EnOB

Forschung für
Energieoptimiertes Bauen

Symposium

„Integrale Planung und Simulation in Bauphysik und Gebäudetechnik“

26.3.2012 – 28.3.2012

Gefördert durch das

Bundesministerium
für Wirtschaft
und Technologie

TECHNISCHE
UNIVERSITÄT
DRESDEN

ptJ
Projektträger Jülich
Forschungszentrum Jülich

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Architektur, Institut für Bauklimatik, Professur für Bauphysik

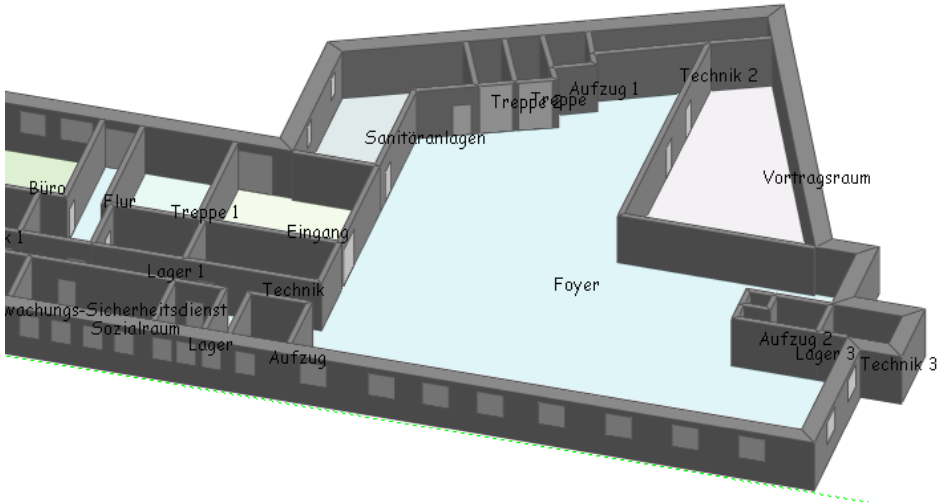
NANDRAD – Überblick über die neue Gebäudesimulations- und Modellierungs-plattform

Dr. Andreas Nicolai und Dr. Anne Paepcke

Dresden, 27.03.2012

NANDRAD – ein klassisches Mehrzonenmodell

Komplexe thermische Gebäudesimulation und energetische Optimierung



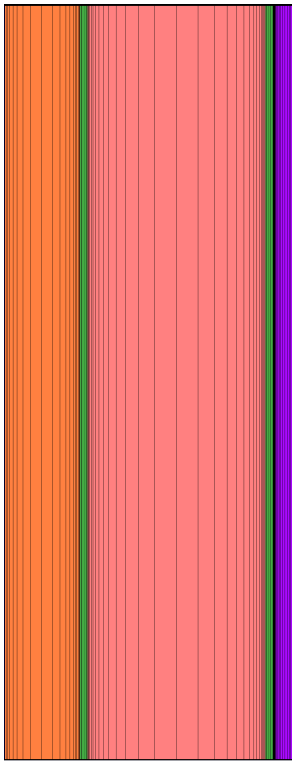
Vorhersagegenauigkeit

- detaillierte Wandkonstruktionen
- nichtlineare Anlagenkomponenten

Modellgröße und Rechengeschwindigkeit

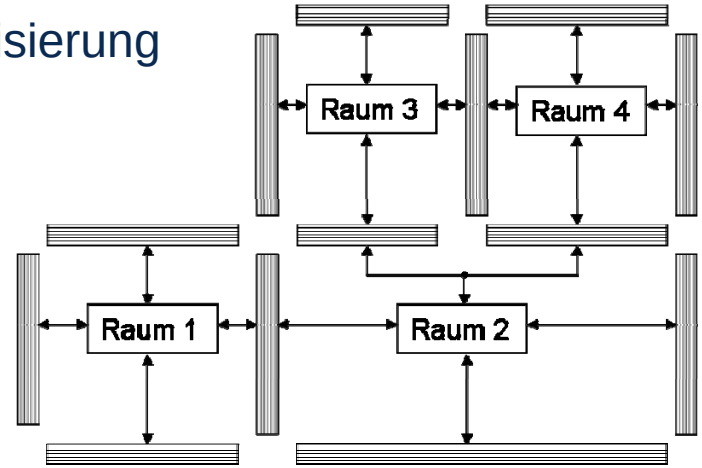
- Numerische Methoden für große Systeme
- Optimale Parallelisierung

Detaillierte Wandaufbauten



1	Historical Brick
2	Lime cement plaster
3	Calcium silicate
4	Glue mortar
5	Clay mortar
6	Foam glass
7	Clinker

- eindimensionaler Wandaufbau
- Berechnung von Wandtemperaturfeldern und induzierten Wärmeströmen
- numerische Diskretisierung (Finite Volumen)



