr im Kern eingeladen ist.

So wurde auch in den Rechnungen, die dem Validierungsband /ZIL 18/ zu Grunde liegen, verfahren. Ein Rand-BE-Typ wurde für die Erzeugung der Reflektor-WQ benutzt. Je ein WQ-Satz wurde für Top-, Bottom- und radialen Reflektor erzeugt. Diese Reflektormaterialien kamen dann in allen betrachteten Betriebszyklen zum Einsatz.

Möglich ist die Verwendung verschiedener Reflektormaterialien weiterhin. Speziell für radiale Reflektoren ist die Verwendung verschiedener Materialien für Reflektoren an Stirnflächen und Ecken, die sich durch unterschiedliche Anteile an Strukturmaterialien in der Kernumgebung unterscheiden, üblich.

## 4.7 Vergleich mit radialen Leistungsprofilen aus KMS bzw. Prozessrechner

In /ZIL 15/ sowie /GRS 16/ wurden Vergleiche der errechneten radialen Leistungsverteilungen mit Kugelmessprotokollen (KMS, Reaktor B) bzw. dem Momentbild des Prozessrechners (Reaktor A) vorgenommen. Dabei wurden Abweichungen von bis zu 10 % in der BE-Leistung beobachtet.

Auf weitere Validierung anhand dieser Art von Daten wurde aus den folgenden Gründen in diesem Vorhaben verzichtet.

Die Prozessrechner-Leistungsverteilung ist oft asymmetrisch, obwohl eine symmetrische Abbrandverteilung und Kernbeladung vorgegeben ist. Hintergrund ist hierbei das Verfahren der Kernüberwachung in deutschen DWR. Für den Zyklus werden mit dem Kernsimulator der Einsatzplanung Leistungsverteilungen für diskrete Abbrandpunkte im Zyklus vorausberechnet. Diese Leistungsverteilungen werden den Leistungsverteilungsdetektoren (LVD) im Kern als „Korrekturfaktoren“ (KOFA) übergeben. Jedem Detektor wird dabei ein Überwachungsbereich zugewiesen. Die Prozessrechnerausgabe ergibt sich dann als Extrapolation des aktuellen lokalen Messsignals auf diesen Überwachungsbereich. Die LVD müssen mehrfach im Zyklus kalibriert werden. Wenn seit der letzten Kalibrierung einige Tage vergangen sind, können die Signalstärken verschiedener LVD bei gleicher Leistung voneinander abweichen, was letztlich zu einer asymmetrischen Leistungsverteilung im Prozessrechner führen kann. Des Weiteren können die KOFA nicht die tatsächliche Leistungsgeschichte im Zyklus antizipieren,

sodass die den KOFA zugrundeliegende Abbrandverteilung nicht der tatsächlichen entspricht.

Da aus den vorgenannten Gründen die tatsächliche Leistungsverteilung nicht mit ausreichender Genauigkeit aus den Prozessrechnerdaten abzulesen ist, wurde im Vorhaben die radiale Leistungsverteilung nur im Vergleich mit Referenzrechnungen verifiziert.

## 4.8 Verwendung der Korrektur „b1-kritisches Spektrum“ in der WQ-Erzeugung

Im Leistungsbetrieb ist der Reaktorkern im kritischen Zustand. Einzelne Nodes des Reaktorkerns sind dabei jedoch in einem Rückwirkungszustand, der bei Betrachtung des Materials in reflektierenden Randbedingungen in der WQ-Erzeugung zu einem über- oder unterkritischen Zustand führt. Kritisch ist der Reaktorkern insgesamt daher wegen der Neutronenströme zwischen den Nodes. Diese führen auch zu einer Änderung des lokalen Neutronenspektrums.

Um die Neutronenleckage auch in der WQ-Erzeugung zu berücksichtigen, bieten viele Gittercodes die Möglichkeit der b1-Spektral-Korrektur. Mit dieser Korrektur wird ein sog. „*material buckling*“ /SCA 18/ so verwendet, dass das homogenisierte unendliche System kritisch wird. Das zugehörige Neutronenenergiespektrum wird dann zur Erzeugung der WQ eingesetzt.

Da Kritisches-Spektrum-Korrekturen in den allermeisten Kernsimulatoren eingesetzt werden, wurde die b1-Spektral-Korrektur als Standard in KMACS implementiert. Sie wurde für die Ergebnisse in /ZIL 18/ verwendet.

## 4.9 Berechnung von Reaktivitätskoeffizienten

Am Beispiel des BEAVRS-Reaktors wurde die Berechnung von Reaktivitätskoeffizienten mit KMACS erprobt.

