TU Clausthal

EnInnov2026

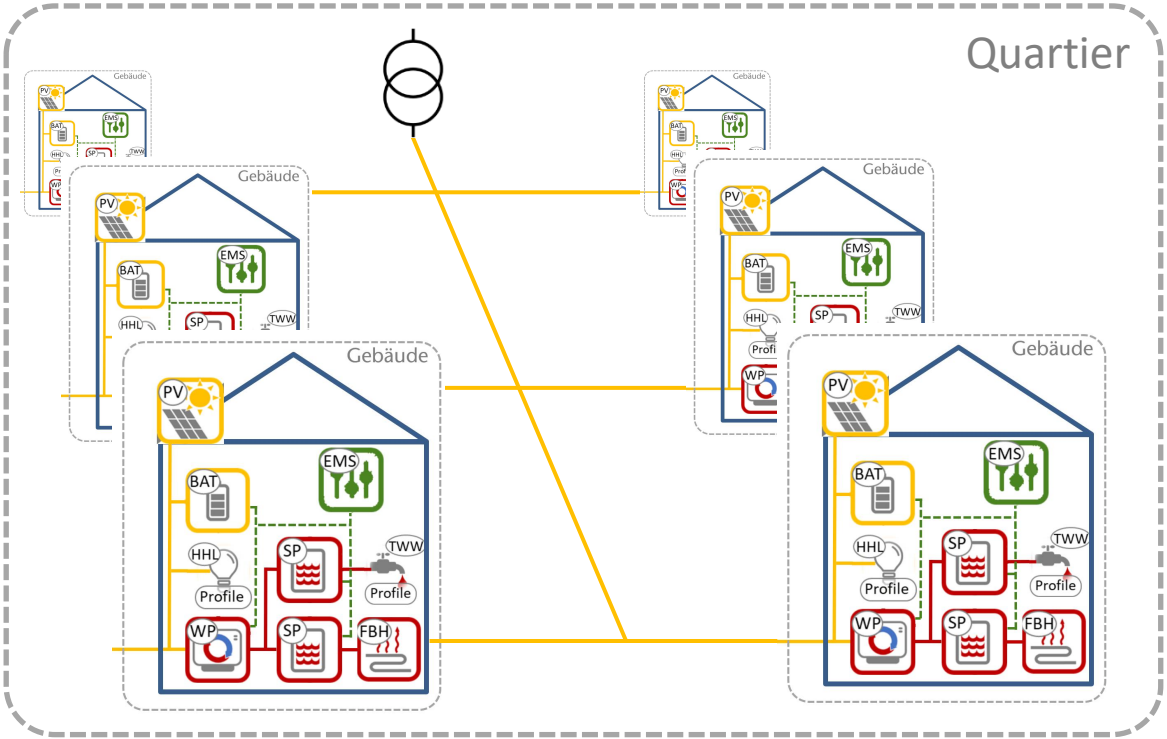
Numerisch stabiles thermisches Gebäudemodell für die elektrische Simulation von Niederspannungsnetzen

Hannes Hanse, M. Sc.
hannes.hanse@tu-clausthal.de

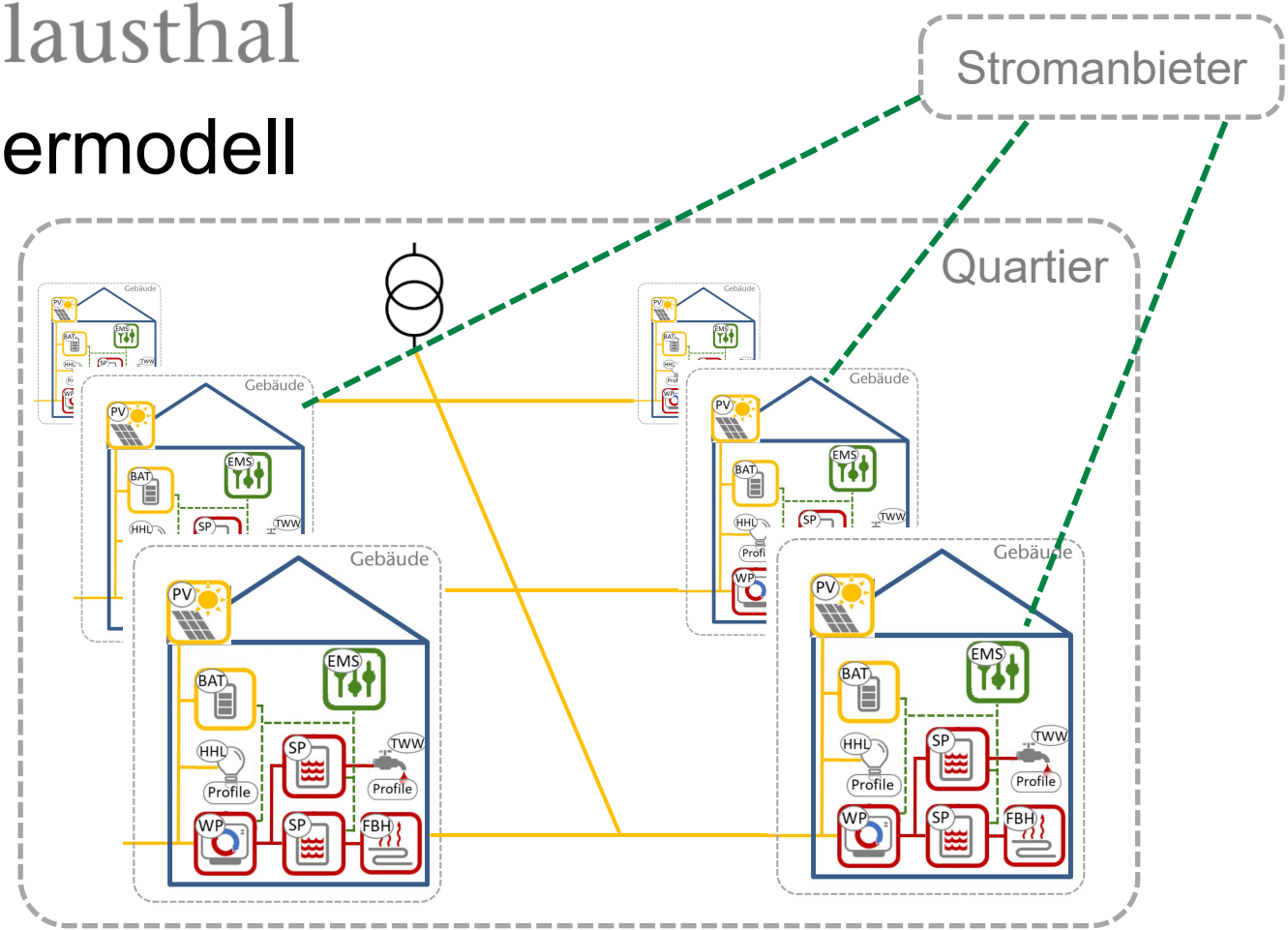
Prof. Dr.-Ing. Ines Hauer
ines.hauer@tu-clausthal.de