Bilanzgleichungen

Raummodell

- je Raum/Zone:

$$C\frac{dT}{dt} = \sum \dot{Q}$$

Wandmodell

- durch Raumdiskretisierung entstehen *mehrere* Differenzialgleichungen für *jedes* Bauteil (Wand,Decke,...)

$$\frac{du_i}{dt} = \frac{A}{V_i}\left(\lambda_{i-1/2}\frac{T_{i-1}-T_i}{\Delta x_{i-1/2}} - \lambda_{i+1/2}\frac{T_i - T_{i+1}}{\Delta x_{i+1/2}}\right)$$

TECHNISCHE UNIVERSITÄT DRESDEN

Zeitintegration

Lösung der Bilanzgleichungen

Raummodell

$$C\frac{dT}{dt} = \sum \dot{Q}$$

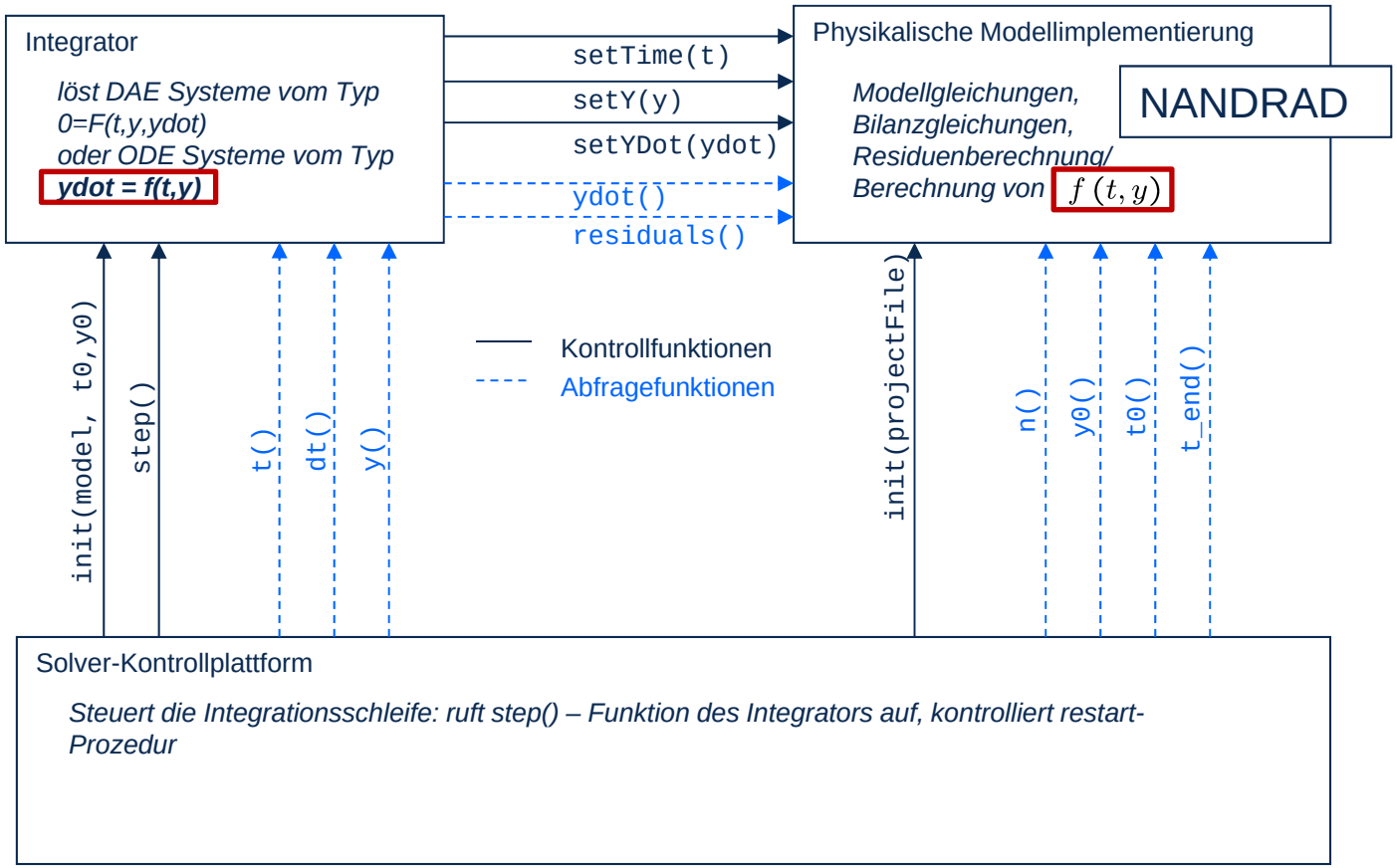
Wandmodell

$$\frac{du_i}{dt} = \frac{A}{V_i}\left(\lambda_{i-1/2}\frac{T_{i-1}-T_i}{\Delta x_{i-1/2}} - \lambda_{i+1/2}\frac{T_i - T_{i+1}}{\Delta x_{i+1/2}}\right)$$

