DIE GEBÄUDESIMULATIONSPLATTFORM NANDRAD – PHYSIKALISCHES MODELL, UMSETZUNGSKONZEPT UND TECHNOLOGIEN IM ÜBERBLICK

Andreas Nicolai[1], Anne Paepcke[1]
[1]Institut für Bauklimatik, Fakultät Architektur, TU Dresden

KURZFASSUNG

Die Gebäudesimulationsplattform NANDRAD erlaubt die Berücksichtigung und Modellierung komplexer physikalischer Teilkomponenten, so z.B. raumaufgelöster Wandkonstruktionen. Die Vielzahl von gekoppelten Zonen-, Wand,- und Anlagenkomponenten und deren Verknüpfung erfolgt automatisiert aus einem Gebäudedatenmodell. Flexible, nutzerdefinierte Erweiterungen werden für komplexe Anlagen- und Regelungsmodelle zur Verfügung gestellt. Im Artikel werden grundlegende Konzepte vorgestellt und mit etablierten Plattformen wie TRNSYS verglichen und die Neuerungen herausgestellt. Die automatisierte, graphenbasierte Auflösung von Modellabhängigkeiten, die Behandlung nicht-linearer Subsysteme, und die Integration des Gesamtsystems wird vorgestellt. Der Schwerpunkt der numerischen Umsetzung liegt auf der Beherrschung großer Systeme mit nicht-linearer Kopplung unter Einhaltung geforderter Genauigkeiten unter Verwendung variabler Zeitschritte und Integrationsordnung.

The building simulation platform NANDRAD is capable of modeling complex physical components such as spatially discretized wall constructions. The large number and variety of coupled zonal, wall and HVAC models and their interaction is automatically composed based on a building information model (BIM). Flexible, user-defined extensions for complex plant models and control algorithms are supported. In the article we cover the fundamental concepts and compare those to established simulation models like TRNSYS. We highlight new features and capabilities and describe the automated graph-based evaluation of model interdependencies, the treatment of non-linear, coupled sub-systems and the integration of the resulting system. The focus of our numerical solution method lies on applicability to large systems with non-linear model coupling while maintaining requested integration tolerances. This is achieved through utilization of variable step sizes and integration method order.

EINLEITUNG

Im Bereich der Gebäudesimulation gibt es derzeit eine Vielzahl an praxistauglichen Tools, unter anderem TRNSYS, EnergyPlus, BSim, ESP-r, IDA, und viele weitere, welche unter anderem vom US Dep. of Energy (2012) gelistet werden.

Dabei haben nur wenige Programme den Anspruch eine erweiterbare Plattform zu sein. Darunter verstehen wir die Möglichkeit der funktionalen Erweiterung des Simulationssystems durch den Nutzer, auch nach Erstellen/Compilieren des Simulationssystemes. Dieses schließt auch die Unterstützung neuartiger Modelle und deren Parametrisierung ein, welche zur Konzeptionszeit des Gebäudesimulationskerns noch nicht bekannt oder eingeplant waren. Hinsichtlich dieser Funktionalität haben derzeit nach Auffassung der Autoren nur EnergyPlus und TRNSYS derartige Fähigkeiten, obwohl die große Komplexität moderner Anlagensysteme und Regelalgorithmen dieses immer häufiger verlangen.

Der Entwicklungsstand der numerischen Rechenkerne der meisten Gebäudesimulationstools ist dabei zumeist nicht auf dem aktuellen Stand der Technik und Forschung. Zum Beispiel wird explizite Eulerintegration in EnergyPlus verwendet (EnergyPlus, 2010), und TRNSYS (TRNSYS, 2010) setzt Einschrittverfahren in Verbindung mit Festpunktiteration bzw. Powell ein. Weder TRNSYS noch EnergyPlus kontrollieren dabei aktiv den numerischen Approximationsfehler.

Für die effiziente Integration großer nichtlinearer Differentialgleichungssysteme, welche bei komplexen Gebäude- und Anlagenmodellen entstehen, existieren bereits weiterentwickelte Verfahren, z.B. die Solver der SUNDIALS Bibliothek (Hindmarsh et al., 2005). Diese verwenden eine adaptive Anpassung der Methodenordnung und Zeitschrittsteuerung, integrierte Fehlerkontrolle, und Lösung nichtlinearer Gleichungssysteme mittels modifizierter Newton-Verfahren. Für die dabei häufig zu lösenden linearen Gleichungssysteme mit schwach besetzten Matrizen sind nach aktuellem Stand der Forschung Krylov-Unterraum-Methoden, wie z.B. GMRES, BiCGStab

ideal (vgl. Saad, 2003). In Kombination mit geeigneten Vorkonditionierern können so auch große Gebäudekomplexe, z.B. Bürogebäude mit > 200 Zonen und entsprechend vielen Wänden, effizient berechnet werden.