1. MOTIVATION

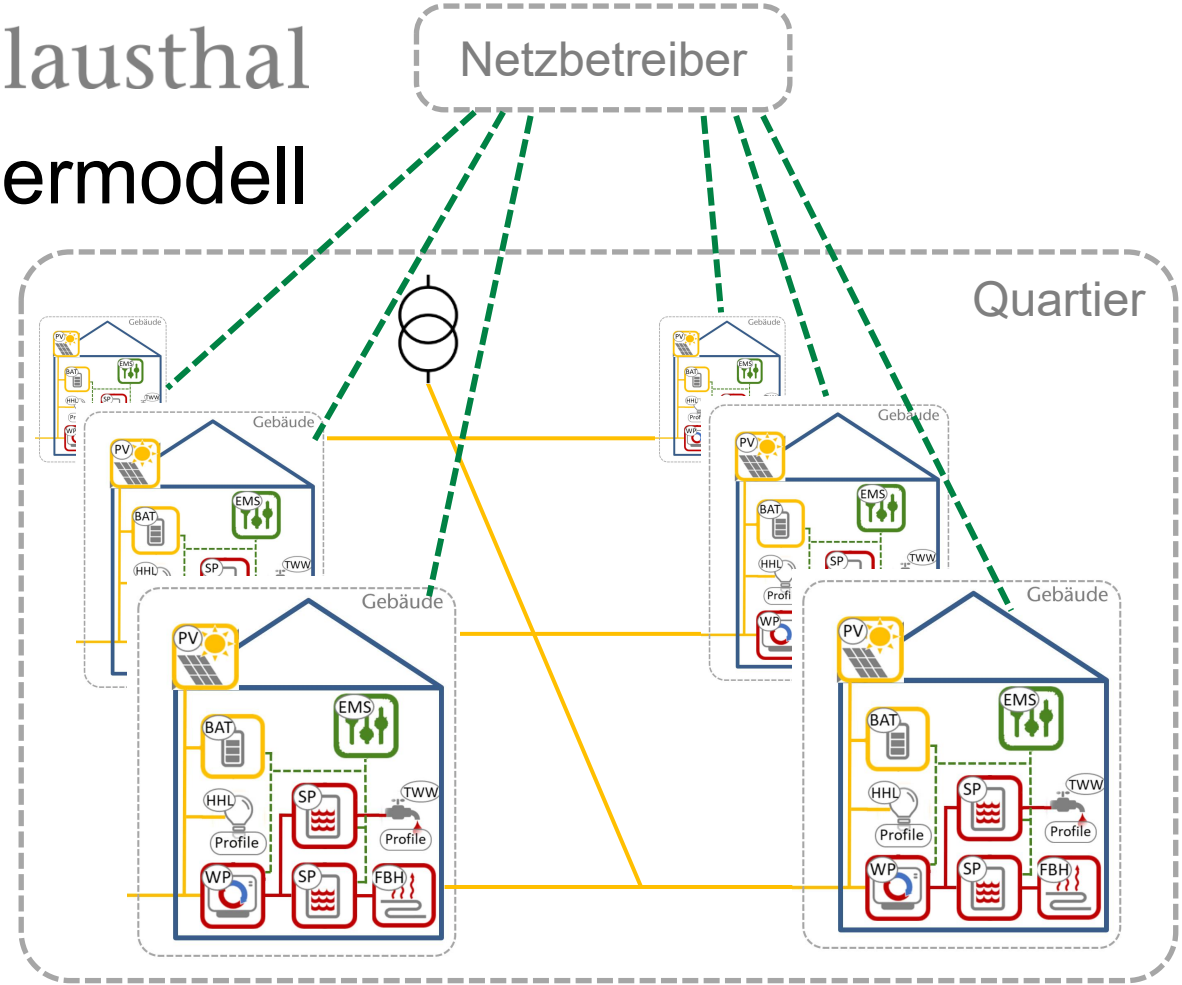
Quartiermodell



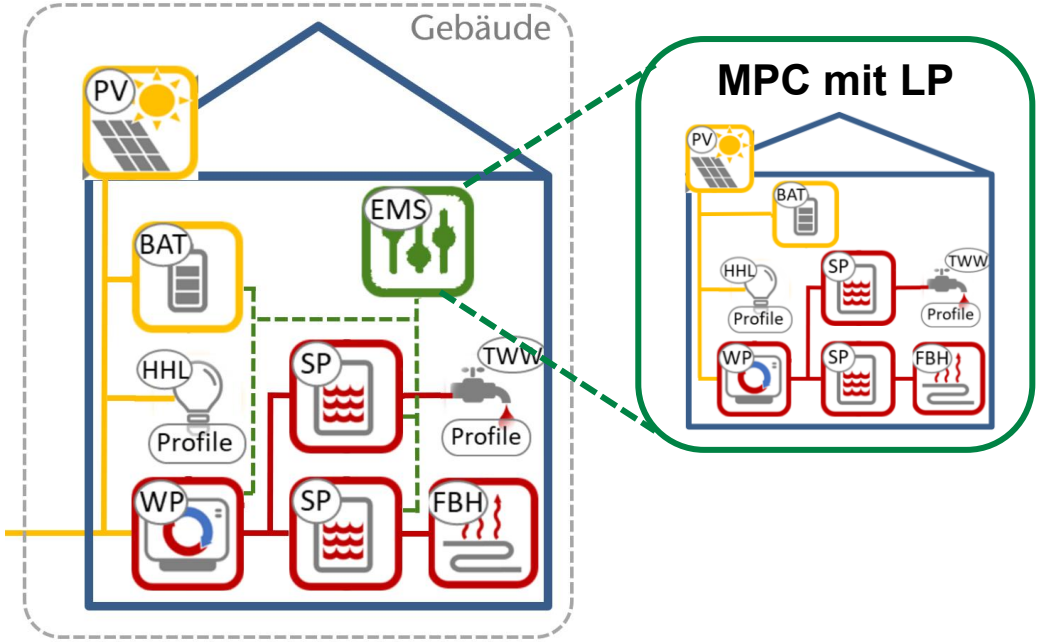
Quartiermodell



Quartiermodell

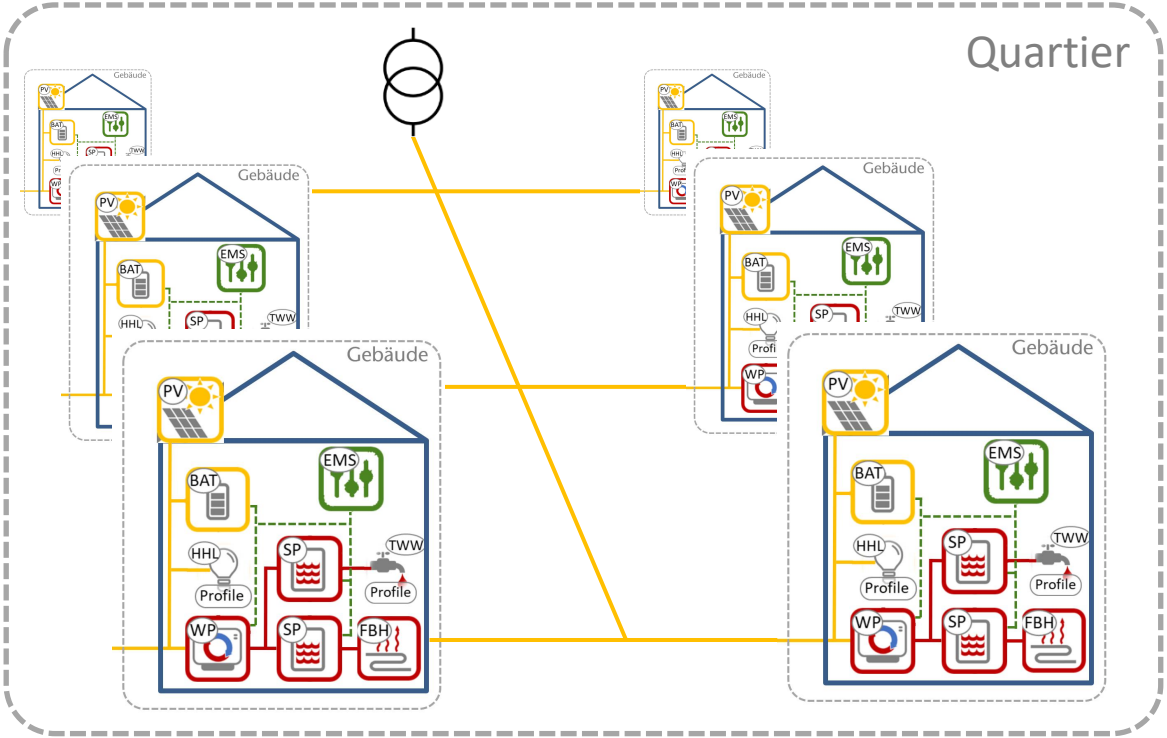


Gebäudemodell



- Zeitreihensimulation
- Thermisch & elektrisch
- Modellprädiktive Regelung im EMS