Wie in der Messung wurden für die Berechnung der Steuerbankwirksamkeiten beim Reaktorzustand Nulllast-Heiß (hot zero power, HZP) die Bänke eingefahren und für jede zusätzlich eingefahrene Bank die kritische Borkonzentration errechnet.

Für die Berechnung des isothermen Temperaturkoeffizienten wurden, ausgehend von einem kritischen Nulllast-Reaktorzustand, Moderator- und Brennstofftemperaturen überall im Kern gleichermaßen geändert (die Moderatordichte ändert sich entsprechend) und die Änderung des Multiplitationsfaktors ermittelt.

Die Ergebnisse dieser Rechnungen im Vergleich mit den Messwerten sind in /ZIL 17a/ dargestellt.

### 4.10 Zyklusübergreifende Rechnungen

Zyklusübergreifende Rechnungen mit KMACS wurden für die Reaktoren A und B durchgeführt.

Der Anfangsabbrand des ersten jeweiligen Zyklus wurde aus der jeweiligen Referenzrechnung übertragen. Für die weiteren Zyklen wurde jeweils nur der Anfangsabbrand derjenigen BE von der Referenz übernommen, die nicht in den bisherigen KMACS-Zyklen in Einsatz waren.

Die zyklusübergreifenden Rechnungen waren der Schwerpunkt der Validierungsaktivitäten des Vorhabens. Dabei wurden verglichen:

- Kritische Borkonzentration (KMACS vs. Messung)
- Radiale Leistungsverteilung (KMACS vs. Referenzrechnung)
- Abbrandverteilung zu Zyklusende (KMACS vs. Referenzrechnung)
- Nodeweise Leistungsverteilung (KMACS vs. Referenzrechnung, Reaktor B)
- Axiale Leistungsverteilung einzelner BE (KMACS vs. KMS, Reaktor B)

Die meisten in Kapitel 0 genannten Modelländerungen basieren auf Verbesserungsbedarf, der in den zyklusübergreifenden Rechnungen erkannt wurde.

Bislang noch nicht als Modell in KMACS integriert ist die Berücksichtigung des radioaktiven Zerfalls während des Brennelementwechsel bzw. der Verweilzeit im Brennelementlagerbecken, s. Abschnitt 4.11.

## 4.11 Berücksichtigung der Abklingzeit zwischen den Zyklen

In den zyklusübergreifenden Rechnungen zu Reaktor B waren große Abweichungen in der Leistung für einige BE auffällig /GRS 17/, die vor dem betrachteten Zyklus mehrere Jahre im Lagerbecken verbrachten. Besonders deutlich ist dies für MOX-BE.

Um einzuordnen, ob diese Abweichungen tatsächlich in dem nicht berücksichtigten radioaktiven Zerfall begründet liegen, wurde dieser exemplarisch für einzelne betroffene BE in die WQ-Erzeugung einbezogen. Dazu wurde für jeden Node nach dem Zyklusanfangsabbrand die Nulllastzeit in SCALE-TRITON-NEWT einbezogen, bevor die Branch-Rechnungen durchgeführt wurden. Die in /GRS 17/ dokumentierten Ergebnisse zeigen ein Angleichen an die Referenz.

Dass diese Abklingzeit speziell für MOX-BE relevant ist, hängt auch damit zusammen, dass für diese auch das spaltbare Material selbst bei Lagerzeiten mehrerer Jahre wesentlich zerfällt (Pu-241 hat eine Halbwertszeit von 14,35 Jahren).

Da eine erneute WQ-Erzeugung abhängig vom Zyklusanfangsabbrand für jeden Node de facto zu WQ-Sätzen für jeden Node führen würde, kann dieses Verfahren wegen zu großen Ressourcenbedarfs nicht in KMACS umgesetzt werden.

Eine alternative Lösung über Korrekturen der makroskopischen WQ anhand des nodalen Inventars analog zur Methode in SIMULATE /BAH 05/ ist in Überlegung.

## 4.12 Vergleich verschiedener Kern-Thermohydraulikmodelle

Das Kern-Thermohydraulikmodell in KMACS /BOU 18/ ist ein Parallelkanalmodell, in dem jedem BE der gleiche Massenstrom aufgeprägt wird. Eintrittstemperatur und Druck geben die Randbedingungen vor, anhand derer der Thermohydraulikcode das Temperatur- und Dichteprofil entlang des Brennelements ermittelt.

Um zu überprüfen, ob Querströmungen bei Normalbetrieb eine Rolle spielen, und ob das Thermohydraulikmodell evtl. unnötig fein aufgelöst ist, wurde eine vergleichende Studie mehrerer Thermohydraulikmodelle mit dem in KMACS integrierten gekoppelten Code-System ATHLET-QUABOX/CUBBOX durchgeführt.

