TRANSFORMATION DER GEBÄUDEENERGIESIMULATION NANDRAD MIT VARIABLEM ZEITSCHRITTLÖSER IN EINE FMU FÜR CO-SIMULATION

A. Nicolai[1] , A. Paepcke[1]
[1]Institut für Bauklimatik, TU Dresden, Dresden, Germany
andreas.nicolai@tu-dresden.de

KURZFASSUNG

Moderne Gebäudesimulationsmodelle sind mit variabler Zeitschrittsteuerung implementiert, wobei die Integrationsverfahren lokale Fehlerschätzer zur Anpassung der Zeitschritte verwenden. Wenn solche Berechnungsverfahren über eine Co-Simulationsschnittstelle gekoppelt werden, wobei der Simulationsmaster ebenso variable Kommunikationsschrittweiten verwenden kann, wird die Verwaltung des modellseitigen Integrator- und Modellzustandes aufwändig. Dieser Artikel beschreibt die Transformation des Gebäudenergiesimulationsmodells NANDRAD in eine Functional Mockup Unit, welche den FMI Schnittstellenstandard 2.0 unterstützt. Wir diskutieren verschiedene Aspekte der Softwareumsetzung mit dem Schwerpunkt auf robuster und effizienter Implementierung. Details wie die Behandlung der unterschiedlichen internen Integrationsschrittweiten und Kommunikationsintervalle, Rücksetzung des Solverzustands, Iteration über Kommunikationsintervalle und die Behandlung der Simulationsausgaben werden am Beispiel von NANDRAD diskutiert.

ABSTRACT

Modern building simulation models are implemented with variable time step solvers that use local error tests for time step adjustment. When interfacing such solvers via co-simulation interfaces through a master that may itself use variable communication step sizes, the management of the internal solver and model state becomes non-trivial. The article describes the transformation of the stand-alone building energy simulation model NANDRAD into a Functional Mockup Unit that implements the Co-Simulation interface standard 2.0. We discuss the various aspects related to robust and efficient implementations, such as treatment of differences between internal integration steps and communication intervals, restoration of the solver states, iteration over communication intervals, and handling of simulation outputs.

EINLEITUNG

NANDRAD ist eine Gebäudeenergiesimulation, welche ähnlich wie z.B. EnergyPlus oder TRNSYS ein Multizonenmodell berechnet. Es zeichnet sich dadurch aus, dass es stets detaillierte und raumaufgelöste Wand-, Fußboden- und Deckenkonstruktionen berücksichtigt und somit Wärmespeichereffekte viel genauer abbilden kann, als es mit CTF-Methoden in EnergyPlus/TRNSYS möglich ist. Für die Simulation ergibt sich aus der Raumdiskterisierung der partiellen Differenzialgleichungen eine größere Anzahl von Unbekannten. Auch stellen Modellfeatures wie Flächenheizungen einschließlich der Regelung nach der operativen Temperatur besondere Anforderungen an das Lösungsverfahren (Paepcke and Nicolai, 2014). Eine hohe Berechnungsgeschwindigkeit wird unter anderem dadurch erreicht, dass eine variable Zeitschrittsteuerung sowie ein modifiziertes Krylow-Newton-Verfahren eingesetzt werden. Auch sind verschiedene Detailoptimierungen, z.B. bei der Ausgabebehandlung für die Simulationsperformance relevant und müssen im Rahmen der Simulationskopplung beachtet werden. Die variable Zeitschrittanpassung ist auch elementar für eine robuste Anbindung von Anlagen- und Regelungstechnik, welche zeitweise aus Gründen der numerischen Stabilität sehr kleine Zeitschritte verlangt.

Simulationskopplung ist eine Möglichkeit, die Kompetenzen verschiedener Modellentwickler und deren Lösungsverfahren zu kombinieren, und dadurch eine aussagefähigere und umfängliche Modellierung des Gebäudes, der Anlage, der Umwelt und Nutzer zu erreichen. Neben spezifischen Kopplungsmethoden für Raum- und Bauteilsimulation, z.B. eines Luftströmungsnetzwerkes und der Bauteilsimulation (Nicolai et al., 2007; und Pazold und Antretter, 2013) ergeben sich durch Standardisierung von Kopplungsschnittstellen (Modelica Association Project „FMI“, 2014) viel allgemeinere Möglichkeiten. Im FMI Standard wird die technische Seite des Informationsaustauschs geregelt, wobei zwischen ModelExchange und CoSimulation unterschieden wird. Im letzteren Kopplungsmodus bringt eine FMU (Functional Mockup Unit, d.h. ein gekapseltes Simulationsmodell) einen eigenen Zeitintegrationsalgorithmus mit und kommuniziert zu bestimmten Zeitpunkten mit anderen FMUs. Die Kontrolle der FMUs und die Kommunikation wird durch einen Simulationsmaster gesteuert. Dafür lässt der Master die einzelnen FMUs immer nur eine bestimmte Zeitspanne rechnen. Die Länge des Rechenschritts wird vom Master bestimmt,