Derart große Systeme werden ebenso durch eine entsprechend komplexe Erweiterung der physikalischen Beschreibung erzeugt. So ist es möglich, anstelle von vereinfachten Wärmetransfer- und Speichermodellen für Konstruktionen (z.B. der CTF-Methode) auch raumaufgelöste Bauteile zu verwenden. Dieses ermöglicht genauere Abschätzung der tatsächlich zur Verfügung stehenden Wärmespeicherkapazität und zusätzlich der Berücksichtigung von integrierten Phasenwechsel-materialien (PCM). Vor allem für die in Europa häufig verwendete, eher massive Bauweise ohne aktive Klimatisierung ist die korrekte Abbildung des thermischen Speichervermögens unter anderem zur Bewertung von Behaglichkeitskriterien und zur Optimierung des sommerlichen Wärmeschutzes wichtig.

Eine Erweiterung existierender Simulations-plattformen ist aufgrund verschiedener Gründe schwierig, beziehungsweise nicht möglich. Neben lizenzrechtlichen Problemen stellen die zumeist monolithischen Fortran-Quelltexte eine Hürde für die Integration eines fundamental unterschiedlichen Lösungsverfahrens dar.

Zielsetzung

Im Rahmen eines BMWi Projektes wurde daher eine Technologieentwicklung mit dem Ziel einer modernen Gebäudesimulationsplattform angestoßen, mit folgenden Entwicklungszielen:

- Effizienter numerischer Rechenkern unter Verwendung moderner Verfahren,
- Parallelisiertes Lösungsverfahren für effiziente Nutzung von Mehrkern-Desktop-Rechnern,
- Projektdateigesteuerte Parametrisierung des Solvers, d.h. Trennung von Modellimplementierung und Datensätzen,
- Plattformfunktionalität, d.h. Erweiterung des Systems ist durch den Nutzer hinsichtlich Modellfunktionalität und Parametrisierung möglich.
- Kopplungsfähigkeit mit anderen Tools, u.a. auf Basis von aktuellen Schnittstellenstandards wie dem Functional Mockup Interface (MODELISAR, 2010)
- Implementierung in Standard C++ mit Unterstützung gängiger Betriebssysteme (Windows, Linux, Mac OS)

Ergebnis der bisherigen Arbeit ist die Simulationsplattform NANDRAD. Schwerpunkt der noch laufenden Forschung ist dabei die Entwicklung der Solvertechnologie und Demonstration anhand verschiedener Testfälle, jedoch nicht die Implementierung von zahlreichen Modellen und Modellvariationen, wie sie z.B. in TRNSYS zu finden sind. Im Folgenden sollen die wesentlichen Konzepte und Bausteine von NANDRAD vorgestellt werden.

PHYSIKALISCHES MODELL

Mehrzonenmodell

Den Kern von NANDRAD bildet eine klassische Mehrzonensimulation. Das bedeutet, für jede thermische Zone wird eine Energiebilanzgleichung (1) gelöst.

$$\frac{\mathrm{d}Q_R}{\mathrm{d}t} = \sum_i^{n_F} \dot{Q}_{F,i} + \sum_i^{n_W} \dot{Q}_{W,i} + \sum_j \dot{Q}_j \qquad (1)$$

Die Wärmespeicherung der umliegenden Konstruktionen wird hier über Transferterme zu den Wandoberflächen $\dot{Q}_{W,i}$ beschrieben. Solarstrahlung durch Fensterflächen des Raumes beinhaltet $\dot{Q}_{F,i}$, alle weiteren Wärmequellen und –senken sind in $\sum_j \dot{Q}_j$ zusammengefasst. Eventuell im Raum befindliche Speichermassen m_M werden dem Speicherterm der Raumluft, Gl. (2), zugeschlagen:

$$\frac{\mathrm{d}Q_R}{\mathrm{d}t} = \left(m_M c_M + \varrho_L c_L V_L\right)\frac{\mathrm{d}\vartheta_R}{\mathrm{d}t} \qquad (2)$$

Dabei bezeichnen c_M und c_L jeweils die spezifischen Wärmekapazitäten der Raumspeicher-massen und der Luft, ρ_L die Dichte der Luft und V_L das Luftvolumen. ϑ_R beschreibt die Raumluft-temperatur.