Quartiermodell



Quartiermodell

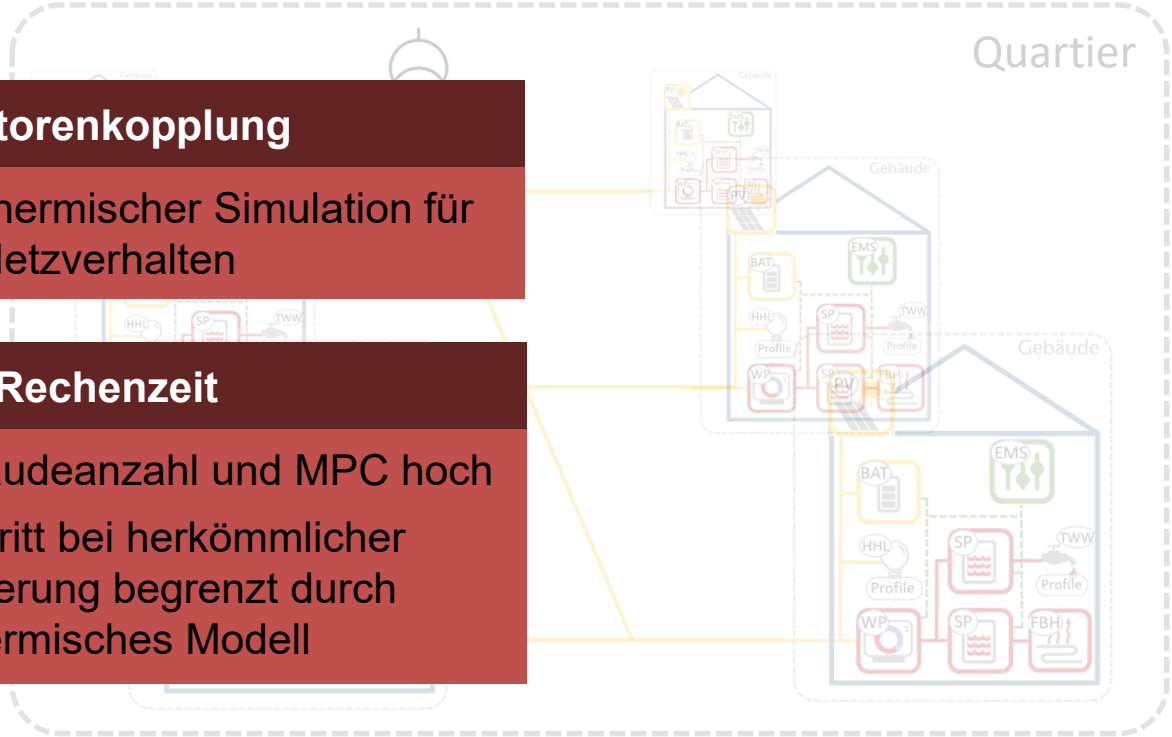
Sektorenkopplung

→ Relevanz thermischer Simulation für Netzverhalten

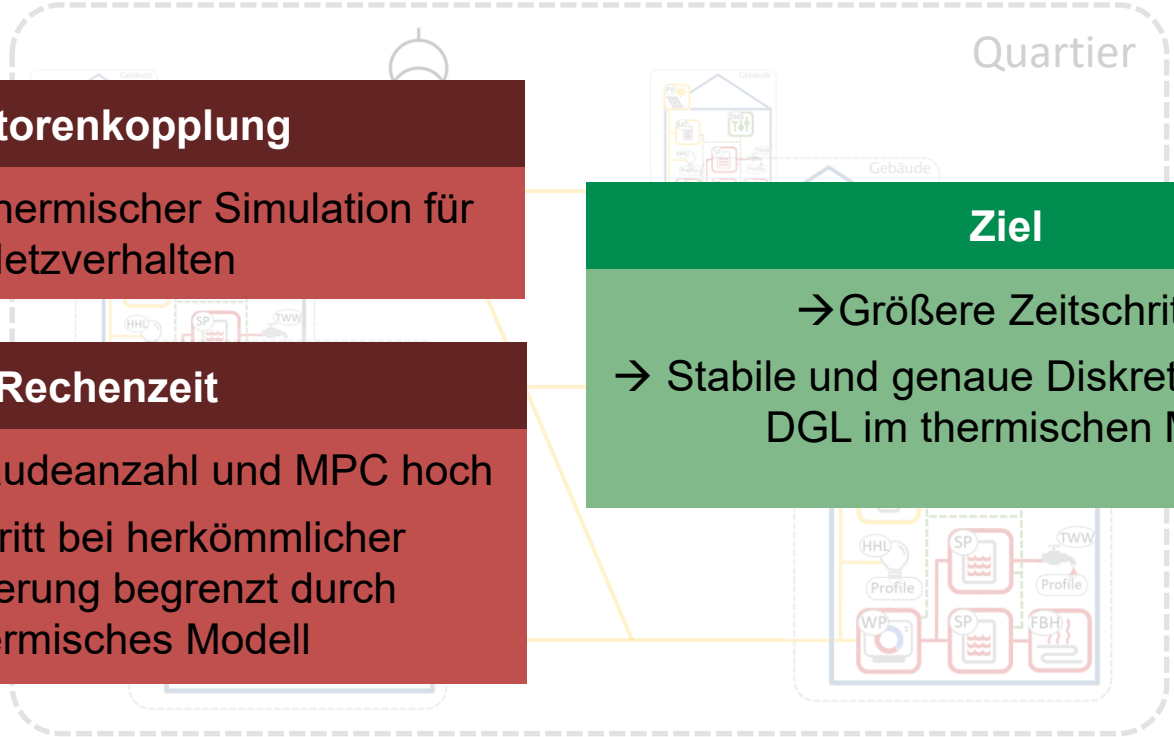
Rechenzeit

→ Durch Gebäudeanzahl und MPC hoch

→ Zeitschritt bei herkömmlicher Modellierung begrenzt durch thermisches Modell



Quartiermodell



Sektorenkopplung

→ Relevanz thermischer Simulation für Netzverhalten

Rechenzeit

→ Durch Gebäudeanzahl und MPC hoch

→ Zeitschritt bei herkömmlicher Modellierung begrenzt durch thermisches Modell

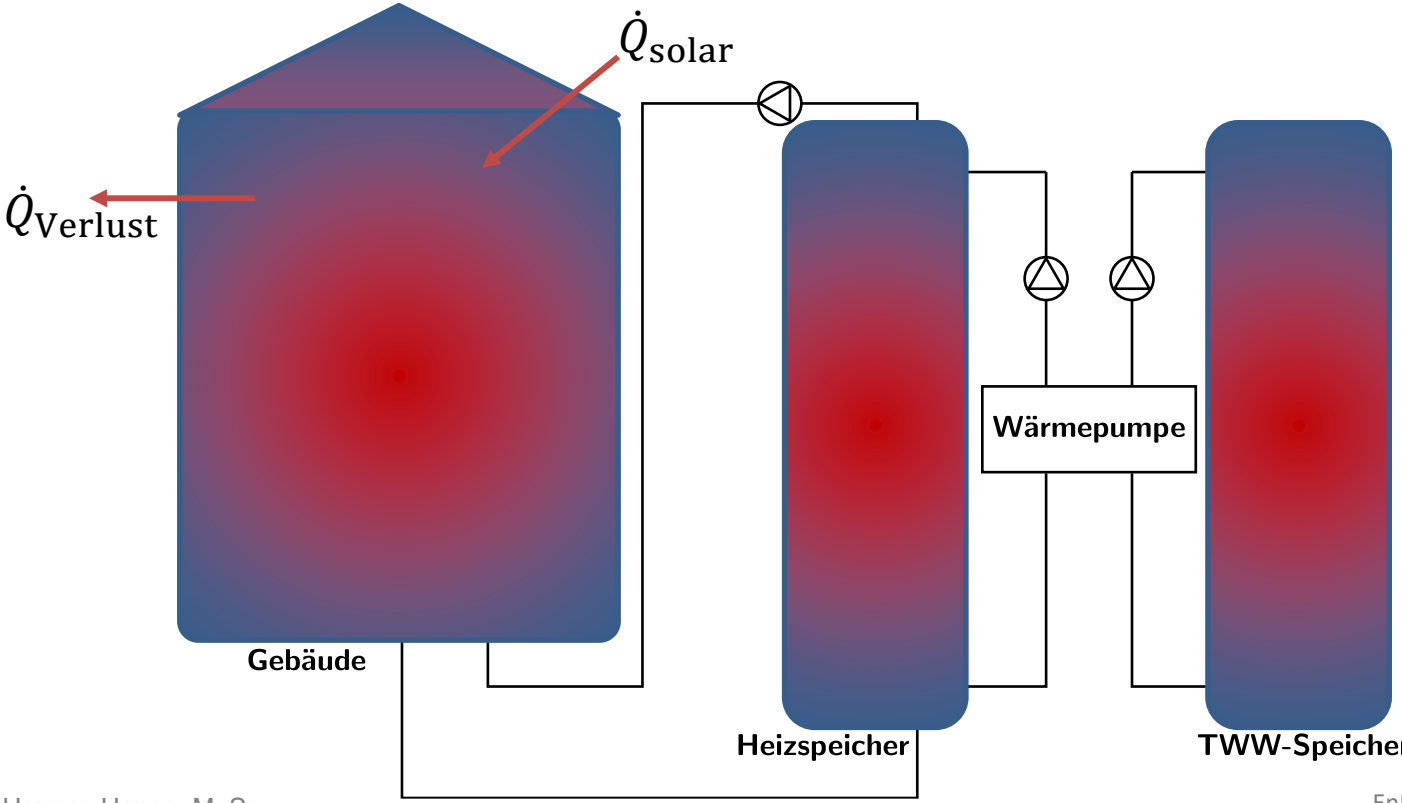
Ziel

→Größere Zeitschritte