$$\dot{y} = f\left(t, y\right)$$

System gewöhnlicher Differenzialgleichungen

Integrationsplattform



NANDRAD Modell-Schnittstelle

```
class NandradModel : public SOLFRA::ModelInterfaceODE {
public:
    /*! Number of unknowns/states, size n. */
    virtual unsigned int n() const;
    /*! Initial condition vector, size n. */
    virtual const double * y0() const;
    /*! Start time point in [s]. */
    virtual double t0() const;
    /*! Initial time step. */
    virtual double dt0() const;
    /*! End time point in [s]. */
    virtual double tEnd() const;

    /*! Update state of model to new time point. */
    virtual CalculationResult setTime(double t);

    /*! Update state of model to new set of unknowns. */
    virtual CalculationResult setY(const double * y);

    /*! Stores the computed derivatives of the solution variables in the vector ydot. */
    virtual CalculationResult ydot(double * ydot);
```

Funktionen zum Setzen eines neuen Zustands

NANDRAD Modell-Schnittstelle

```
class NandradModel : public SOLFRA::ModelInterfaceODE {
public:
    /*! Number of unknowns/states, size n. */
    virtual unsigned int n() const;
    /*! Initial condition vector, size n. */
    virtual const double * y0() const;
    /*! Start time point in [s]. */
    virtual double t0() const;
    /*! Initial time step. */
    virtual double dt0() const;
    /*! End time point in [s]. */
    virtual double tEnd() const;

    /*! Update state of model to new time point. */
    virtual CalculationResult setTime(double t);

    /*! Update state of model to new set of unknowns. */
    virtual CalculationResult setY(const double * y);

    /*! Stores the computed derivatives of the solution variables in the vector ydot. */
    virtual CalculationResult ydot(double * ydot);
```