Die Ergebnisse wurden in /GRS 18/ vorgestellt und bilden einen Teil des Validierungsberichts /ZIL 18/. Da im Ergebnis Querströmungen vernachlässigt werden können und eine Gruppierung mehrerer Brennelemente in einem thermohydraulischen Kanal zu deutlichen Abweichungen in den Kernaustrittstemperaturen von Referenzrechnungen führen, wurde das im Prototyp enthaltene Thermohydraulik-Modell beibehalten.

### 4.13 Abhängigkeit des Massenstroms von der Kerneintrittstemperatur

Gegen Ende des Vorhabens wurden Abweichungen der kritischen Borkonzentration und der Leistungsverteilung beim Streckbetrieb genauer untersucht.

Das natürliche Ende eines DWR-Betriebszyklus $EOC_{nat}$ ist erreicht, wenn bei Volllast eine kritische Borkonzentration nahe Null vorliegt. Ein typischer Wert sind 10ppm bei $EOC_{nt.}$ Dass eine Borkonzentration von 0 nicht erreicht wird hat den Grund, dass bei geringerer Borkonzentration immer mehr Kühlmittel aus dem Primärkreis durch Deionat ersetzt werden muss, um die Borkonzentration weiter abzusenken. Das Volumenregelsystem erreicht aber nur begrenzte Einspeiseraten. Um weiterhin Kritikalität zu gewährleisten, kann wegen des negativen Moderatortemperaturkoeffizienten die mittlere Kühlmitteltemperatur abgesenkt werden. Dies wird im Streckbetrieb dadurch erreicht, dass die Reaktorleistung reduziert und, wenn die Regelung dies erlaubt, die Kerneintrittstemperatur gesenkt wird

Diese Leistungs- und Eintrittstemperaturänderungen wurden bereits in /GRS 16/, /GRS 17/ und /GRS 18/ berücksichtigt.

Mit der Änderung der Eintrittstemperatur geht aber, konstanter Volumenstrom der Hauptkühlmittelpumpen vorausgesetzt, auch eine Änderung des Massenstroms durch den Kern einher, die erst in /ZIL 18/ berücksichtigt wurde, und zu einer besseren Übereinstimmung mit der gemessenen Borkonzentration führt.

# 5 Diskussion und Ausblick

Mit KMACS 1.0 verfügt die GRS über einen Kernsimulator, der die wesentlichen für Einsatzplanung und Zyklusnachrechnung von DWR üblichen Simulationen ermöglicht.

Errechnet werden können:

- Der Verlauf der kritischen Borkonzentration
- Die Leistungsverteilung im Zyklusverlauf
- Reaktivitätskoeffizienten
    - Steuerelementwirksamkeiten
    - Borwirksamkeit
    - Moderatortemperaturkoeffizient
    - Brennstofftemperaturkoeffizient

Die Rechenergebnisse, die im Validierungsband /ZIL 18/ gesammelt sind, fallen zum überwiegenden Teil in den Rahmen dessen, was die American Nuclear Society als zulässige Abweichung zwischen Reaktorrechnung und Messung an der Anlage definiert /ANS 11/.

Verbesserungspotential durch eine Berücksichtigung des radioaktiven Zerfalls zwischen den Zyklen und Validierung bei Einsatz des HELIOS-Gittercodes und des DYN3D-Neutronendiffusionscodes wurde benannt.

KMACS bietet über das übliche Einsatzfeld eines Kernsimulators hinaus vielfältige Anwendungsmöglichkeiten:

- Das Modell des Reaktorkerns kann für Transienten und Störfallrechnungen herangezogen werden
- Das Kernmodell kann (mit Modifikationen) für die Untersuchung aktueller Phänomene in DWR herangezogen werden, z. B.
    - Brennelement-Verbiegungen
    - Neutronenflussschwankungen
- Mit VENTINA als Abbrandcode kann das Nuklidinventar und die Zerfallswärme aller BE im Kern in nodaler Auflösung verfolgt werden

## Literaturverzeichnis


/ANS 11/ American Nuclear Society: Reload startup physics tests for pressurized water reactors, ANSI/ANS-19.6.1-2011.

/BEL 14/ Bell, I., Wronski, J., Quoilin, S. und Lemort, V.: Pure and Pseudo-pure Fluid Thermophysical Property Evaluation and the Open-Source Thermophysical Property Library CoolProp, Industrial & Engineering Chemistry Research, vol. 53, pp. 2498–2508, 2014.