→ Stabile und genaue Diskretisierung der DGL im thermischen Modell

2. THERMISCHES MODELL

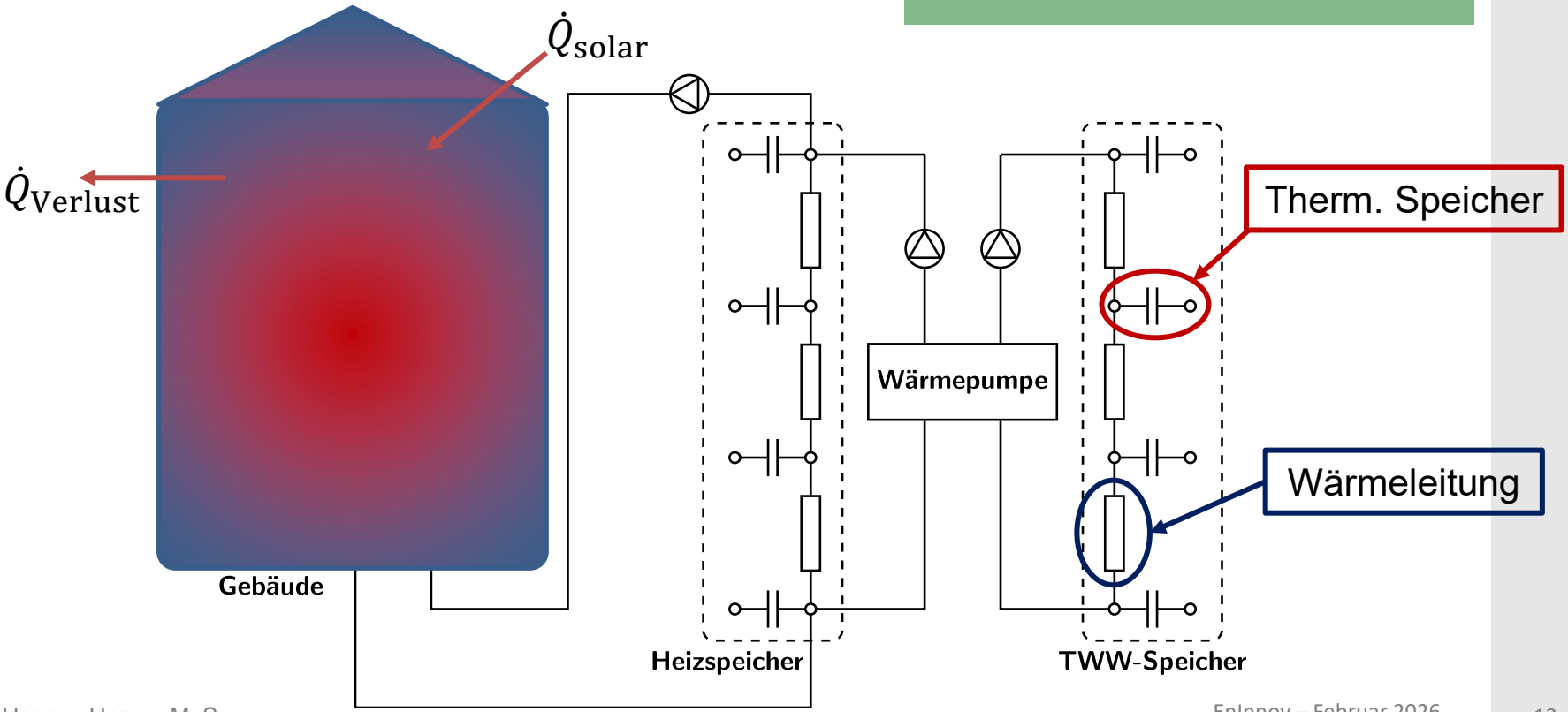
Thermisches Modell



Thermisches Modell

RC-Modell

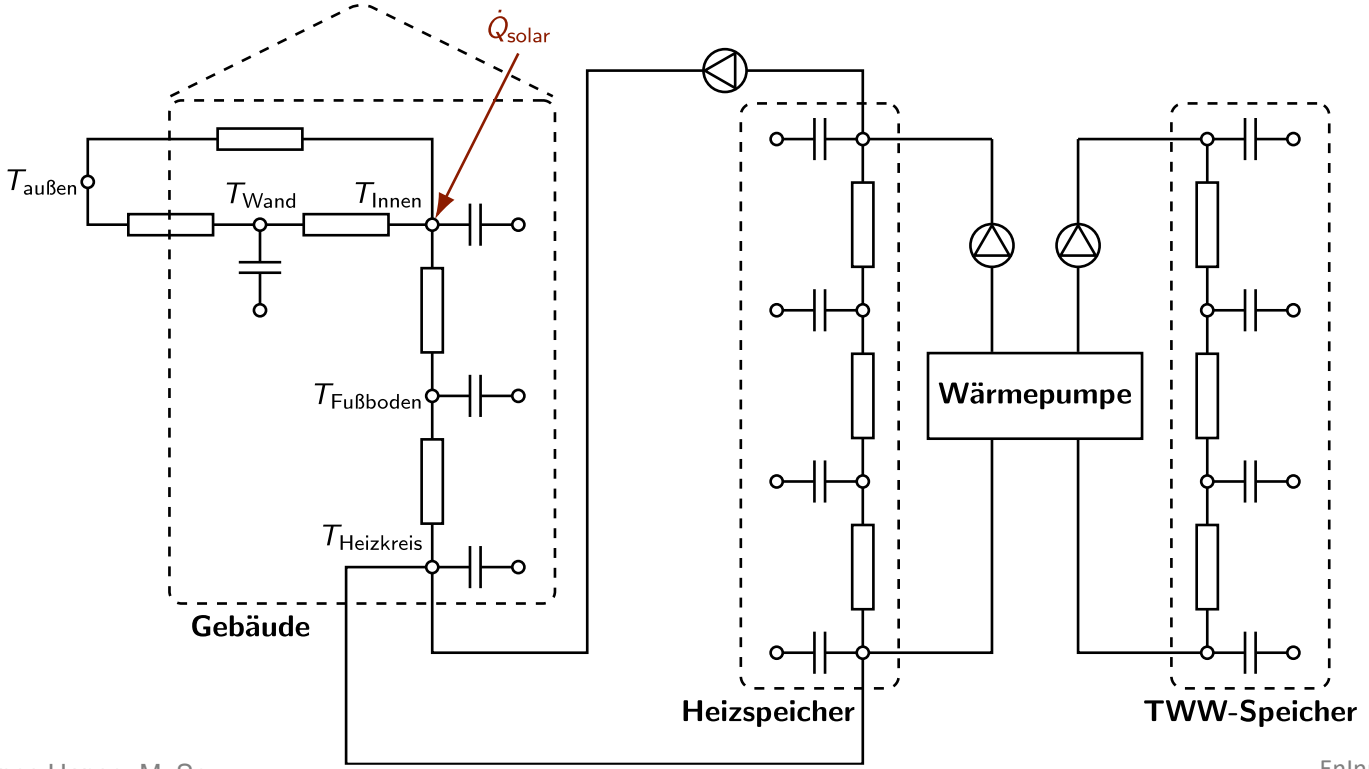
- Berücksichtigt Trägheit
- Vertretbarer Rechenaufwand



Thermisches Modell

RC-Modell

- Berücksichtigt Trägheit
- Vertretbarer Rechenaufwand



Zeitliche Diskretisierung

	DGL	VDI 6007	Expl.	Impl.	CN
Leitung	$\frac{\mathrm{d}Q_{1\to2}}{\mathrm{d}t} = (T_1 - T_2)\, Y_{\mathrm{cond}}$				
Konvektion	$\frac{\mathrm{d}Q_{1\to2}}{\mathrm{d}t} = (T_1 - T_2)\, \dot{m}\, c_{\mathrm{p}}$				

Zeitliche Diskretisierung

	DGL	VDI 6007	Expl.	Impl.	CN
Leitung	$\frac{\mathrm{d}Q_{1\to2}}{\mathrm{d}t} = (T_1 - T_2)\, Y_{\mathrm{cond}}$	▪ Übertragungsfunktionsmethode (TFM) ▪ zeitinvariant			
Konvektion	$\frac{\mathrm{d}Q_{1\to2}}{\mathrm{d}t} = (T_1 - T_2)\, \dot{m}\, c_{\mathrm{p}}$				

Zeitliche Diskretisierung

	DGL	VDI 6007	Expl.	Impl.	CN
Leitung	$\frac{\mathrm{d}Q_{1\to2}}{\mathrm{d}t} = (T_1 - T_2)\, Y_{\mathrm{cond}}$	▪ Übertragungsfunktionsmethode (TFM) ▪ zeitinvariant System ist zeitvariant → Methode ungeeignet			
Konvektion	$\frac{\mathrm{d}Q_{1\to2}}{\mathrm{d}t} = (T_1 - T_2)\, \dot{m}\, c_{\mathrm{p}}$				