Abrufen der Modell-antwort

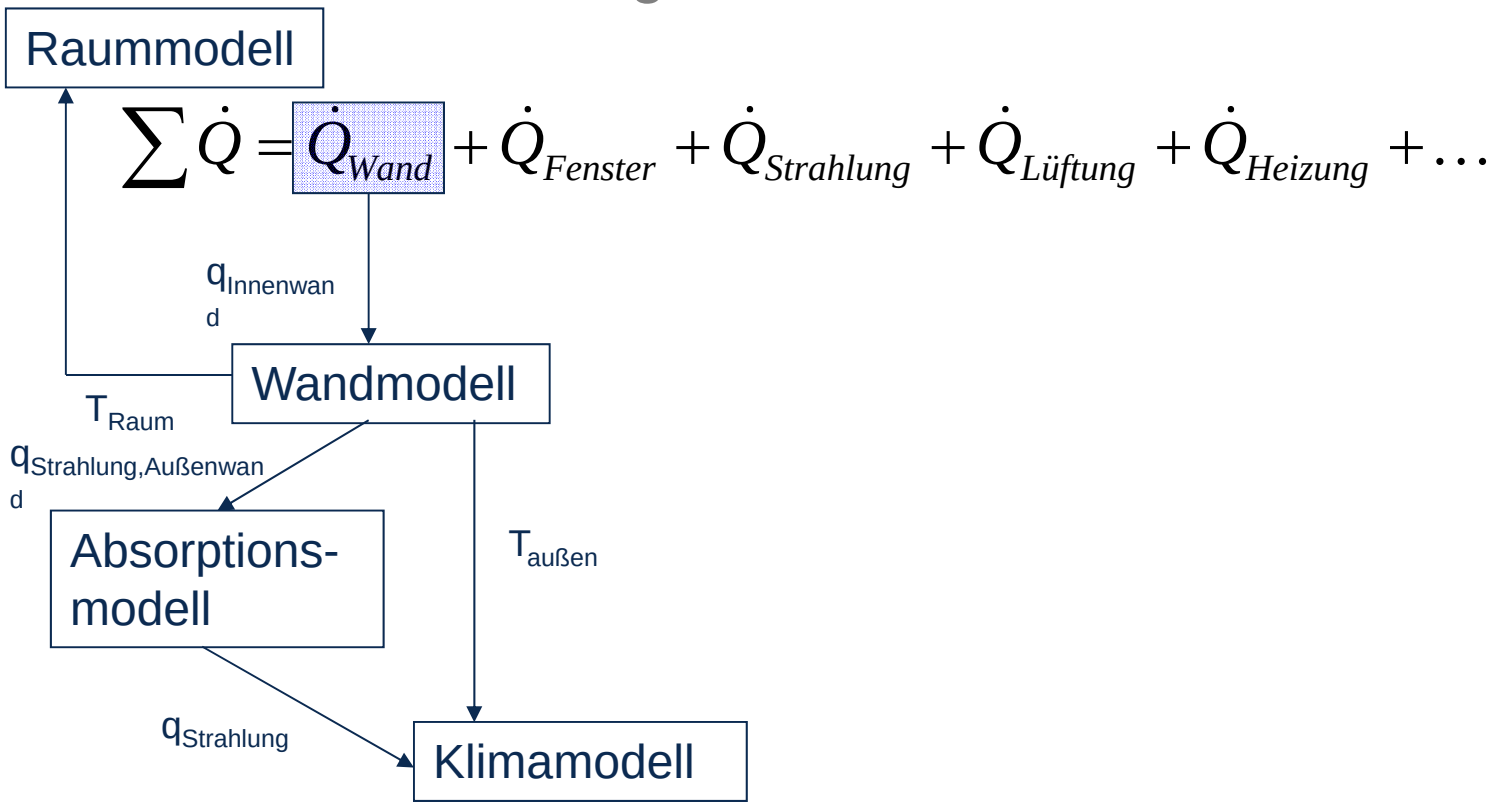
TECHNISCHE UNIVERSITÄT DRESDEN

Auswertung des physikalischen Modells

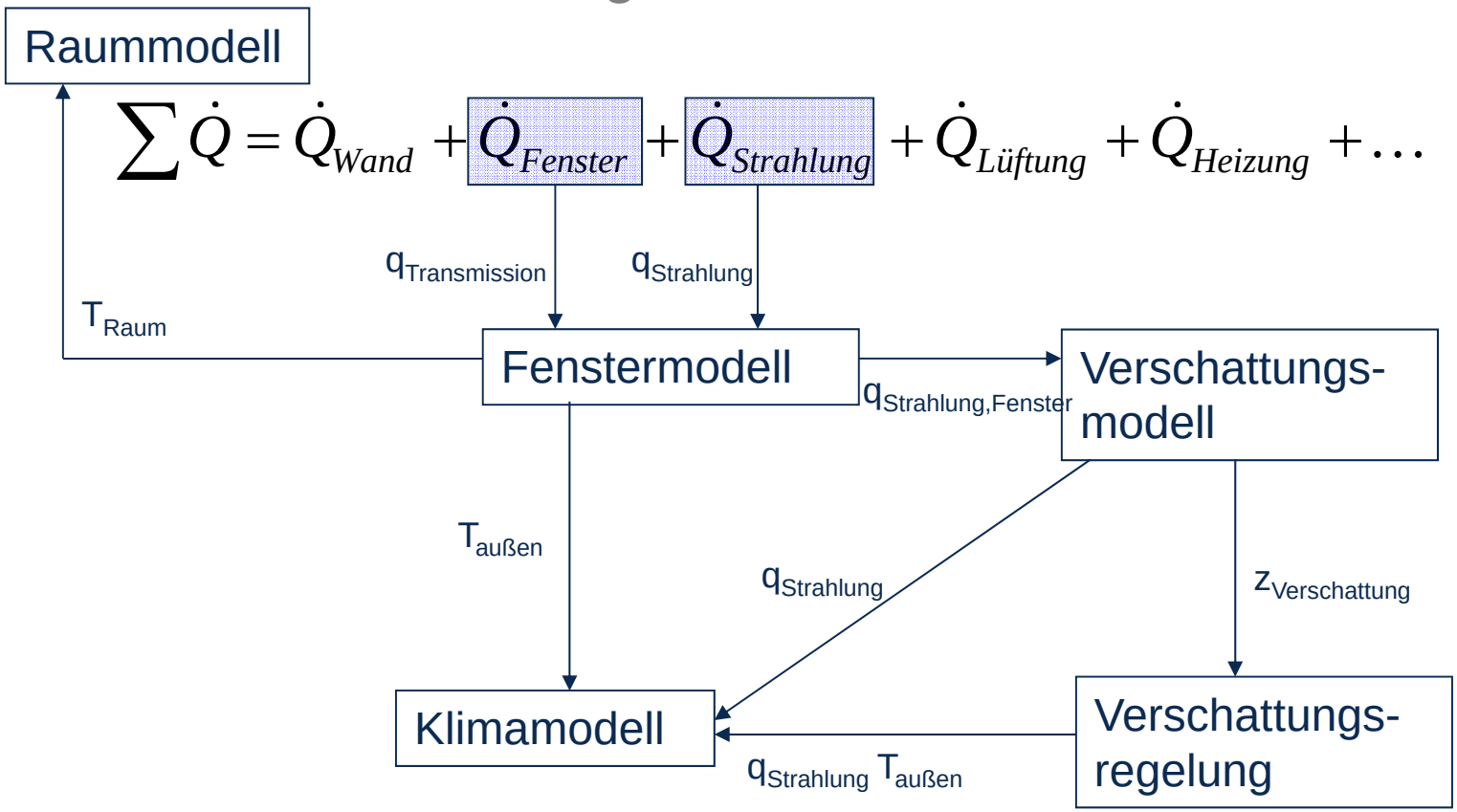
Modellauswertung

$$\sum \dot{Q} = \dot{Q}_{Wand} + \dot{Q}_{Fenster} + \dot{Q}_{Strahlung} + \dot{Q}_{Lüftung} + \dot{Q}_{Heizung} + \dots$$