Wärmeaustausch zwischen Zonen, bzw. zwischen Zonen und der Umgebung erfolgt über Wärmeleitung durch Wand-, Fußboden- und Deckenkonstruktionen, sowie Enthalpietransport aufgrund von Anlagen-technik. Dieser schließt auch Leckageströme zwischen Zonen aufgrund von lüftungsbedingten Über- und Unterdrücken ein.

Alle Konstruktionen werden mittels eindimensionaler instationärer Wärmeleitung beschrieben. In der numerischen Lösung wird je nach Wandaufbau eine automatische Diskretisierung mit variabler Schichtdicke durchgeführt. Für jedes so entstandene Finite Volumen wird eine Energiebilanzgleichung gelöst. Die raumseitigen Randbedingungen stellen

dann die Kopplungsterme zu den dazugehörigen Raumenergiebilanzen her. Zusätzliche Kopplung der Wandflächen durch innenseitigen langwelligen Strahlungsaustausch ist angedacht, wird aber im aktuellen nicht-geometrischen Datenmodell noch nicht unterstützt.

Außenseitig wird Wärmeübergang an die Umgebungsluft berücksichtigt. Weiterhin wirkt kurzwellige Solarstrahlung auf die Wand mit konstantem Absorptionskoeffizient. Das Solarstrahlungsmodell, welches Lasten für Wand- und Fensterflächen liefert, ist eine austauschfähige Komponente und bildet derzeit lediglich den Einfluss von Ausrichtung/Neigung der Flächen und die jeweilige Eigenverschattung ab.

Anlagentechnik und Regelung

Neben der thermischen Bilanzierung existieren zahlreiche weitere Modelle für Anlagentechnik, Nutzerverhalten und andere. Diese erzeugen Wärmequellen und –senken im Raum und haben so eine Rückwirkung auf die Raumenergiebilanzgleichungen. Im Fall von Luftströmungsmodellen bzw. Rohrheizungssystemen können diese Wärmeströme von den Zustandsgrößen in mehreren Zonen abhängen. Auch entstehen dabei zyklische Zusammenhänge, welche es zu berücksichtigen gilt.

Insbesondere Anlagenmodelle können zusätzliche Erhaltungsgrößen S_e verlangen, die innerhalb der Raumbilanzen nicht explizit vorgesehen sind. Diese müssen ebenso im Kontext des Lösungsverfahrens berücksichtigt werden.

LÖSUNGSVERFAHREN

Mathematische Formulierung

Die Erhaltungsgrößen aus den Raum- und Wandbilanzgleichungen sind die Energien $Q_{R,k}$ (k - Laufindex der Zonen), die Energiedichten $u_{i,j}$ (i - Laufindex der Konstruktionen, j - Laufindex der Diskretisierung), und die zusätzlichen Erhaltungsgrößen S_e aus expliziten Modellen.

Werden diese zusammengefasst in einem Vektor $y = \left[Q_{R,1},\ldots,Q_{R,nR},u_{1,1},\ldots,u_{1,m1},u_{2,1},\ldots,u_{nW,mW},S_0,\ldots,S_{ne}\right]^T$, kann das mathematische Problem abstrakt als System gewöhnlicher Differenzialgleichungen geschrieben werden, Gl. (3).

$$\frac{\partial y}{\partial t} = f(t,y) \tag{3}$$

Dieses genügt der Form klassischer Anfangswertprobleme welche durch Zeitintegration gelöst werden können.

Lösungsalgorithmus

Zur Integration des Systems gewöhnlicher Differentialgleichungen bedienen wir uns etablierter Solvertechnologie, dem CVODE Solver aus der SUNDIALS Bibliothek (Hindmarsh et al., 2005).

Kern des Lösungsalgorithmus ist ein Mehrschrittverfahren, welches durch variable Anpassung von Methodenordnung und Zeitschrittgröße ein Optimum an Simulationsgeschwindigkeit bei Einhaltung geforderter Genauigkeiten ermöglicht. Intern wird eine polynomiale Approximation der zeitlichen Veränderung der Erhaltungsgleichungen angesetzt. Das Predictor-Corrector-Verfahren nutzt dabei das Extrapolationspolynom zur Vorhersage der nächsten Lösung und korrigiert die Schätzwerte daraufhin in einer modifizierten Newton-Iteration. Der Integrationsfehler wird durch Taylorreihen-entwicklung bestimmt, wobei maßgeblich die Differenz zwischen der Schätzlösung des Extrapolationspolynoms und der exakten Lösung eingeht.