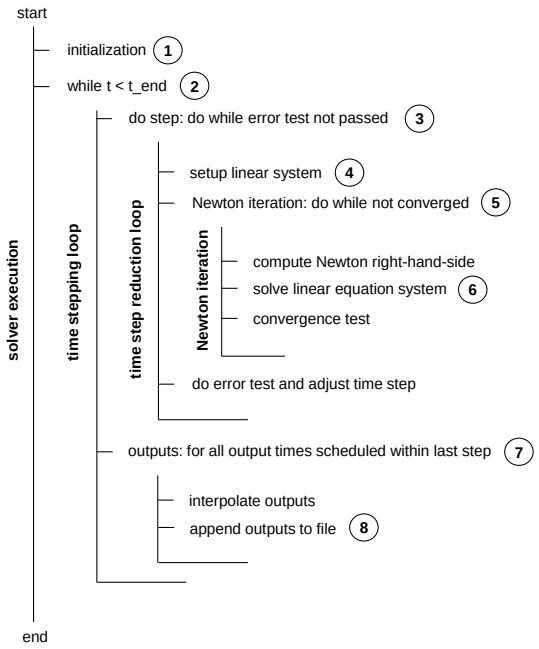

Abbildung 1: Ablaufschema des NANDRAD Solverkerns

wobei die meisten aktuell verfügbaren Simulationsmaster nur konstante Schrittweiten unterstützen.

Zum Verständnis der Funktionsweise der NANDRAD-FMU innerhalb der Co-Simulation ist es hilfreich, die Funktionsweise des NANDRAD-Solvers bei alleinstehender Ausführung zu kennen.

FUNKTIONSWEISE DES NANDRAD-SOLVERS

Der NANDRAD Rechenkern verwendet ein BDF-Verfahren zur Zeitintegration unter Verwendung der SUNDIALS Bibliothek (Hindmarsh et al., 2005). Die Implementierung besteht, wie in Abbildung 1 gezeigt, im Wesentlichen aus vier geschachtelten Schleifen mit den folgenden wesentlichen Bausteinen:

1. Initialisierung des Berechnungsmodells, muss auch bei FMU Ausführung nur einmal gemacht werden
2. Hauptschleife, führt Integrationsschritte aus, bis Simulationsende erreicht ist
3. Fehlerkontrollschleife, wird solange durchlaufen, bis Fehlerschranke unterschritten wird (in jedem invaliden Schleifendurchlauf wird der Zeitschritt reduziert)
4. Berechnung der Jacobi-Matrix und des Vorkonditionierers, allerdings nur bei Konvergenzfehlern oder starken Zeitschrittänderungen; zumeist bleibt die Jacobi-Matrix für mehrere Schritte bestehen
5. Newton-Iterationsschleife, wird solange durchlaufen, bis das Verfahren konvergiert oder abbricht
6. Lösen des linearen Gleichungssystems, mittels vorkonditioniertem Krylow-Unterraumverfahren
7. Alle im letzten Zeitschritt geplanten Ausgaben bearbeiten (Integrationsschritte sind nicht an Ausgabezeitschritte gekoppelt)
8. Anhängen der berechneten Ergebnisse an die Ausgabedateien (Ergebnisdaten werden nicht im Speicher gehalten)

Die Implementierung der physikalischen Gleichungen ist innerhalb des NANDRAD Rechenkerns in einem zustandsbehafteten Modellobjekt gekapselt. Dieses Modellobjekt hält Ergebnisgrößen seiner implementierten Modellgleichungen. Bei der Auswertung von Gleichungen wird auf vorab berechnete und abgelegte Ergebnisse direkt zugegriffen. Nach Auswertung aller Gleichungen sind alle Ergebnisgrößen zueinander und zu den Eingangsgrößen stimmig. Der Zustand dieses NANDRAD-Modellobjekts wird von außen durch Setzen eines Zeitpunkts und/oder der primären Lösungsvariablen, bzw. FMU-Importvariablen geändert (Details dazu in Paepcke et al., 2014). Durch diese Implementierung konnte das NANDRAD-Modell nahezu direkt in eine FMU mit ModelExchange Schnittstelle eingebettet werden, welche jedoch außerhalb des Fokus dieses Artikels liegt.