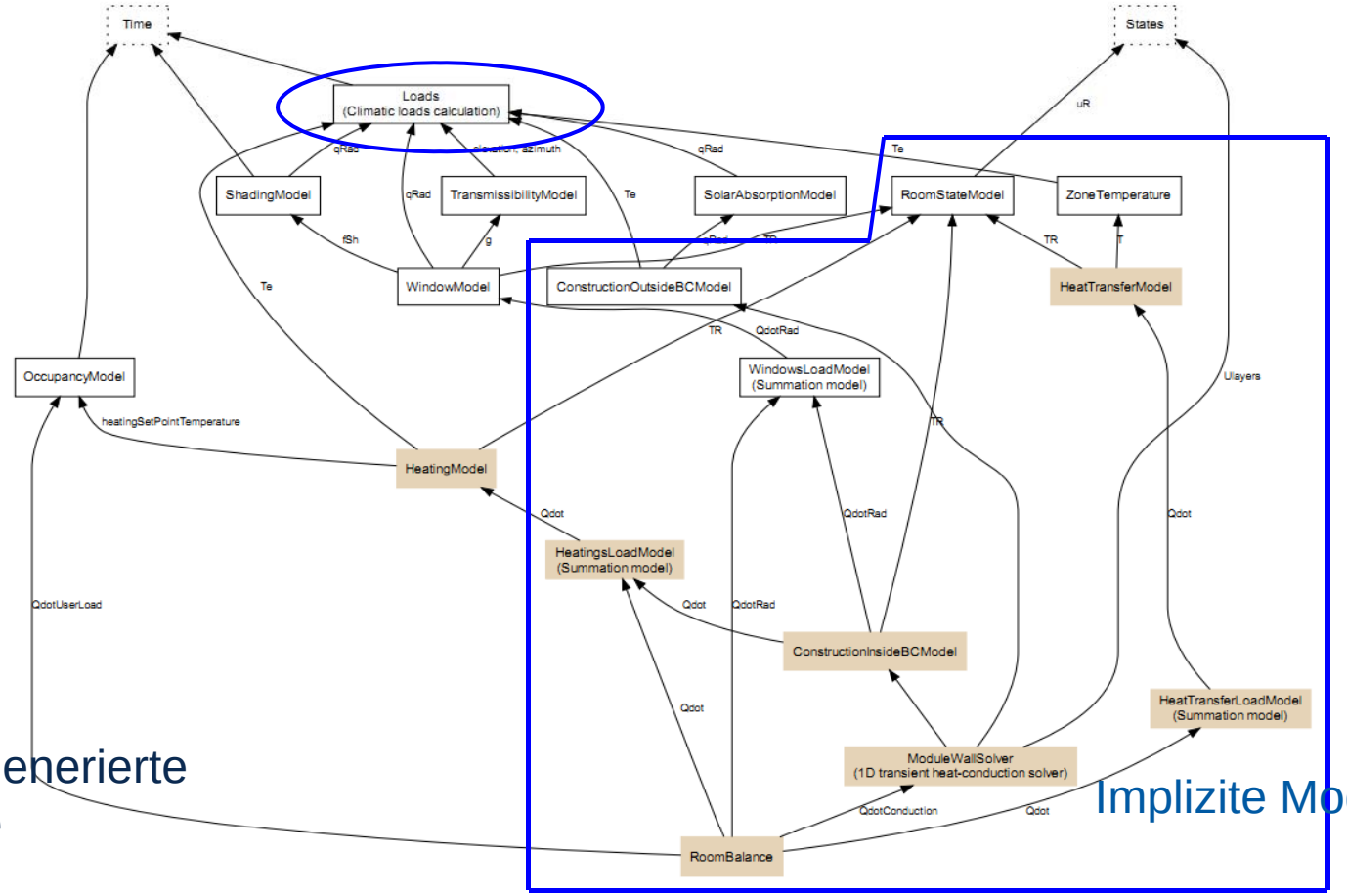
Modellauswertung



Modellauswertung



Modellauswertung



Nutzergenerierte Modelle

Implizite Modelle

Modellauswertung

Implizite Modelle

- Gebäudemodell
  - Wandtemperaturen, Raumtemperaturen, Wärmeströme auf Wandoberflächen, durch Fenster
- Anlagenkomponenten
  - Wärme- und Enthalpieströme durch Lüftung, Heizung, Verschattung
  - Basis-Modelle für Anlagenkomponenten

Nutzergenerierte Modelle

- Komplexe Anlagenmodelle, Verschattungs- und Lüftungssteuerung, ...