Zeitliche Diskretisierung

	DGL	VDI	Explizites Euler-Verfahren	Impl.	CN
Leitung	$\frac{\mathrm{d}Q_{1\to2}}{\mathrm{d}t} = (T_1 - T_2)\, Y_{\mathrm{cond}}$		$Q_{1\to2,t} = \left(T_{1,t-1} - T_{2,t-1}\right) Y_{\mathrm{cond}}$		
Konvektion	$\frac{\mathrm{d}Q_{1\to2}}{\mathrm{d}t} = (T_1 - T_2)\, \dot{m}\, c_{\mathrm{p}}$		$Q_{1\to2,t} = (T_{1,t-1} - T_{2,t-1})\, \dot{m}\, c_{\mathrm{p}}$		

Zeitliche Diskretisierung

	DGL	VDI	Explizites Euler-Verfahren	Impl.	CN
Leitung	$\frac{dQ_{1\to2}}{dt} = (T_1 - T_2) \cdot Y_{cond}$				
Konvektion	$\frac{dQ_{1\to2}}{dt} = (T_1 - T_2)\, \dot{m}\, c_p$		Numerisch instabil bei $\Delta t > 5$ min → Methode ungeeignet		

```
# Einspeisung Wärmepumpe
self.qPkt_wp = mPkt_wp * self.cp_fluid * (temp_wp_out - self.temp_u) / 3.6
qPkt_vPkt_wp_o = mPkt_wp * self.cp_fluid * (temp_wp_out - self.temp_o) / 3.6
qPkt_vPkt_wp_mo = mPkt_wp * self.cp_fluid * (self.temp_o - self.temp_mo) / 3.6
qPkt_vPkt_wp_mu = mPkt_wp * self.cp_fluid * (self.temp_mo - self.temp_mu) / 3.6
qPkt_vPkt_wp_u = mPkt_wp * self.cp_fluid * (self.temp_mu - self.temp_u) / 3.6
```

Zeitliche Diskretisierung

	DGL	VDI	Expl.	ISO 52016 / Impl. Euler-Verfahren	CN
Leitung	$\frac{\mathrm{d}Q_{1\to2}}{\mathrm{d}t} = (T_1 - T_2)\, Y_{\mathrm{cond}}$			$Q_{1\to2,t} = \left(T_{1,t} - T_{2,t}\right) Y_{\mathrm{cond}}$	
Konvektion	$\frac{\mathrm{d}Q_{1\to2}}{\mathrm{d}t} = (T_1 - T_2)\, \dot{m}\, c_{\mathrm{p}}$			$Q_{1\to2,t} = \left(T_{1,t} - T_{2,t}\right) \dot{m}\, c_{\mathrm{p}}$	

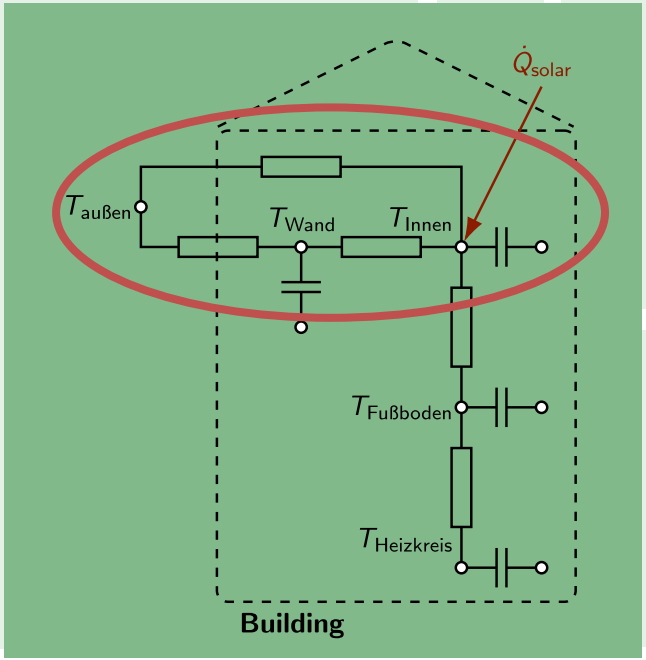
Zeitliche Diskretisierung

	DGL	VDI	Expl.	ISO 52016 / Impl. Euler-Verfahren	CN
Leitung	$\frac{\mathrm{d}\boldsymbol{Q}}{\mathrm{d}t} = \boldsymbol{M} \cdot \boldsymbol{Q} + \boldsymbol{b}$			$\frac{\boldsymbol{Q}_t - \boldsymbol{Q}_{t-1}}{\Delta t} = \boldsymbol{M}\boldsymbol{Q}_t + \boldsymbol{b}_t$	
Konvektion					

Zeitliche Diskretisierung

	DGL	VDI	Expl.	ISO 52016 / Impl. Euler-Verfahren	CN
Leitung	$\frac{\mathrm{d}Q}{\mathrm{d}t} = M \cdot Q + b$			Hohe Ungenauigkeit bei $\Delta t > 5\ \mathrm{min}$ → Methode ungeeignet	
Konvektion					

Zeitliche Diskretisierung

	DGL	VDI	Expl.	ISO 52016 / Impl. Euler-Verfahren	CN
Leitung				Hohe Ungenauigkeit bei $\Delta t > 5\ \mathrm{min}$ → Methode ungeeignet	
Konvektion					

Zeitliche Diskretisierung

	DGL	VDI	Expl.	Impl.	Crank-Nicolson
Leitung	$\frac{\mathrm{d}\boldsymbol{Q}}{\mathrm{d}t} = \boldsymbol{M} \cdot \boldsymbol{Q} + \boldsymbol{b}$				$\frac{\boldsymbol{Q}_t - \boldsymbol{Q}_{t-1}}{\Delta t} = \frac{1}{2}\boldsymbol{M}(\boldsymbol{Q}_{t-1} + \boldsymbol{Q}_t) + \frac{1}{2}(\boldsymbol{b}_{t-1} + \boldsymbol{b}_t)$
Konvektion					

Zeitliche Diskretisierung

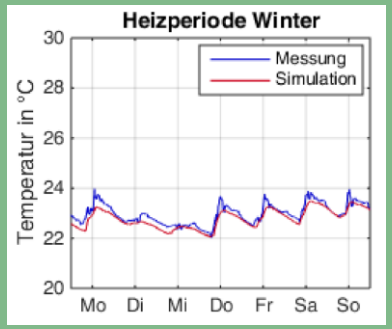
	DGL	VDI	Expl.	Impl.	Crank-Nicolson
Leitung	→ Numerisch stabil → Genauer impl. Euler				$\frac{\boldsymbol{Q}_t - \boldsymbol{Q}_{t-1}}{\Delta t} = \frac{1}{2}\boldsymbol{M}(\boldsymbol{Q}_{t-1} + \boldsymbol{Q}_t) + \frac{1}{2}(\boldsymbol{b}_{t-1} + \boldsymbol{b}_t)$
Konvektion	$\frac{\mathrm{d}\boldsymbol{Q}}{\mathrm{d}t} = \boldsymbol{M} \cdot \boldsymbol{Q} + \boldsymbol{b}$				
	Aufwendige Matrix-Implementierung				