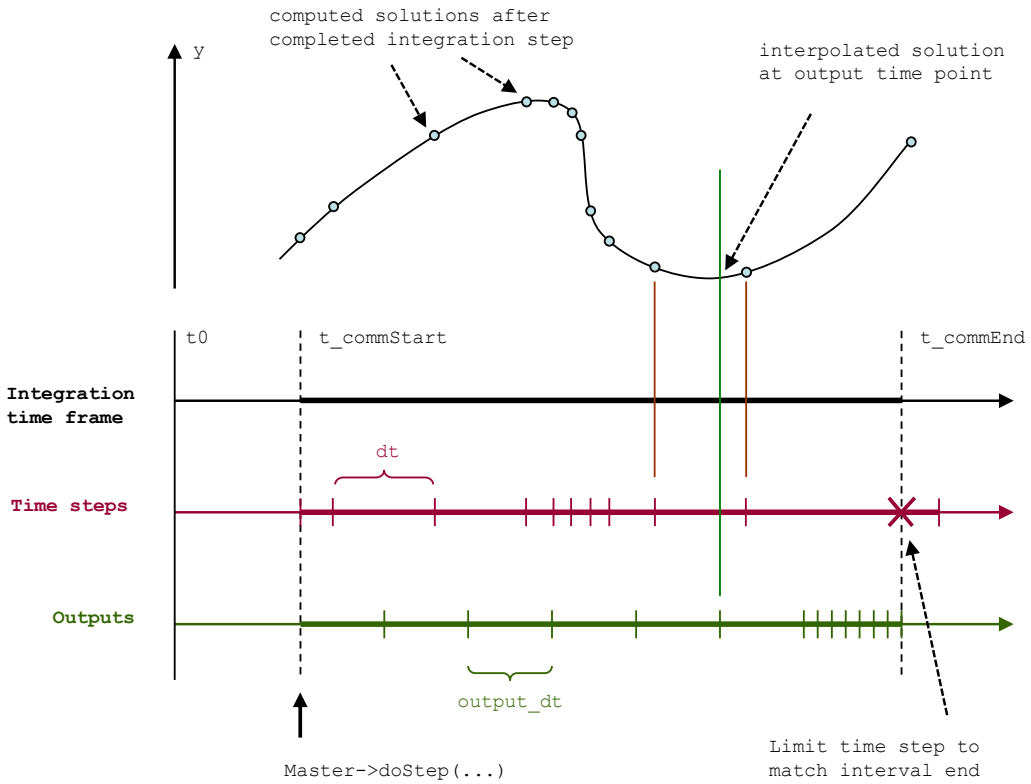

Abbildung 2: Integrationsverlauf innerhalb eines Kommunikationsintervalls

ZEITINTEGRATION INNERHALB DER CO-SIMULATION

Sobald der NANDRAD Rechenkern als Co-Simulations-FMU fungiert, wird die Hauptschleife (2) dahingehend modifiziert, dass die Integration nicht mehr bis zum Ende durchläuft, sondern nur bis zu einem vom Master gegebenen Endzeitpunkt des Kommunikationsintervalls.

Dieses ist in Abbildung 2 verdeutlicht, welches im oberen Teil beispielhaft den Verlauf einer Lösungsgröße zeigt. Nach der Initialisierung der FMU ruft der Master nach Austausch aller Eingangs- und Ausgangsgrößen die Funktion `doStep()` auf und fordert die FMU zur Berechnung der Lösung innerhalb des Kommunikationsintervalls auf. Ein Integrationsverfahren mit variabler Zeitschrittsteuerung wird nun Zeitschrittlängen (`dt`) in Abhängigkeit von Verlauf und Gradient der Berechnungsvariablen wählen. Nach jedem abgeschlossenen Zeitschritt hat der Integrator eine Lösung berechnet (Punkte im oberen Diagramm von Abb. 2). Nun werden alle im gerade abgeschlossenen Intervall geplanten Ausgaben berechnet, wobei die Länge der Ausgabeintervalle (`output_dt`) variieren kann. Für jeden Ausgabezeitpunkt werden die Ergebnisse interpoliert. Falls mehrere Ausgaben innerhalb eines Zeitschrittes berechnet werden sollen, werden nacheinander alle Ausgaben interpoliert und in Dateien geschrieben.

Wichtig ist hierbei zu beachten, dass nach der Interpolation der Zustand des physikalischen Modells auf den *Zeitpunkt der Ausgabe* aktualisiert wurde, und nicht mehr dem Zustand *am Ende des Zeitschritts* entspricht. Dieses ist eine subtile Fehlerquelle und bei der Kommunikation der Ergebnisgrößen zum Master zu beachten (siehe unten).

Die Zeitschrittwahl eines Zeitintegrationsverfahrens mit variablen Zeitschritten, z.B. beim CVODE mit BDF Verfahren im NANDRAD-Solver, hängt vom Verlauf der Lösungsvariable und Konvergenzverhalten ab und wird innerhalb der Integrationsfunktion selbst angepasst. Beim Aufruf des Integrators muss daher zusätzlich sichergestellt werden, dass der letzte Integrationsschritt exakt auf das Ende des Kommunikationsintervalls fällt und damit die gewählte Zeitschrittlänge entsprechend reduziert wird (Abbildung 2).

Ist das Kommunikationsintervall beendet, geht die Kontrolle an den Master zurück, welcher dann Ergebnisgrößen aus der FMU abrufen kann.

CO-SIMULATIONS-MASTER OHNE ITERATION

Bei einem nicht-iterierendem Master-Algorithmus werden Kommunikationsintervalle strikt chronologisch abgearbeitet, d.h. die Hauptschleife (2) in Abbildung 1 wird in Kommunikationsintervalle zerlegt, zwischen denen die Kontrolle an den Master abgegeben wird. Für NANDRAD bedeutet die Unterstüt-

zung eines solchen Masteralgorithmus lediglich zwei grundlegende Anpassungen:

a. Begrenzung der Integrationszeitschrittlänge im letzten Integrationsschritt, sodass Integrator exakt das Ende des Kommunikationszeitschritts trifft, und

b. aktualisieren des Modellzustands auf das Ende des Kommunikationsintervals, falls im letzten Zeitschritt Ausgaben geschrieben wurden.