/BAH 05/ Bahadir, T., Lindahl, S. und Palmtag, S.: SIMULATE-4 Multigroup Nodal Code with Microscopic Depletion Model, in Mathematics and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications, (Avignon, France), 2005.

/BOU 18/ Bousquet, J. und Zilly, M.: KMACS User Manual, Version 1.0, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH, Garching, 2018.

/GOD 17/ Godfrey, A., Collins, B., Gentry, C., Stimpson, S., Ritchie, J.: Watts Bar Unit 2 Startup Results with VERA, CASL-U-2017-1306-000, U.S. Department of Energy, 31. März 2017.

/GRU 05/ Grundmann, U., Rohde, U., Mittag, S. und Kliem, S.: DYN3D version 3.2, code for calculation of transients in light water reactors (LWR) with hexagonal or quadratic fuel elements - description of models and methods, Forschungszentrum Dresden-Rossendorf. Dresden, 2005.

/GRS 16/ GRS-BMUB-BfS, Projektgespräch zum Vorhaben 3615R01344. Köln, 6. Juni 2016.

/GRS 17/ GRS-BMUB-BfE, Projektgespräch zum Vorhaben 3615R01344. Köln, 28. März 2017.

/GRS 18/ GRS-BMUB-BfE, Projektgespräch zum Vorhaben 4715R01344, Bonn, 14. Februar 2018.

/HOR 13/ Horelik, N., Herman, B., Forget, B. und Smith, K.: Benchmark for evaluation and validation of reactor simulations (BEAVRS), v1.0.1, in M&C 2013, Sun Valley, Idaho, USA, 5.-9. Mai 2013.

/JUN 13/ Jung, Y., Shim, C., Lim, C. und Joo, H.: Practical numerical reactor employing direct whole core neutron transport and subchannel thermal/hydraulic solvers, Annals of Nuclear Energy, 62, pp. 357–374, 2013.

/KÜN 17/ Küntzel, M.: KMACS Developer Manual, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH. Köln, 2017.

/LEP 15/ Leppänen, J., et al.: The Serpent Monte Carlo code: Status, development and applications in 2013, Annals of Nuclear Energy, 82, pp. 142-150, 2015.

/PYT 18/ Python Software Foundation: Python 2.7.15 documentation, 2018. https://docs.python.org/2/.

/SCA 18/ SCALE 6.2.3: A Comprehensive Modeling and Simulation Suite for Nuclear Safety Analysis and Design, ORNL/TM-2005/39, Oak Ridge National Laboratory, USA, März 2018.

/SMG 17/ Software Management Group: Maßnahmen zur Qualitätssicherung von Computerprogrammen der GRS, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH. Köln, 2017.

/STU 12/ Studsvik Scandpower: The HELIOS Lattice Physics Code, Version 1.12, 2012.

/USD 97/ U.S. Department of Commerce, FRAPCON-3: Modifications to fuel rod material properties and performance models for high-burnup application, National Technical Information Service, Vol. 1, 1997.

/YAM09/ Ben-Kiki, O., Evans, C., Net, I.:YAML Ain't Markup Language (YAML) version 1.2, http://yaml.org/spec/1.2.

/ZIL15/ Zilly, M., Aures, A., Hannstein, V., Küntzel, M., Pasichnyk, I., Périn, Y., Seubert, A., Velkov, K.: Erstellung und Qualifizierung eines nuklearen Kernsimulators für die Sicherheitsbewertung aktueller Reaktorkernbeladungen, GRS-A-3805, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH: Köln, 2018.

/ZIL17/ Zilly, M.: QS-Plan für den GRS-Kernsimulator KMACS, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH. Köln, 2017.

/ZIL17a/ Zilly, M., Bousquet, J, Velkov, K., Ban, Y. und Joo, H.: Analyzing the BEAVRS HZP case with KMACS using 2-group data from SCALE-NEWT, HELIOS and nTRACER, International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering, Jeju, Korea, April 16-20, 2017.

/ZIL18/ Zilly, M. und Périn, Y. KMACS Validation Report, Version 1.0, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH: Köln, 2018.

**Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH**

Schwertnergasse 1
**50667 Köln**
Telefon +49 221 2068-0
Telefax +49 221 2068-888

Boltzmannstraße 14
**85748 Garching b. München**
Telefon +49 89 32004-0
Telefax +49 89 32004-300

Kurfürstendamm 200
**10719 Berlin**
Telefon +49 30 88589-0
Telefax +49 30 88589-111

Theodor-Heuss-Straße 4
**38122 Braunschweig**
Telefon +49 531 8012-0
Telefax +49 531 8012-200

www.grs.de

**ISBN 978-3-946607-84-7**