3. MODELLVALIDIERUNG

Modellparametrierung & -validierung

Parametrierung

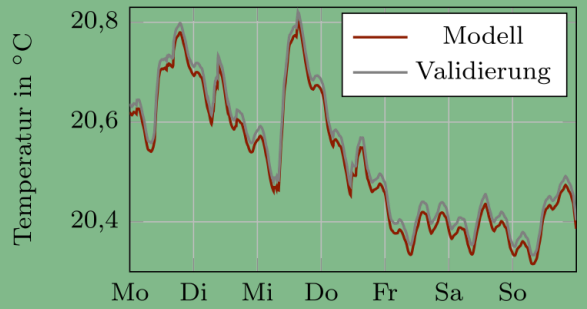
- R & C für Gebäudemodell in vorherigem Projekt an Messdaten parametriert



[1]

Validierung

- Vergleich des neuen Modells bei zeitlich hoher Auflösung mit Modell aus vorherigem Projekt

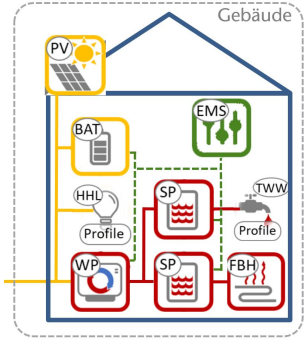


[1: E. Schneider et al.: Entwicklung und messwertbasierte Parametrierung eines vereinfachten Gebäudemodells für die Quartiersmodellierung, 2019]

4. ERGEBNISSE

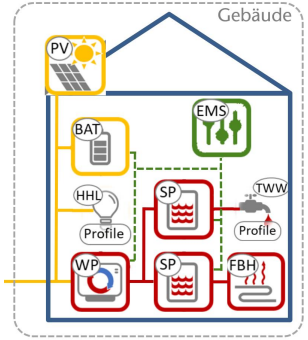
Auswertung

- 1 beispielhaftes Gebäude
- Simulation: Januar 2019

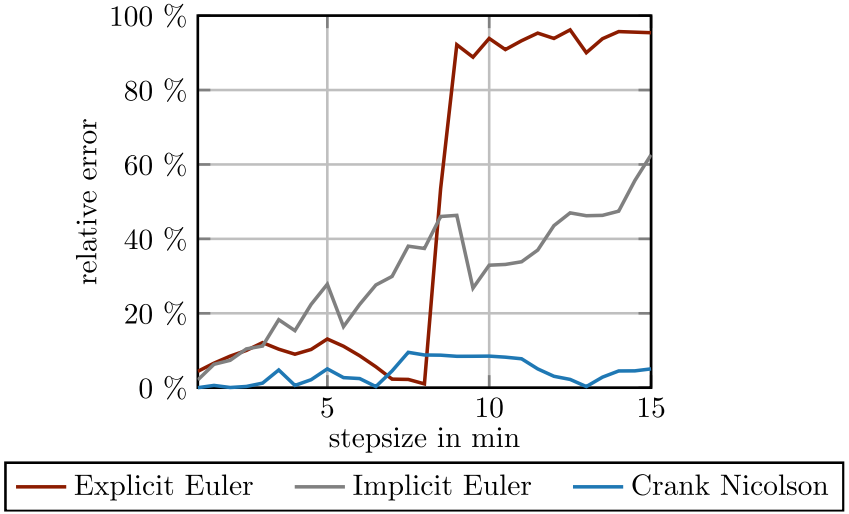


4. ERGEBNISSE

Relativer Modellfehler

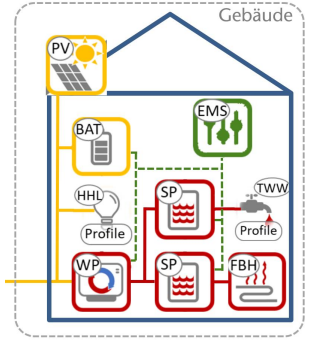


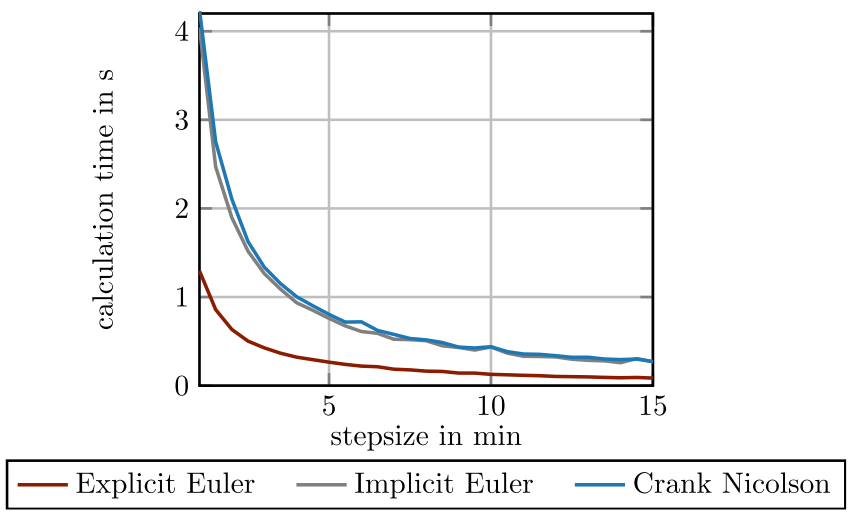
- Expl. Euler ab $\Delta t > 7$ min unbrauchbar
- Großer Modellfehler beim impl. Euler
- Crank-Nicolson:
  - Modellfehler < 5 % bei $\Delta t = 15$ min



Rechenzeit

- Höhere Rechenzeit für CN & impl. Euler bei gleichem Δt
- CN bei $\Delta t = 15$ min ca. 80 % schneller als expl. Euler bei $\Delta t = 1$ min