Der letzte Punkt ist daher notwendig, da bei NANDRAD die Funktionen zum Abruf der FMU-Ergebnisgrößen aus Performancegründen direkt auf den aktuellen Modellzustand (Speicheradressen) zugreifen. Bei nutzerdefinierten Ausgabeintervallen und vom Master gewählten Kommunikationsintervallen kann *nicht* davon ausgegangen werden, dass die Endzeitpunkte von Integrationsschritten, Ausgabeintervallen und Kommunikationsintervallen zusammenfallen.

CO-SIMULATIONS-MASTER MIT ITERATION

Wird eine Co-Simulation ohne Möglichkeit des Zurücksetzens einer FMU durchgeführt, sind bei nicht-trivialen FMUs und ungünstig gewählten Kommunikationsintervallen Stabilitätsprobleme wahrscheinlich (Paepcke et a., 2016). Robuste Master-Implementierungen können daher von einzelnen FMUs den aktuellen Zustand speichern und zurücksetzen. Aus Sicht einer FMU bedeutet dieses im Wesentlichen die Implementierung der Schnittstellen: `fmi2GetFMUstate()` und `fmi2SetFMUstate()`.

Sobald eine FMU diese Funktionalität unterstützt, kann der Master das gleiche Kommunikationsintervall mehrfach durchlaufen, oder auch die Intervalllänge reduzieren. Aus Abbildung 2 wird ersichtlich, dass in diesem Fall das Ausgabeverhalten überarbeitet werden muss. Bei alleinstehender Ausführung von NANDRAD werden Ergebnisdaten, sobald berechnet, an Ausgabedateien angehangen. Die Haltung aller Ergebnisgrößen im Hauptspeicher ist bei komplexen Gebäudegeometrien in der Regel nicht möglich. Das kontinuierliche Erweitern der Ergebnisdateien ermöglicht zudem eine Verfolgung des Simulationsfortschritts durch externe PostProzessing-Programme. Dafür wird in NANDRAD ein spezielles Datenkontainerformat (Vogelsang & Nicolai, 2011) verwendet.

Dieses Verfahren ist bei einem mehrfachen Durchlaufen eines Kommunikationsintervals nicht anwendbar, da so mehrere Ausgaben für den gleichen Ausgabezeitpunkt geschrieben werden, und die Ergebnisdateien keine chronologisch ansteigenden Ausgabezeitpunkte enthalten. Es sind zwei Möglichkeiten der Implementierung zu diskutieren:

Variante 1: Bisheriger Ergebnisdateien im Binärformat werden selektiv überschreiben. Wird die FMU vom Master aufgefordert, ein Kommunikationsintervall zu wiederholen, wird der Ausgabestream (Dateizeiger) einer Ausgabedatei auf die Position verschoben, die nach Abschluss des *letzten* Kommunikationsintervalls erreicht war. Bei nun folgenden Ausgaben werden existierende Ergebnisse in der Ausgabedatei einfach überschrieben. Vorteilhaft ist hierbei, dass der für die Ausgaben verantwortliche Quelltext nicht verändert werden muss und lediglich das Speichern/Zurücksetzen der Streampositionen/Dateizeiger implementiert wird. Unter Linux/Unix Betriebssystemen ist dieses Verfahren auch sehr performant, da das wiederholte Schreiben gleicher Dateibereiche im Wesentlichen im Hauptspeicher abläuft und erst nach einiger Zeit die Speicherblöcke auf den physischen Datenträger geschrieben wird. Unter Windows (getestet Windows 7/8.1) ist dieses Verfahren je nach Länge der Daten deutlich langsamer und kann daher nicht generell empfohlen werden.

Variante 2: Alle in einem Ausgabeintervall zu schreibenden Ergebnisdaten werden zunächst im Speicher gesammelt. Wird ein Kommunikationsintervall wiederholt, werden die Datenstrukturen geleert und im Verlauf der Integration erneut gefüllt. Wird ein neues Kommunikationsintervall begonnen, z.B am `noSetFMUStatePriorToCurrentPoint` Argument von `doStep()` erkennbar, werden bislang die gesammelten Daten an die Dateien angehängt.

Dieses Verfahren bringt einen Mehraufwand bei der Implementierung mit sich und könnte unter Umständen je nach Modellgröße und Ausgabenanzahl im Kommunikationsintervall eine erhebliche Vergrößerung des Hauptspeicherbedarfs bedingen. Letzteres war in bisherigen Tests jedoch unerheblich und daher ist Variante 2 in NANDRAD implementiert. Außerdem ist diese Implementierung auch bei Ausgabedateien im Textformat anwendbar.

In realistischen Simulationsszenarien wird die Kommunikationsschrittweite häufig durch Genauigkeits-/Stabilitätsforderungen des Master begrenzt. Dadurch ist die Anzahl der Ausgabezeitpunkte je Kommunikationsintervall klein und auch der Performanceunterschied zwischen beiden Varianten ist zu vernachlässigen. Die Wahl der Implementierungsvariante sollte daher anhand des Aufwands der individuellen Quelltextanpassung entschieden werden.