Die Zeitschrittweiten werden dynamisch dem berechneten Integrationsfehler und den Konvergenzraten angepasst. Dabei reagiert der Solver sehr schnell auf wechselnde Zeitschrittanforderungen, wie sie in Gebäude-simulationen mit anspruchsvoller Regelungstechnik häufig auftreten.

Problemspezifische Optimierung

Zur effizienten Nutzung des CVODE Solvers müssen verschiedene Aufgaben seitens des Nutzers, d.h. bei der Implementierung des physikalischen Modells, geleistet werden:

1. effiziente Implementierung der Systemfunktion $f(t,y)$,
2. optimale Anordnung der Erhaltungsgrößen im Vektor y zur Minimierung der Bandbreite in der Jacobimatrix, welche im Newton-Verfahren benötigt wird,
3. Wahl eines geeigneten iterativen Lösers für lineare Gleichungssysteme und Bestimmung optimaler Abbruchkriterien,
4. Auswahl und Implementierung eines problemangepassten Vorkonditionierers.

Weiterhin sind zahlreiche Verwaltungs- und Datenstrukturen zu implementieren.

Die Aufgabenkomplexe 2-4 werden maßgeblich von der Entscheidung beeinflusst, ob eine parallelisierte oder serielle Numerik verwendet werden soll. Bei serieller Implementierung ist die Reduktion der

Bandbreite essentiell, da innerhalb der Band- oder ILU-Vorkonditionierer direkte Verfahren im Lösungsschritt der Vorkonditionierer eingesetzt werden. Bei paralleler Implementierung stehen typischerweise die Kommunikationszeiten im Vordergrund, sodass Block-Band-Vorkonditionierer mit einer geeigneten Systemzerlegung empfehlenswert erscheinen. Beim aktuellen Stand der Entwicklung kann jedoch noch keine abschließende Empfehlung hinsichtlich der parallelen Implementierung (auf Desktop Mehrkernsystemen mit gemeinsamen Speicher ohne Ethernet-kommunikation) gegeben werden.

Bandbreitenreduktion

In der aktuellen Implementierung wird die Bandbreitenreduktion des Systems durch einen graphentheoretischen Ansatz, dem Cuthill-McKee-Algorithmus (Cuthill and McKee, 1969) durchgeführt. Dabei werden alle Zustandsgrößen einer Wand zusammen als ein Knoten aufgefasst, und die Anordnung von Räumen und Wänden innerhalb des Lösungsvektors wird hinsichtlich der geringstmöglichen Bandbreite variiert. Dies ist deshalb möglich, da im aktuellen Modell lediglich die äußeren Wandelemente mit Raumbilanzgleichungen verknüpft sind.

Vorkonditionierer und iterativer Löser

Die große Variabilität der zu lösenden Probleme birgt das Problem der Konvergenzabschätzung innerhalb des linearen Gleichungssystemlösers. Zur Erläuterung betrachte man einen Schritt im nichtlinearen Newton-Verfahren, Gl. (4).

$$\left.\frac{\partial F}{\partial y}\right|_m \Delta y^{m+1} = -F\left(t, y^m\right) \tag{4}$$

Die Jacobi-Matrix $\partial F/\partial y$ wird mit bekannten Größen zum Iterationsschritt *m* ausgewertet, ebenso wie die Systemfunktion $F\left(t, y^m\right)$, welche durch Einsetzen des Interpolationspolynoms in die Differenzialgleichung entsteht. Die Änderung der Erhaltungsgrößen Δy^{m+1} von einem Newtonschritt zum nächsten wird durch Lösung des Gleichungssystems (4) berechnet. Das Konvergenzkriterium des Newton-Verfahrens fordert nun, dass die Vektornorm der Änderung Δy^{m+1} eine definierte Schranke unterschreitet.

Wird das Gleichungssystem nun iterativ gelöst, d.h. Gl. (4) wird in Matrixnotation übersetzt,

$$Ax = b \qquad \mathrm{x} \equiv \Delta \mathrm{y}^{m+1}, A \equiv \left.\frac{\partial F}{\partial y}\right|_m$$

so muss die Ergebnisgröße *x* mindestens so gut approximiert werden, dass die Fehlergrenze des Newton-Schritts prinzipiell eingehalten werden kann. Iterative Verfahren wie GMRES oder BiCGStab liefern jedoch in jedem Schritt nur das Residuum $r = b - Ax$ zurück, nicht jedoch die Lösungsgröße *x*. Damit ist die Forderung nach einer Mindestgenauigkeit der Lösungsvariable in eine Unterschreitung einer Schranke des Residuums zu überführen. Dies ist jedoch bei flexiblen Modellen und damit dynamisch veränderlichen Jacobi-Matrizen und zusätzlich verwendeten Vorkonditionierern schwierig und derzeit noch Teil der Forschung. Deshalb wird in der aktuellen Implementierung ein extrem niedriger Residualwert gefordert, um die Genauigkeitsforderung des Newton-Verfahrens in jedem Fall zu befriedigen.