Großer Zeitgewinn durch größeres Δt in anderen Modellteilen

5. PYTHON-BIBLIOTHEK

$$
\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} Q_1 \\ \vdots \\ Q_i \\ \vdots \\ Q_n \end{bmatrix} = \underbrace{\begin{bmatrix} -\frac{Y_{\mathrm{cond},1,2}}{C_{\mathrm{p},1}} & \frac{Y_{\mathrm{cond},1,2}}{C_{\mathrm{p},2}} & & 0 & & \cdots & \\ & & & \vdots & & & \\ & \cdots & \frac{Y_{\mathrm{cond},i-1,i}}{C_{\mathrm{p},i-1}} & -\frac{Y_{\mathrm{cond},i-1,i}}{C_{\mathrm{p},i}} + \frac{Y_{\mathrm{cond},i,i+1}}{C_{\mathrm{p},i}} & -\frac{Y_{\mathrm{cond},i,i+1}}{C_{\mathrm{p},i+1}} & \cdots & \\ & & & \vdots & & & \\ & \cdots & & 0 & & -\frac{Y_{\mathrm{cond},n-1,n}}{C_{\mathrm{p},n-1}} & \frac{Y_{\mathrm{cond},n-1,n}}{C_{\mathrm{p},n}} \end{bmatrix}}_{\mathbf{M}_{\mathrm{hs,cond,internal}}} \cdot \begin{bmatrix} Q_1 \\ \vdots \\ Q_i \\ \vdots \\ Q_n \end{bmatrix}
$$

$$
+ \underbrace{\begin{bmatrix} -\frac{\dot{m}_{\mathrm{load}}+\dot{m}_{\mathrm{unload}}}{C_{\mathrm{p},1}} & \frac{\dot{m}_{\mathrm{load}}}{C_{\mathrm{p},2}} & & 0 & & \cdots & \\ & & & \vdots & & & \\ & \cdots & \frac{\dot{m}_{\mathrm{unload}}}{C_{\mathrm{p},i-1}} & -\frac{\dot{m}_{\mathrm{load}}+\dot{m}_{\mathrm{unload}}}{C_{\mathrm{p},i}} & \frac{\dot{m}_{\mathrm{load}}}{C_{\mathrm{p},i+1}} & \cdots & \\ & & & \vdots & & & \\ & \cdots & & 0 & & \frac{\dot{m}_{\mathrm{unload}}}{C_{\mathrm{p},n-1}} & -\frac{\dot{m}_{\mathrm{load}}+\dot{m}_{\mathrm{unload}}}{C_{\mathrm{p},n}} \end{bmatrix}}_{\mathbf{M}_{\mathrm{hs,conv,forced}}} \cdot c_{\mathrm{p,F}} \cdot \begin{bmatrix} Q_1 \\ \vdots \\ Q_i \\ \vdots \\ Q_n \end{bmatrix} + \underbrace{\begin{bmatrix} T_{\mathrm{wp,vl}} \cdot \dot{m}_{\mathrm{load}} \cdot c_{\mathrm{p,F}} \\ 0 \\ \vdots \\ 0 \\ T_{\mathrm{hkf}} \cdot \dot{m}_{\mathrm{unload}} \cdot c_{\mathrm{p,F}} \end{bmatrix}}_{\mathbf{b}_{\mathrm{hs,conv,forced}}}
$$

$$
+ \underbrace{\begin{bmatrix} -\frac{\dot{m}_{\mathrm{conv,free},1\to2}}{C_{\mathrm{p},1}} & \frac{\dot{m}_{\mathrm{conv,free},1\to2}}{C_{\mathrm{p},2}} & & 0 & & \cdots & \\ & & & \vdots & & & \\ & \cdots & \frac{\dot{m}_{\mathrm{conv,free},i-1\to i}}{C_{\mathrm{p},i-1}} & -\frac{\dot{m}_{\mathrm{conv,free},i-1\to i}+\dot{m}_{\mathrm{conv,free},i\to i+1}}{C_{\mathrm{p},i}} & \frac{\dot{m}_{\mathrm{conv,free},i\to i+1}}{C_{\mathrm{p},i+1}} & \cdots & \\ & & & \vdots & & & \\ & \cdots & & 0 & & \frac{\dot{m}_{\mathrm{conv,free},n-1\to n}}{C_{\mathrm{p},n-1}} & -\frac{\dot{m}_{\mathrm{conv,free},n-1\to n}}{C_{\mathrm{p},n}} \end{bmatrix}}_{\mathbf{M}_{\mathrm{hs,conv,free}}} \cdot c_{\mathrm{p,F}} \cdot \begin{bmatrix} Q_1 \\ \vdots \\ Q_i \\ \vdots \\ Q_n \end{bmatrix}
$$

$$
+ \underbrace{\begin{bmatrix} -\frac{A_{\mathrm{Sp,M},1}}{C_{\mathrm{p},1}\cdot R_{\mathrm{Sp,M},1}} & 0 & 0 \\ \vdots & & \\ 0 & -\frac{A_{\mathrm{Sp,M},i}}{C_{\mathrm{p},i}\cdot R_{\mathrm{Sp,M},i}} & 0 \\ \vdots & & \\ 0 & 0 & -\frac{A_{\mathrm{Sp,M},n}}{C_{\mathrm{p},n}\cdot R_{\mathrm{Sp,M},n}} \end{bmatrix}}_{\mathbf{M}_{\mathrm{hs,loss}}} \cdot \begin{bmatrix} Q_1 \\ \vdots \\ Q_i \\ \vdots \\ Q_n \end{bmatrix} + \underbrace{\begin{bmatrix} \frac{T_{\mathrm{U}}\cdot A_{\mathrm{Sp,M},1}}{R_{\mathrm{Sp,M},1}} \\ \vdots \\ \frac{T_{\mathrm{U}}\cdot A_{\mathrm{Sp,M},i}}{R_{\mathrm{Sp,M},i}} \\ \vdots \\ \frac{T_{\mathrm{U}}\cdot A_{\mathrm{Sp,M},n}}{R_{\mathrm{Sp,M},n}} \end{bmatrix}}_{\mathbf{b}_{\mathrm{hs,loss}}}
$$

ThermoBuilPy

- Definition beliebiger $x\mathrm{R}y\mathrm{C}$-Systeme über Definition der R & C
- Automatische:
  - Matrizenerstellung
  - Berechnung
  - Auswertung
- Deutlich geringere Fehleranfälligkeit
- Erkennbare physikalische Zusammenhänge → Einfaches Debugging
- Schnittstelle für (MI)LP-Optimierung

- Veröffentlichtes Python-Package
- Dokumentation auf GitHub

[https://pypi.org, https://brand.github.com/foundations/logo]

Vielen Dank für Ihre Aufmerksamkeit

Gefördert durch:

Bundesministerium
für Wirtschaft
und Energie

aufgrund eines Beschlusses
des Deutschen Bundestages

Diese Arbeit wurde im Rahmen des Vorhabens „**WQeff**“ (**FKZ 03EN3070E**) durch das Bundesministerium für Wirtschaft und Energie aufgrund eines Beschlusses des Deutschen Bundestages gefördert.
Die Autor:innen danken dem BMWE für die finanzielle Förderung.

FRAGEN?

ThermoBuilPy - Anwendung

1. Speicher definieren

```
st1=ThermalStorage.newStorage(cap=1000,temp=30)
st2=ThermalStorage.newStorage(cap=2000,temp=40)
```

2. Wärmeübertragung definieren

```
con1=Conduction(st1,st2,coeff=20)
```

3. Thermisches System definieren

```
thSys=ThermalSystem.newThermalSystem(
    storages=[st1, st2],
    conductions=[con1],)
```

4. Zeitreihe simulieren

```
thSys.simulate(num_steps=num_steps,stepsize=stepsize)
```

5. Auswertung

```
thSys.plot_temps([st1,st2])
```