Implementierung nutzerdefinierter Modelle

Aus Modellschnittstelle:

- Initialisierung

setup()

- Aktualisierung mit neuem Zeitpunkt

setTime()

- Aktualisierung mit neuen Eingabewerten

update()

Eigene Modellgrößen:

Modellergebnis, Modellparameter, Modellabhängigkeiten

Implementierung nutzerdefinierter Modelle

Beispiel: Heizungsmodell

Modell-ergebnis

Parameter

Modellabhängig-keiten

```
class HeatingModelSimple : public GenericModel {
public:
    // ***KEYWORDLIST-START***
    enum Variables {
        V_Qdot                  'Heating load [W].'
    };
    enum Parameters {
        P_MaxPower              'Maximum power.'
        P_SetPointTemperature   'Set point temperature [K].
    };
    enum InputReferences {
        InputRef_Temperature,   'Room temperature [K].'
        NUM_InputRef
    };
    // ***KEYWORDLIST-END***
};
```

Implementierung nutzerdefinierter Modelle

Beispiel: Heizungsmodell

Daten aus anderen Modellen

Parameter

Übergabe Modellergebnis

```
void HeatingModelSimple::update() {

    const double RoomTemperature     = *inputValues()[InputRef_Temperature];

    const double SetpointTemperature = m_parameters[P_SetpointTemperature].value;
    const double MaxPower            = m_parameters[P_MaxPower].value;

    // compute heating load
    double Qdot = 0.0;
    if (RoomTemperature < SetpointTemperature)
        Qdot = MaxPower;

    m_variables[V_Qdot].value = Qdot;

}
```

Zustandsbasierte Modellauswertung

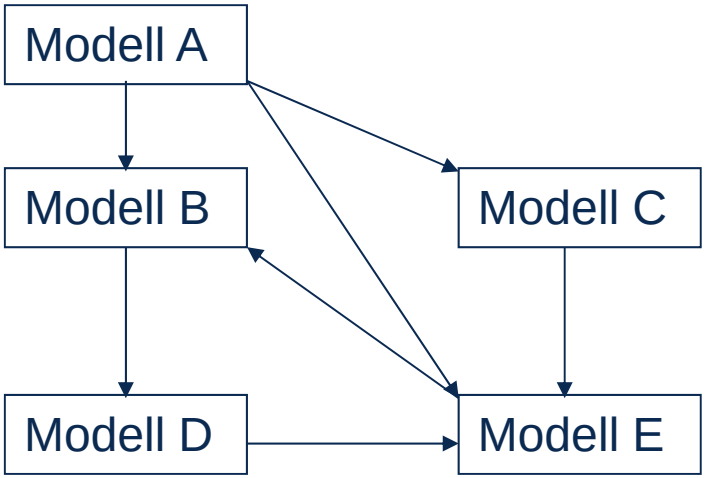
Modellobjekte

- halten Ergebniszustände der Modelle
- sind über die Modellgrenzen voneinander abhängig
- sind zum Teil erst zur Laufzeit bekannt

Problem: Aktualisierung und Auswertung

- iterativ → instabil
- gekoppelt → zeitaufwändig

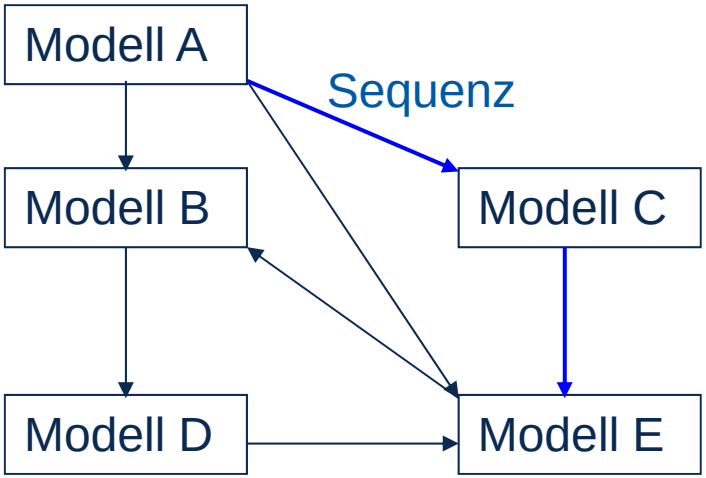
Bestimmung der Auswertungsreihenfolge



Modellbaum

- jedes Modellobjekt entspricht einem Baumknoten
- Auswertung durch Graphenalgorithmen

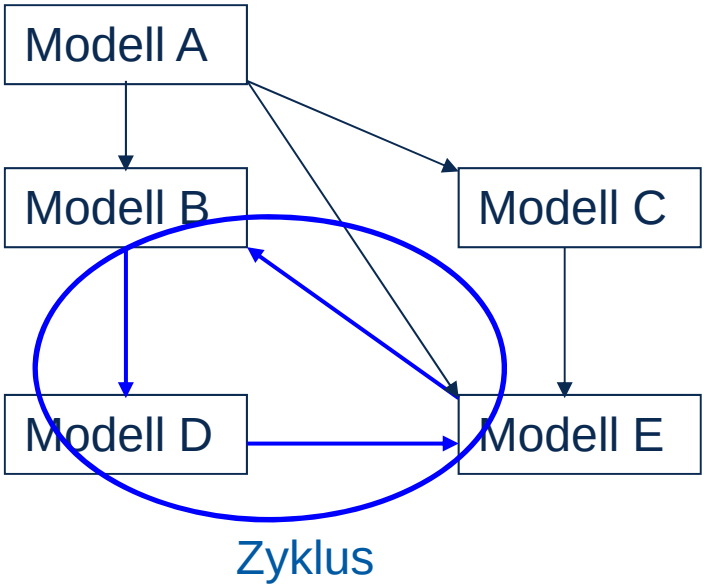
Bestimmung der Auswertungsreihenfolge



Modellbaum

- jedes Modellobjekt entspricht einem Baumknoten
- Auswertung durch Graphenalgorithmen
- Auswertungs-reihenfolge
  - Sequentiell