SPEICHERN UND ZURÜCKSETZEN DES FMU-ZUSTANDS

Jeder Masteralgorithmus mit Iterationsalgorithmus muss die gekoppelten FMUs das Kommunikationsintervall mehrfach durchlaufen lassen. Dafür wird eine FMU zunächst aufgefordert, den internen Zustand zu speichern. Vor erneutem Ablauf eines

```
// Define für das Speichern einer Variable in den Speicherbereich
#define SERIALIZE(type, storageDataPtr, value)\
{\
  *(type *)(storageDataPtr) = (value);\
  (storageDataPtr) = (char *)(storageDataPtr) + sizeof(type);\
}

// Define für das Laden einer Variable aus dem Speicherbereich
#define DESERIALIZE(type, storageDataPtr, value)\
{\
  (value) = *(type *)(storageDataPtr);\
  (storageDataPtr) = (char *)(storageDataPtr) + sizeof(type);\
}

// Macro für das wahlweise Ausführen einer Operation:
// 0 - Serialisierung, 1 - Deserialisierung, 2 - Größenberechnung
#define SERIALIZE_OP(op, type, storageDataPtr, value, mem_size)\
  switch (op) {\
  case 0:\
    SERIALIZE(type, storageDataPtr, value); break;\
  case 1:\
    DESERIALIZE(type, storageDataPtr, value); break;\
  case 2:\
    mem_size += sizeof(type); break;\
  }

// ...

// Serialisierungscode, je nach op=0,1,2 wird eine der Operationen ausgeführt
// storageData ist ein void-Zeiger, memSize vom type size_t und initialisiert mit 0
SERIALIZE_OP(op, double, storageData, x, memSize);    // serialisiere "double x"
SERIALIZE_OP(op, long int, storageData, i, memSize);  // serialisiere "long int i"
```

Abbildung 3: Macro-basierte Implementierung der Serialisierungsfunktionalität

Kommunikationsintervalls fordert der Master von der FMU, den gespeicherten Zustand zurückzusetzen. Je nach Masteralgorithmus kann eine FMU vom Master sogar aufgefordert werden, mehrere Zustände zu speichern.

Für ein Komplexmodell wie NANDRAD wird der aktuelle Zustand der FMU durch folgende Komponenten definiert:

1. Zustand des Integrators (Zeitpunkt, Lösungsvariablen und Historie, Steuerungsvariablen)
2. Zustand des Gleichungssystemlösers, im Falle des GMRES nur Steuerungsvariablen
3. Zustand der Jacobi-Matrix (wird nur gelegentlich aktualisiert)
4. Zustand des Vorkonditionierers (Teilmenge der Jacobi-Matrix und im Fall eines ILU Vorkonditionierers die faktorisierte Form des Vorkonditionierers)
5. Integrale Modellzustände (integrale Ausgaben, Zustände von Hystereseschleifen, etc.)

Der Zustand des physikalischen Modells muss nicht gespeichert werden, da dieser aus den im Integrator gespeicherten primären Lösungsvariablen direkt neu berechnet wird (im Predictor-Schritt des Integrators).

In dem NANDRAD zugrunde liegenden Solverframework sind die oben aufgeführten Komponenten in verschiedenen, dynamisch allozierten Speicherbereichen abgelegt. Für das Sichern des FMU-Zustands müssen diese nun in einen anderen Speicherbereich kopiert werden, wobei zwei Varianten denkbar sind:

1. 1-zu-1 Kopie der originalen Speicherstruktur in eine neue Speicherstruktur
2. Serialisierung der originalen Speicherstruktur in einen linearen, zusammenhängenden Speicherbereich

Bei der ersten Variante muss zunächst eine identische Speicherstruktur (bestehend aus verknüpften Objekten, z.B. struct oder class-Objekte, mit Zeigern auf andere Objekte) erstellt werden. Dann müssen selektiv Daten aus einer Speicherstruktur in die andere kopiert werden, wobei Verknüpfungsinformationen (d.h. Zeiger und Speicheradressen) ausgespart werden dürfen. Bereits das Erstellen der identischen Speicherstruktur ist quelltextseitig kompliziert, da diese Funktionalität an die zum Teil zeitaufwändige Initialisierungsfunktionalität des Modells gekoppelt ist. Außerdem entsteht so in der Speicherkopie eine

ebenso zerklüftete Speicherstruktur wie im Original, welches die Kopieroperation verlangsamen könnte.

In Variante 2 muss zunächst die Größe des linearen Speicherbereichs festgestellt werden, darauffolgend wird dieser Speicherbereich als Block reserviert und anschließend werden selektiv alle zu sichernden Daten übertragen. Beim Zurücksetzen der FMU müssen diese Daten in exakt der selben Reihenfolge wieder ausgelesen und in die originale Speicherstruktur übertragen werden. Dabei ist es hilfreich, eine Makro-basierte Implementierung zu verwenden, wobei Datentypen und Reihenfolge *nur einmal kodiert werden*, und je nach Makro-Parameter für alle drei Anwendungsbereiche (Größenbestimmung, Serialisierung und Deserialisierung) der dazu passende Quelltext erstellt wird. Abbildung 3 skizziert diese Implementierungsform. Der Quelltext, der die eigentlichen Daten serialisiert, im Beispiel die Variablen x und i, ist für alle drei Operationen identisch und muss somit nur einmal geschrieben und auch nur an einer Stelle gepflegt werden. Kerngedanke dieser Implementierung ist das *Weiterschieben* des Zeigers innerhalb des linearen Speicherbereichs um die Länge des transferierten Datentyps nach jeder Operation. Größere lineare Speicherbereiche, z.B. Vektoren mit Lösungsvariablen, lassen sich über effizientere Kopieroperationen (z.B. `memcpy`) übertragen, wonach wiederum der Zeiger um die kopierten Speicherlänge inkrementiert wird.