Weiterhin gibt es je nach Iterationsverfahren einen mehr oder weniger glatten Verlauf der Residuen bzw. der Lösungsvariable. Das bedeutet unter anderem, dass das Verhältnis zwischen Norm des Residiums und Norm des Lösungsvektors stark schwankt. Daher wird in der aktuellen Implementierung ein GMRES Verfahren verwendet, welches einen glatten Konvergenzverlauf aufweist.

Effiziente Implementierung der Systemfunktion

Der wichtigste Aspekt bei der Implementierung einer effizienten Gebäudesimulationsplattform ist die Auswertung der Modellbausteine. In NANDRAD werden alle physikalischen Gleichungen oder Modellkomplexe in sogenannten Berechnungs-modellobjekten gekapselt. Unter einem *Modellobjekt* wird eine konkrete Instanz eines *ModelTyps* verstanden. Ein *ModelTyp* kapselt seine physikalische Modellbeschreibung in Form einer objektorientiert implementierten Klasse.

Instanzen eines Modeltyps können mehrfach mit unterschiedlicher Parametrisierung auftreten, so existieren z.B. innerhalb eines Gebäudes mehrere thermische Zonen, abstrahiert als Modellobjekte des Typs *Raumbilanz (RoomBalanceModel)*.

Jedes dieser Modellobjekte hat einen oder mehrere Ergebniszustände. Weiterhin verfügt jedes Modellobjekt über eine Aktualisierungsroutine, in welcher alle Ergebniszustände berechnet werden.

Als Beispiel sei der Modelltyp *Wärmeübergang zwischen Wandfläche und Raum* genauer ausgeführt (Gl. 5).

$$\dot{Q} = A\alpha\left(\vartheta_W - \vartheta_R\right) \tag{5}$$

Der Ergebniszustand ist der Wärmestrom $\dot{Q}$ zwischen Wand und Raum. Die Wandober-flächentemperatur ϑ_W und die Raumtemperatur ϑ_R sind Eingangsgrößen, der Übergangskoeffizient α und die Fläche A konstante Parameter. Bei der Auswertung der Gleichung bezieht der Modelltyp alle

Eingangsgrößen von anderen, vorher aktualisierten Modellobjekten. In diesem Fall werden die Temperaturen von dem Modellobjekt der Wand und dem Modellobjekt des benachbarten Raumes bezogen.

In der Implementierung wird für jede Modellergebnisgröße ein fester Speicherbereich definiert. Abhängige Modellobjekte greifen direkt auf diesen Speicherbereich zu, welches eine sehr effiziente Berechnung ohne Erzeugung temporärer Variablenkopien erlaubt. Unter anderem aus diesem Grund stellt sich Implementierung in der Programmiersprache C++ als besonders günstig heraus.

Graphenbasierte Bestimmung der Modellauswertungsreihenfolge

Die Auswertung des physikalischen Gesamtmodells beschränkt sich aus Sicht der Systemfunktion auf die sequenzielle bzw. parallele Aktualisierung von Modellbausteinen. Dabei ist aufgrund der oben geschilderten Modellabhängigkeiten die Reihenfolge der Auswertung wichtig.

Da die Simulationsplattform generisch erweiterbar sein soll, d.h. nutzerdefinierte Modelle können neue Abhängigkeiten einführen, lässt sich keine Auswertungsreihenfolge im Programm fixieren. Allerdings kennt jedes Modellobjekt eine Liste aller Modellobjekte, von denen es Eingangsgrößen bezieht.

Modellobjekte, verbunden durch Abhängigkeiten, formen einen Graphen, beispielhaft dargestellt in Abbildung 1. Von den Autoren wurde nun ein Graphenalgorithmus zur Auflösung der Berechnungsreihenfolge entwickelt. Dieser Algorithmus wandelt den ursprünglichen, ungeordneten Graphen in einen geordneten Graphen um. Dabei werden zunächst Gruppen identifiziert, welche nach den folgenden Kriterien zu unterteilen sind:

- Sequenzen von Modellobjekten, die Auswertung erfolgt nacheinander
- parallele Regionen, diese Modellobjekte beziehungsweise Sequenzen können parallel ausgewertet werden, und
- zyklisch verknüpfte Gruppen von Modellobjekten, die Aktualisierung dieser Zyklen bedingt das Lösen eines nicht-linearen Gleichungssystems

Abbildung 2 zeigt die Umordnung des originalen Graphen in die geordnete Form durch den Graphenalgorithmus.