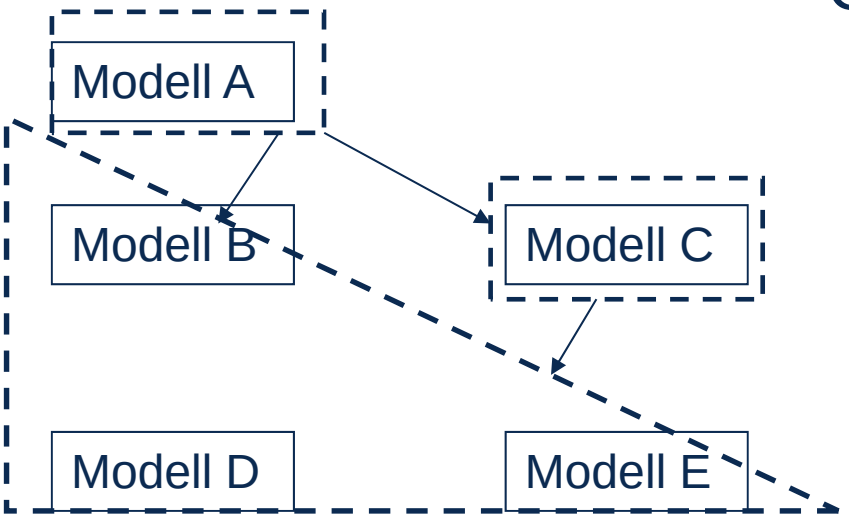
Bestimmung der Auswertungsreihenfolge



Modellbaum

- jedes Modellobjekt entspricht einem Baumknoten
- Auswertung durch Graphenalgorithmen
- Auswertungs-reihenfolge
  - Sequentiell
  - Zyklisch