Dieses Implementierungskonzept wurde für alle oben erwähnten Komponenten umgesetzt. Für die SUNDIALS-Bibliothek (CVODE Integrator) wurde ein Patch erstellt, der diese Funktionalität nachrüstet (vom Autor als OpenSource-Quelltext erhältlich).

Ein weiterer Vorteil der 2. Serialisierungsvariante ist die Möglichkeit, ohne weiteren Aufwand die FMU-Funktionalität `fmi2SerializeFMUstate()` und `fmi2DeSerializeFMUstate()` anzubieten. Diese Funktionen erlauben dem Master, den internen Zustand einer FMU komplett zu holen und z.B. in eine Datei zu speichern. Bei einem Neustart des Masters kann dieser nun die Zustände aller FMUs wiederherstellen und die Integration fortsetzen.

NANDRAD kann mittels dieser Funktionalität auch für Optimierungsrechnungen eingesetzt werden, bei dem z.B. der Master unter Verwendung von modifizierten FMU-Parametrisierungen mehrfach neu gestartet wird und einen Teil der Simulation zum Zweck der Parameteroptimierung mehrfach durchläuft.

ERGEBNISDATEIBEHANDLUNG UND INITIALISIERUNG VON KOMPLEXEN SIMULATIONS-FMUS

Gebäudesimulationsprogramme generieren zur Auswertung der Simulation üblicherweise zahlreiche Ergebnisse und Dateien, sowie zusätzlich Protokolldateien. Bei NANDRAD werden diese Dateien in einer Verzeichnisstruktur abgelegt, welche in der Initialisierungsphase angelegt wird. Das Basisverzeichnis ergibt sich entweder aus dem Namen der Projektdatei oder kann vom Anwender via Kommandozeilenargument gesetzt werden. Dieses ist hilfreich, wenn mehrere Simulationen parallel ausgeführt werden, zum Beispiel zum Zweck einer Variantenstudie.

Bei alleinstehender Ausführung von NANDRAD folgt die Initialisierung dem Schema in Abbildung 4.

1. Start des Programms
2. Lesen der Kommandozeilenparameter
   (Basisverzeichnis für Ausgaben ist nun bekannt)
3. Erstellen der Verzeichnisstruktur und Logdatei
4. Lesen der Projektdatei
   (Fehler/Warnungen werden in Logdatei geschrieben)
5. Initialisieren des Berechnungsmodells
   (Liste verfügbarer Ergebnisgrößen ist erstellt)
6. Erstellen der möglichen Ergebnisdateien
   und Schreiben der Dateiheader
   (Dateien sind am Start des Datenblocks positioniert)
7. Schreiben der Anfangsbedingungen
8. Simulationsstart

Abbildung 4: Standard-Initialisierung

Nach Starten einer so initialisierten Simulation kann das vorab beschriebene Prozedere der Ausgabedateninterpolation und Anhängen an Ergebnisdateien verwendet werden.

Ausgabebasisverzeichnis

Wenn NANDRAD als FMU gestartet wird, ist zunächst die Frage nach dem zu verwendenden Ausgabeverzeichnis zu klären. Es ist z.B. möglich, dass das gleiche Gebäudemodell mehrfach in einem Masterszenario vorkommt, z.B. bei einer Quartierssimulation. In diesem Fall werden von der gleichen FMU mehrere Instanzen erstellt, deren Ergebnisdateien sich natürlich nicht überschreiben dürfen. Daher kann das Ausgabeverzeichnis beim Erstellen der FMU nicht festgelegt werden.

Die FMU Schnittstelle für das Instantiieren einer FMU `fmi2Instantiate()` liefert zwar den Pfad für die FMU-internen Resourcen, jedoch keinen Ergebnispfad. Dieser muss daher während des Initialisierungsprozederes als Parameter vom Master gesetzt werden. Für die am IBK entwickelten FMUs wird dafür stets ein Parameter `ResultsRootDir` vom Typ String und `variability="fixed"` definiert.

Schritte 3 bis 5 der Initialisierung (Abbildung 4) können erst ausgeführt werden, sobald `ResultsRootDir` gesetzt ist. Ein Master kann jedoch laut Standard einen solchen Parameter mehrfach setzen, zum Beispiel zunächst einen Standardwert und dann einen anwenderspezifischen Parameter. Daher wird erst in der Funktion

`fmi2EnterInitializationMode()` die eigentliche Modellinitialisierung durchgeführt. Zu diesem Zeitpunkt ist jedoch noch nicht bekannt, ob die NANDRAD Simulation von Anfang an, d.h. mit leeren Ergebnisdateien beginnen soll, oder ob der Master einen vorherigen Simulationslauf an einer bestimmten Stelle erneut startet. Daher kann dieser Teil der Initialisierung (Schritte 6 und 7 in Abbildung 4) noch nicht durchgeführt werden.