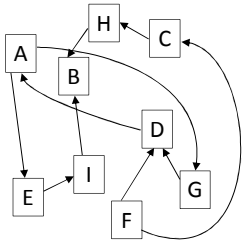

Abbildung 1 Ungeordneter Graph von Modellobjekten und deren Verknüpfung

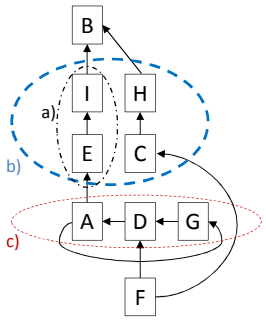

Abbildung 2 Umwandlung in einen geordneten Graphen und Identifikation der Gruppen: a) Sequenz von Modellobjekten, b) parallele Auswertung mehrerer Sequenzen/Zyklen, c) zyklisch abhängige Modellobjekte (gekoppelte Lösung)

Bei der Initialisierung wird nun eine Datenstruktur generiert, die als primäre Ordnungsstruktur eine Sequenz von Aktualisierungsschritten beinhaltet. Während der Modellauswertung werden diese hintereinander abgearbeitet. Jeder Aktualisierungsschritt enthält eine Anzahl von parallelisierbaren Sequenzen bzw. Zyklen. Sind alle aktualisiert, wird zum nächsten Schritt übergegangen.

Die Auswertung des Graphen in Abbildung 2 erfolgt in 4 Schritten: zunächst Modellobjekt B, dann parallel die Sequenzen I – E, und H – C, drittens die Lösung des Zyklus A-D-G und zuletzt das Modellobjekt F.

Schlußendlich sind alle Modellobjekte aktualisiert und das Ergebnis der Systemfunktion an den Integrator zurückgeliefert.

Zyklenberechnung

Eine zentrale Aufgabe bei der Modellauswertung ist die Lösung zyklisch abhängiger Modellobjekte. Solche entstehen zum Beispiel bei Berücksichtigung einer geregelten Heizungsanlage. Zur Berechnung werden alle Ergebnisgrößen der beteiligten Modellobjekte in einen Vektor überführt, und das nichtlineare Gleichungssystem wird mittels Newton-Verfahren gelöst. Dabei wird berücksichtigt, dass sich dieses Gleichungssystem innerhalb eines übergeordneten nichtlinearen Newton-Algorithmus

befindet. Um numerische Robustheit zu sichern, wird das innere Gleichungssystem entsprechend mit höherer Genauigkeit gelöst.

GENERIERUNG DES MODELLGRAPHEN AUS DER PROJEKTDATEI

Die Parametrisierung der Berechnungsmodelle wird dem Simulationssystem in Form einer Projektdatei übergeben. Materialdaten, Konstruktionsaufbauten und Klimadaten werden in Datenbanken verwaltet und aus der im XML-Format abgelegten Projektdatei referenziert. Während der Initialisierung des Berechnungskerns werden Modellobjekte/Instanzen von Modelltypen aus der Parametrisierung generiert.

Hierbei wird grundsätzlich zwischen implizit und explizit parametrisierten Modellen unterschieden.

Explizite Modellparametrisierung

Explizite Parametrisierung bedeutet in diesem Kontext, dass jedes Modellobjekt über einen Parameterblock in der Projektdatei definiert wird. Teil der Definition ist die explizite Angabe des jeweiligen Modelltyps. Weiterhin wird eine Liste mit Parametern (Werte, Zeichenketten, etc.) für die Initialisierung übergeben. Der Solver kann anhand des eindeutigen Typbezeichners aus einer Moduleregistrierung das jeweilige Modellobjekt instanziieren und mit dem Parameterblock initialisieren.

Jeder Parameterblock und die dazugehörige Modellinstanz wird durch eindeutige ID Nummern gekennzeichnet. Verknüpfungen zwischen Modellobjekten werden durch Modell-Eingangs-daten-Referenzen (ModelInputReferences) explizit angegeben. So kann ein Heizkörpermodellobjekt die Flüssigkeitstemperatur eines Zulaufrohrs benötigen. Als Eingangsdaten-Referenz wird die ID des Zulaufrohrobjekts und der ID-Name der physikalischen Ergebnisgröße des Rohrmodells, in diesem Fall der Temperatur definiert.