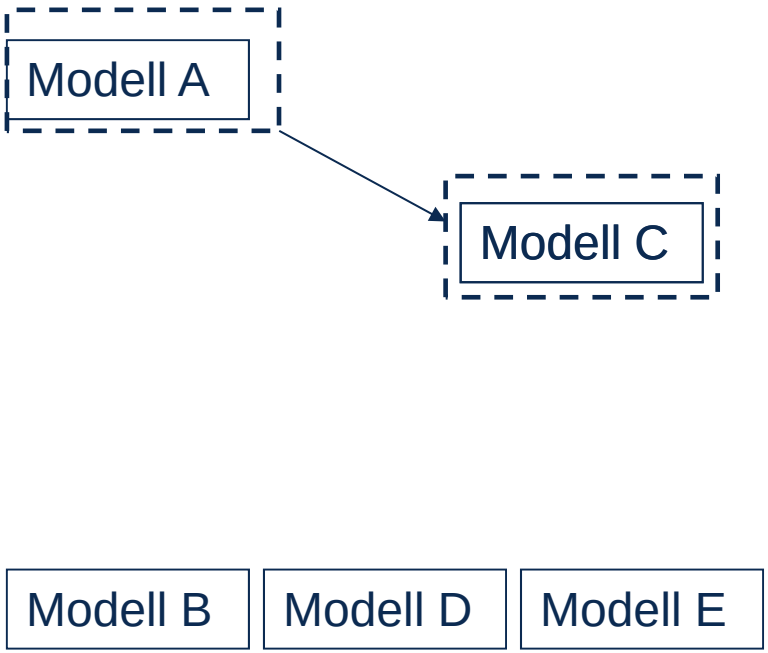
Bestimmung der Auswertungsreihenfolge



Gruppierung

- Ordnung von Zyklen und Sequenzen zu Gruppenobjekten

Bestimmung der Auswertungsreihenfolge



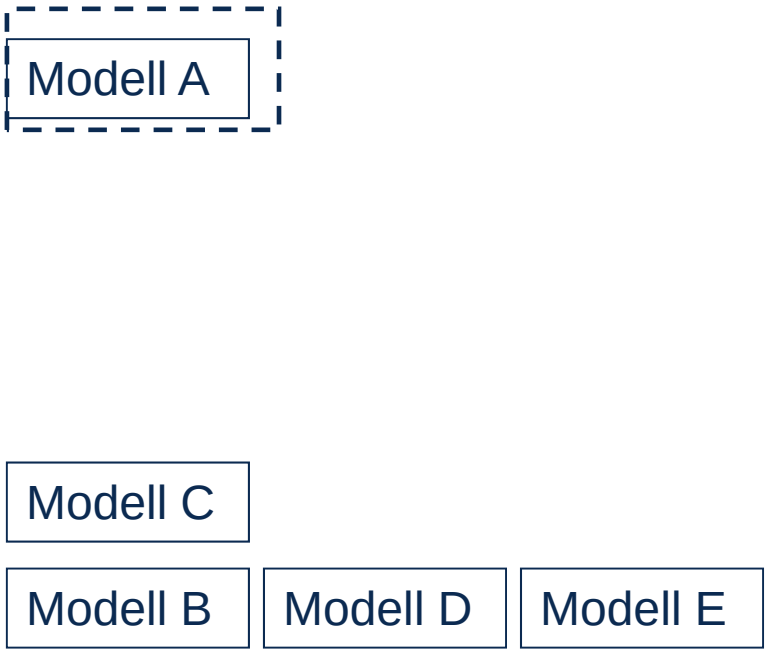
Gruppierung

- Ordnung von Zyklen und Sequenzen zu Gruppenobjekten

Stapel

- Sammeln aller Endknoten und schrittweise Reduzierung des Modellbaumes

Bestimmung der Auswertungsreihenfolge



Gruppierung

- Ordnung von Zyklen und Sequenzen zu Gruppenobjekten

Stapel

- Sammeln aller Endknoten und schrittweise Reduzierung des Modellbaumes

Bestimmung der Auswertungsreihenfolge

Gruppierung

- Ordnung von Zyklen und Sequenzen zu Gruppenobjekten

Stapel

- Sammeln aller Endknoten und schrittweise Reduzierung des Modellbaumes

Bestimmung der Auswertungsreihenfolge

Minimale Mehrfachauswertung

- Sequentielle Modellobjekte berechnen nur einmal
- Zyklen lösen gekoppelte nichtlineare Systeme

Parallelisierbarkeit

- gleichzeitige Endknoten können parallel ausgewertet werden

Nichtlineare Modellobjekte

- Behandlung von Zyklen durch stabile nichtlineare Lösungsverfahren (Newton-Raphson)

TECHNISCHE UNIVERSITÄT DRESDEN

Optimierung der Berechnungs-geschwindigkeit

Matrixstruktur

Newton-Verfahren

$$F\left(t, y, \dot{y}\right) = \dot{y} - f\left(t, y\right)$$

Matrixstruktur

Newton-Verfahren

$$F\left(t, y, \dot{y}\right) = \dot{y} - f\left(t, y\right)$$

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

Matrixstruktur

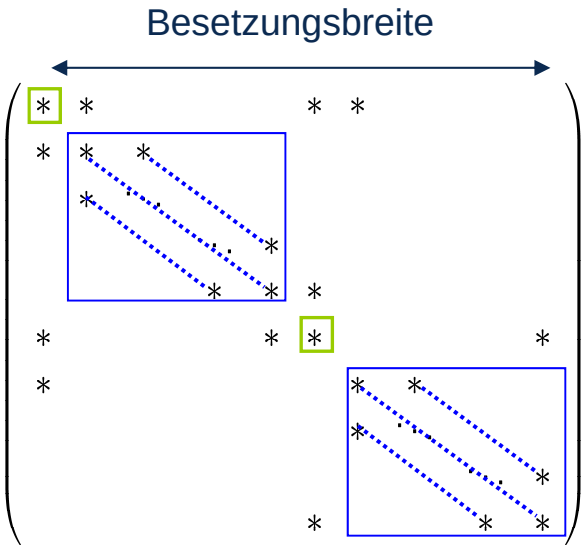
Newton-Verfahren

$$F\left(t, y, \dot{y}\right) = \dot{y} - f\left(t, y\right)$$

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

Jacobi-Matrix

Matrixstruktur



Raumtemperaturen

Wandtemperaturen

Kopplungsmatrix

- Wärmeströme zwischen Raum und Wand
- Direkte Wärmeströme zwischen Räumen (Verbindung durch Heizungsrohre, …)

Verfahren zur Lösung des Gleichungssystems

- Direkte Verfahren: hoher Rechenaufwand für breite Besetzung
- Iterative Verfahren: geeignet für schwachbesetzte Matrizen

→ Iterative Gleichungslöser mit geeigneter Vorkonditionierung (Bandvorkonditionierer, ILU)

Zusammenfassung

Neuentwicklungen

- Erhöhung der Ergebnisgenauigkeit durch detailgetreue Wandmodellierung
- Kopplungskonzept zwischen Gebäudemodell und Anlagenmodellen durch implizite und nutzerdefinierte Modellobjekte
- parallelisierte Auswertung einzelner Modellteilergebnisse durch graphenbasierte Algorithmen
- Integration in eine übergeordnete numerische Zeitintegrationsplattform
- strukturoptimierte Lösungsalgorithmen für die Differentialgleichungen
- Fokus hin zur schnellen Lösung großer Systeme