Ergebnisdateien

Für die NANDRAD-FMU gibt es nach Abschluss des „Initialization-Mode“ zwei Möglichkeiten:

1. Es folgt nach Austausch von Ein- und Ausgangsgrößen ein Aufruf der Funktion `doStep()`, oder
2. der Master startet die FMU nicht am Beginn der Simulation (Masterzeit = 0), sondern setzt die FMU mittels `fmi2DeSerializeFMUstate()` auf einen anderen Zustand/Zeitpunkt.

Im ersten Fall werden die originalen Initialisierungsschritte 6 und 7 nachgeholt und die Simulation dann wie bisher besprochen durchgeführt.

Bei jedem Deserialisierungsaufruf, der auch während einer laufenden Simulation jederzeit vom Master durchgeführt werden kann, muss die NANDRAD-FMU besondere Regeln bei der Ausgabebehandlung einhalten:

a) Sind noch keine Ausgabedateien vorhanden (z.B. wenn sie manuell gelöscht wurden), werden neue Ausgaben erstellt und vom Rücksetzzeitpunkt an fortgeschrieben.

b) Sind Ausgabedateien bereits vorhanden und enthalten Daten, die dem Rücksetzzeitpunkt zeitlich nachgelagert sind, werden die Ausgabedateien entsprechend gekürzt und danach fortgeschrieben. Beispielsweise enthält eine Ausgabedatei die Raumlufttemperaturen der ersten 2 Monate; dann wird die Simulation auf das Ende des 1. Monats zurückgesetzt und damit alle Ausgaben des 2. Monats verworfen.

c) Sind Ausgabedateien mit weniger als den geplanten Ausgaben vorhanden, werden diese einfach geöffnet und fortgeschrieben, auch wenn dann eventuell eine Lücke in den Ergebnisgrößen entsteht.

Normalerweise dürfte bei einem Neustart eines Masters oder bei Verwendung der Serialisierungsfunktionalität innerhalb eines Simulationslaufes stets nur der Fall b) eintreten.

Für das Kürzen der Ergebnisdateien gibt es mehrere Möglichkeiten. Der portabelste Weg, wenn auch nicht der schnellste, ist das Kopieren der zu erhaltenden Daten in eine temporäre Datei und abschließenden Umbenennen der Datei in die Originaldatei.

Fehlerquellen

Es wäre bei Binärdateien auch möglich, innerhalb der Datei den Dateizeiger zu verschieben und existierende Ausgaben so zu überschreiben. Dieses Prozedere birgt jedoch die Gefahr, inkonsistente Ergebnisdaten zu erzeugen. Bei einer Variationsrechnung soll z.B. nur die Lüftungsanlage im Sommer optimiert werden. Dazu rechnet der Master nur die Monate Mai bis August mehrfach durch und überschreibt dabei existierende Ergebnisse dieser Monate. Die Ergebnisdaten der Monate September bis Dezember der ursprünglichen Jahressimulation sind dann inkonsistent zu den in der letzten Variation berechneten Ausgaben, erkennbar an unphysikalischen Sprüngen in Ergebnisgrößen.

Werden Ergebnisdateien im Fall b) gekürzt, kann ein anderes Problem auftreten. Seitens der FMU kann nicht verhindert werden, dass der Master zuerst auf einen früheren Zeitpunkt zurücksetzt (Dateien werden gekürzt) und dann gleich darauf die FMU auf einen späteren Zeitpunkt serialisiert. In diesem Fall wären alle Ausgaben zwischen den Zeitpunkten verloren (eventuell als Fehler im Masteralgorithmus zu interpretieren).

Ein alternativer Ansatz zur Vermeidung derartiger Probleme wäre das Anlegen neuer Verzeichnisstrukturen bei jedem Simulationsstart und Deserialisierungsaufruf. Für die nachgelagerter Auswertung bestünde dann die nicht triviale Aufgabe, die verteilten Ausgabedateien korrekt zusammenzuführen.

ZUSAMMENFASSUNG

Bei der Transformation der Gebäudeenergiesimulation NANDRAD in eine FMU für Co-Simulation Version 2.0 mussten zahlreiche Details beachtet werden. Im Gegensatz zu Simulationsprogrammen mit festen Integrationsschrittweiten und darauf abgestimmten Ausgabeintervallen (z.B. EnergyPlus, TRNSYS), erforderte das Intergrationsverfahren mit variablen Zeitschritten in NANDRAD deutlich mehr Umsetzungsaufwand. Die Vorteile eines solchen modernen Verfahrens hinsichtlich numerischer Stabilität und Effizienz rechtfertigen jedoch den Mehraufwand.