Implizite Modellparametrisierung

Würde ein komplexes Gebäudemodell ausschließlich über explizite Modellparametrisierung beschrieben, wären sehr viele Definitionen und Verknüpfungen notwendig. Dies ist äußerst aufwändig und fehleranfällig. Für häufig verwendete Modell-komponenten sind daher Modelltypen und dazu-gehörige Parametrisierungsblöcke, samt der Verknüp-fungsinformationen, fest im Solver verankert.

Die Parametrisierung erfolgt hierbei in Form einer Beschreibung des Gebäudes durch ein Gebäudedatenmodell (Building Information Model – BIM). Dabei wird die Geometrie des Gebäudes bereits aufbereitet für die energetische Simulation als Liste von Zonen, Konstruktionen, Fenstern und dazwischenliegenden Berührungs-/Kontaktflächen. Aus dieser Gebäudebeschreibung werden automatisch eine Vielzahl von Berechnungsmodellobjekten generiert und die jeweiligen Verknüpfungen, d.h. Abhängigkeiten der Objekte untereinander festgelegt.

Typische Vertreter für implizite Modellobjekte sind die Raumbilanzen, Konstruktionssolverobjekte, Rand- und Übergangsbedingungen, Strahlungs-transmissionsmodelle durch Fenster, und entsprechende Kopplungen zum Klimamodell.

Das Klimamodell selbst, welches die Berechnung der Umgebungsbedingungen sowie der gerichteten Solarstrahlung übernimmt, ist ebenfalls ein solch automatisch generiertes Einzelmodell. Mit Standort des Gebäudes und Klimadatenreferenz kann das Klimamodul aus der Projektdatei initialisiert werden. Alle impliziten Modelle mit klimatischen Eingangsgrößen werden automatisch mit diesem ausschließlich zeitabhängigen Berechnungsmodell verknüpft.

Die Kenntnis der Modelleigenschaften und Verknüpfungen seitens des Solverkerns erleichtert nicht nur die Parametrisierungsarbeit seitens des Nutzers, sondern erlaubt zudem spezifische Optimierungen, z.B. in der Erstellung des Vorkonditionierers.

Rückwirkungen auf implizit generierte Modellobjekte

Für die Rückwirkung von explizit erstellten Modellobjekten auf implizite Modellobjekte wird ein spezieller Mechanismus benötigt. Da im Modell-Verknüpfungskonzept von NANDRAD nur Eingangsgrößen aus anderen Modellen referenziert werden, und implizite Modelltypen die optional definierten expliziten Modelltypen nicht kennen, werden sogenannte Kollektoren-Modellobjekte zum Einsammeln der Ergebnisgrößen expliziter Modelle generiert. Zum Beispiel enthält jede Zonenwärmebilanzgleichung einen Summationsterm aus allen zugeführten Wärmelasten. Das zonenspezifische Kollektor-Modell summiert nun die als Zonen-Wärmelast definierten Ergebnisgrößen aller der Zone zugeordneten expliziten Modellinstanzen. Diese Zuordnung wird über eine Modell-Rückwirkungs-Referenz im Parameterblock des expliziten Modells definiert.

Nutzerdefinierte Erweiterung von Modell und Parametrisierung

Für die Erweiterung des Frameworks hinsichtlich nutzerdefinierter Modelle wird ebenfalls die explizite Modellparametrisierung verwendet. Diese Modelle verlangen zudem die Implementierung eines benutzerdefinierten Modelltypes. Dieser wird durch Ausprogrammieren einer generischen Modell-

schnittstelle für die Verwendung in der NANDRAD Plattform vorbereitet. In der derzeitigen Implementierung muss der neue Modelltyp der Modellregistrierung zur Compilezeit bekannt sein. Später soll der Modelltyp dynamisch zur Laufzeit des Solvers geladen und die ID des neuen Modells automatisch in die Modellregistrierung übernommen werden. Beim Laden der Projektdatei behandelt der Solver den nutzerdefinierten Modelltyp dann wie jeden anderen expliziten Modelltyp und fügt ihn in die Simulation ein.