Die Kern-Integrationsschleife wurde so umgeschrieben, dass eine stückweise Ausführung mit zwischenzeitlicher Rückgabe der Kontrolle an den Aufrufer möglich ist. Beim Integrieren über ein Kommunikationsintervall musste zudem sichergestellt werden, dass Modell- und Integratorzustände exakt die Start- und Endzeitpunkte des Intervalls abbilden. Entsprechend wurde das Zeitschrittadaptionsverfahren angepasst.

Für die effiziente Berechnung verwendet NANDRAD intern zustandsbasierte Modellobjekte, die über einen Auswertungsgraphen aktualisiert werden. Durch diese Form der Implementierung musste bei der Ausgabeinterpolation beachtet werden, dass der Modellzustand vor Rückgabe der Kontrolle an den Master wieder auf das Ende des Kommunikationsintervalls gesetzt wird. So kann der Master im Anschluss die korrekten Ergebnisgrößen direkt abgreifen.

Den weitaus größten Aufwand bei der Umsetzung erforderte die in Version 2.0 des FMI Standards hinzugefügte Funktionalität zum Setzen und Zurücksetzen eines FMU Zustands. Diese Funktionen wurden in NANDRAD modular umgesetzt, d.h. die Zustände der Komponenten *Integrator*, *Gleichungssystemlöser*, *Vorkonditionierer*, *Jacobimatrix*, und *Modell* werden nacheinander in einen sequentiellen Speicherbereich kopiert, bzw. aus diesem zurückkopiert. Die Implementierung wurde durch eine Makro-basierte Form erleichtert, welche die eigentliche Abbildung des zerklüfteten Speicherbereichs der jeweiligen Komponenten in einer einzigen Funktion erlaubt. Dieses erleichtert nicht nur die anfängliche Programmierung, sondern auch die Wartung des Quelltextes, da so die binäre Repräsentation der Modellzustände bei Serialisierung und Deserialisierung garantiert übereinstimmt. Die Makro-Implementierungsform war dabei auch für die in C geschriebenen Teile des Solvers direkt anwendbar.

Die Behandlung von Ergebnisdaten einer komplexen Simulation wie NANDRAD stellte sich als recht kompliziert dar. Bei FMUs, welche z.B. Modelica-Modelle kapseln, wird davon ausgegangen, dass alle interessanten Ergebnisgrößen über die FMU Schnittstelle an den Master gereicht werden und dort protokolliert werden. Daher wird auch bei der Initialisierung der FMU kein Basisverzeichnis für Ergebnisdaten spezifiziert, wodurch die Konvention eines `ResultRootDir` Parameters entstand.

Die Protokollierung von Gebäudesimulationsergebnisgrößen durch den Master ist jedoch aufgrund der Fülle von Ergebnisdaten nicht zweckmäßig. Gleiches gilt für Ergebnisgrößen von FMUs mit CFD-Modellen oder hygrothermischen Transportsimulationen. Daher wurde eine NANDRAD-spezifische Lösung konzipiert und umgesetzt.

In vielen Anwendungsfällen wird der Master jedoch die Simulation von Beginn an komplett durchlaufen, und keinen Neustart in der Mitte der Simulation durchführen. Daher ist die Serialisierungs- und Deserialisierungsfunktionalität im Kontext der Gebäudeenergiesimulation primär für Spezialanwendungen (z.B. Optimierung) notwendig.

DANKSAGUNG

Wir bedanken uns für die vom Bundesministeriums für Wirtschaft und Energie gewährte Förderung im Rahmen des Projektes „EnTool:CoSim“, Förderkennzeichen 03ET1215A.

LITERATUR

Hindmarsh A.C., Brown P.N., Grant K.E., Lee S.L., Serban R., Shumaker D.E. & Woodward C.S.; *SUNDIALS: Suite of nonlinear and differential algebraic equation solvers*, 2005, ACM TOMS, 31, 363-396

Modelica Association Project “FMI”, *Functional Mockup Interface for Model Exchange and Co-Simulation v. 2.0*, 2014

Nicolai A., Grunewald J., Zhang J.S.; *Coupling strategies for combined simulation using multizone and building envelope models*, Conference Proceedings of the 10th International Building Performance Simulation Association Conference and Exhibition, 2007, Beijing,Tsinghua University, pages 917-927

Paepcke A., Nicolai A.; *Anlagenregelung in ODE-Systemen am Beispiel der thermischen Raum- und Gebäudesimulation*, Tagungsband der BauSIM 2014, 5th German-Austrian Conference of the International Building Performance Simulation Association

Paepcke A., Nicolai A., Vogelsang St. & Grunewald J.; *Abschlussbericht über das Projekt „Dynamisches Raummodell“, Förderkennzeichen 0327241E*, 2014, Institut für Bauklimatik , TU Dresden

Paepcke A., Schwan T. & Nicolai A.; *Schnittstellen für die Co-Simulationskopplung zwischen Gebäude- und Heizungsanlagensimulation*, 2016, in Proceedings of the BauSim 2016, Dresden

Pazold M. & Antretter F.; *Hygrothermische Gebäudesimulation mit Multizonen-Gebäudedurchströmungsmodell*. 2013, Bauphysik, 35. (2), pp. 86-92.

Vogelsang St. & Nicolai A.; *Delphin 6 Output File Specification*, Technical Report, Qucosa, 2011, http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa-70337