PROJEKTERSTELLUNG UND ERGEBNISANALYSE

NANDRAD ist derzeit eine Konsolenanwendung. Also solche läuft der Rechenkern auf allen gängigen Betriebssystemen. Eine Entwicklung einer nutzerfreundlichen Programmoberfläche ist derzeit nicht geplant, stattdessen werden Konverter zu EnergyPlus IDF-Dateien sowie IFC-basierten BIM Daten entwickelt. Der Import von IDF-Dateien erlaubt indirekt die Nutzung der vielfältig verfügbaren Frontends für IDF-Datei-Erstellung, z.B. DesignBuilder u.ä.

Ergebnisse der Simulation werden in einem standardisierten DATAIO-Format (Vogelsang und Nicolai, 2011) geschrieben. Dieses Format wird einheitlich von allen am IBK entwickelten Solvern verwendet und kann in einem gemeinschaftlichen Postprozessing Tool (in Entwicklung) verwendet werden.

ZUSAMMENFASSUNG

Im Artikel wurden Konzepte und Grundlagen, auf denen die NANDRAD Simulationsplattform erstellt und implementiert wurde, präsentiert, erläutert und diskutiert. Im Besonderen wurden die grundlegenden Algorithmen bei der Modellauswertung sowie Solverinitialisierung beschrieben. Ein graphenbasierter Berechnungsalgorithmus für die Aktualisierungsreihenfolge der Modellobjekte wurde vorgestellt.

Die Unterstützung der impliziten Modellparametrisierung erlaubt einfache Definition von Gebäuden mit den am häufigsten verwendeten Modellbausteinen einer Mehrzonengebäudesimulation. Diese Funktionalität entspricht im Wesentlichen dem von klassischen Gebäudesimulationsprogrammen ohne Erweiterungsfähigkeit.

Individuelle Modelldefinitionen und entsprechend flexible Modellverknüpfungen werden durch explizite Modellparametrisierungen unterstützt. Damit erhält NANDRAD die geforderte Plattformfunktionalität und lässt sich durch Nutzer flexibel erweitern.

Stand der Entwicklung und Ausblick

Die in diesem Artikel vorgestellte Technologie ist im Rechenkern bereits umgesetzt. Die Modellvielfalt ist derzeit noch begrenzt, sodass Gebäude zunächst passiv bzw. mit idealen Heizungssystemen effizient gerechnet werden können. Die Spezifikation und Unterstützung von expliziten Modellen mit eigenen Zustandsgrößen ist Bestandteil aktueller Arbeiten. Der numerische Rechenkern ist zurzeit seriell implementiert, d.h. die parallel auszuwertenden Modellsequenzen werden noch nacheinander bearbeitet. Eine Parallelisierung der Systemfunktion basierend auf OpenMP ist Teil der aktuellen Forschungsarbeit.

Weitere Forschungsschwerpunkte bei der Solverentwicklung sind die Optimierung der Berechnung großer Modellobjekt-Zyklen, wie sie bei komplexen Haustechnikanlagen auftreten.

Die Spezifikation der Projektdatei, des Datenmodells und weiterer Schnittstellen wird fortlaufend auf der NANDRAD-Webseite dokumentiert (Nicolai et al., 2012). Eine erste öffentlich verfügbare Solverversion wird für 2013 erwartet.

DANKSAGUNG

Die Autoren sind für die gewährte Unterstützung bei der NANDRAD-Entwicklung im Rahmen des BMWi Forschungsprojektes „Energieoptimiertes Bauen“ (Förderkennzeichen 0327241E) dankbar.

LITERATUR

US Dep. of Energy, Building Energy Software Tools Directory, http://apps1.eere.energy.gov/buildings/tools_directory, 16.5.2012

Crawley et al., 2008, Contrasting the capabilities of building energy performance, Building and Environment, Vol. 43, p. 661–673

EnergyPlus, 2010, EnergyPlus Engineering Reference, LBNL

TRNSYS, 2010, TRNSYS 17, Vol. 7, Programmers Guide

MODELISAR (07006), Functional Mock-up Interface for Model Exchange, V1.0, Jan. 2010, Technical Specification, http://www.functional-mockup-interface.org

Cuthill E. and McKee J., 1969, Reducing the bandwidth of sparse symmetric matrices, In ACM Proceedings of the 1969 24th national conference, 157-172

Hindmarsh, A. C. et al., 2005, SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equations Solvers, ACM Trans. Math. Softw., ACM, 31, 363-396

Nicolai et al., 2012, NANDRAD Webseite, http://www.bauklimatik-dresden.de/nandrad

Vogelsang, St. and Nicolai A., 2011, Delphin 6 Output File Specification,http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa-70337

Saad Y. 2003. Iterative Methods for Sparse Linear Systems, SIAM Society for Industrial and Applied Mathematics, Philadelphia.