TECHNISCHE UNIVERSITÄT DRESDEN

Zentrum für Informationsdienste und Hochleistungsrechnen – TU Dresden

# PARADOM

## *Parallele Algorithmische Differentiation in OpenModelica für energietechnische Simulationen und Optimierungen*

Martin Schroschk

7. HPC-Status-Konferenz der Gauß-Allianz
04. - 05.12.2017, HLRS Stuttgart

E-Mail: martin.schroschk@tu-dresden.de

ZIH
Center for Information Services & High Performance Computing

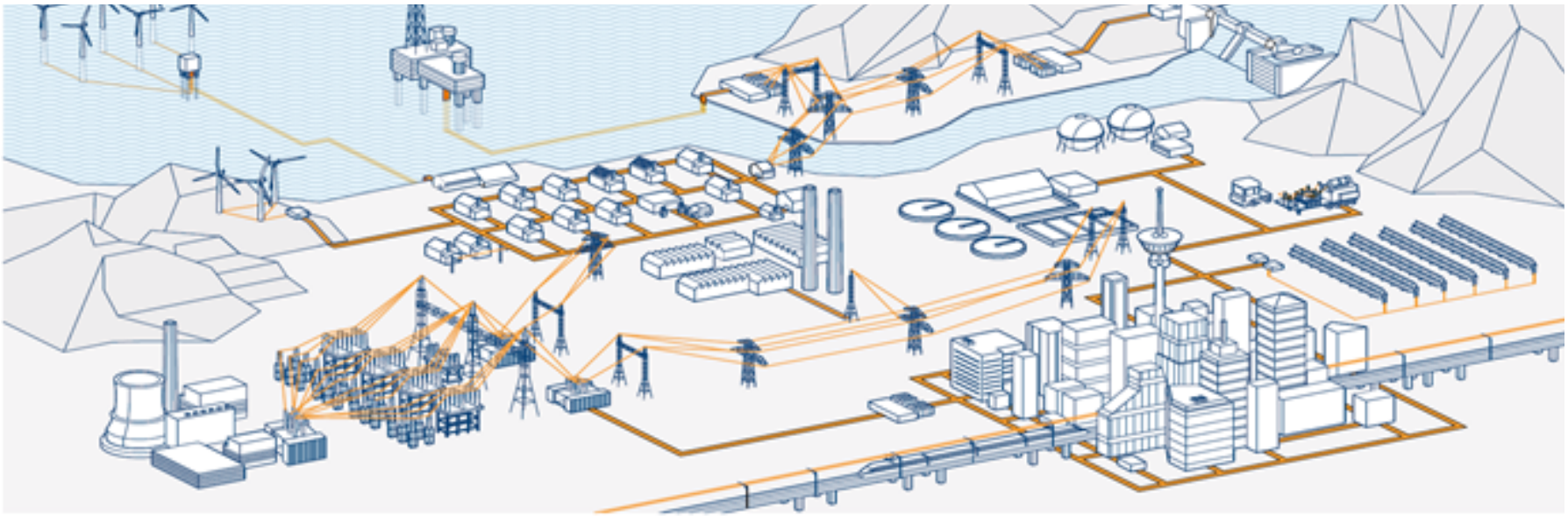
# Stromerzeugung - Früher

Zentrale Großkraftwerke mit vorbestimmten Einsatzplänen

Braunkohle-Kraftwerk bei Jänschwalde, guentherhh, CC BY 2.0

# Stromerzeugung - Heute und Morgen

- Sehr viele Kleinerzeuger (Technologien: Wind, Solar, Biogas, etc.)
- Konventionelle Kraftwerke sichern Grundbedarfe und Lastspitzen ab



©ABB

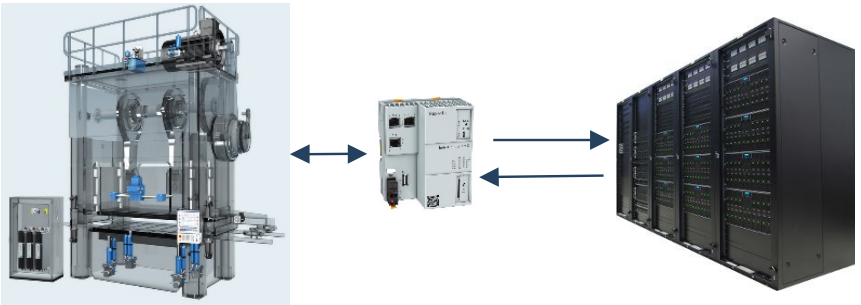
Herausforderungen

- Zusammenfassung und Steuerung der Kleinerzeuger in virtuellen Kraftwerken
- Echtzeitoptimierung aller Erzeuger, d. h. flexible Anpassung an Bedarfe

# Optimale Steuerung und Modellprädiktive Regelung

## Optimale Steuerung

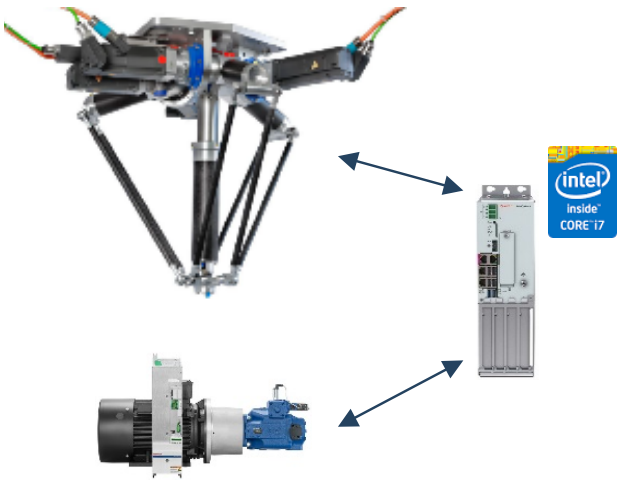
- Messdaten des Prozesses werden an HPC-System gesendet
- Controller triggert dynamische Optimierung
- Optimierung wird auf HPC-System ausgeführt
- Anpassung an geänderte Produktionsbedingungen



©Bosch Rexroth

## Modellprädiktive Regelung

- Vorausschauende Regelung auf Basis eines Modells
- Dynamisches Optimierungsproblem wird in (jedem Controllerschritt) gelöst
- Echtzeitanforderung
- Leistungsfähige Controller-Hardware




©Bosch Rexroth

# Motivation - Zusammenfassung

Aufgaben

- Modellbildung der (energietechnischen) Anlagen und deren Komponenten
- Simulation und Optimierung
  - Komponenten und Prozesse
  - Verhalten von Produkten in der Anwendung
- Online-Optimierung: Flexible Anpassung an Bedarfe und Zustände

Herausforderungen

- Rasch wachsende Systeme, d. h. immer größere und komplexere Modelle
- Grenzen der verfügbaren Optimierungstechnologien bald erreicht

# Konsortium

PARADOM

SIEMENS

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Rexroth
Bosch Group

ABB

FH Bielefeld
University of
Applied Sciences

LTX
Simulation GmbH

TECHNISCHE
UNIVERSITÄT
DRESDEN

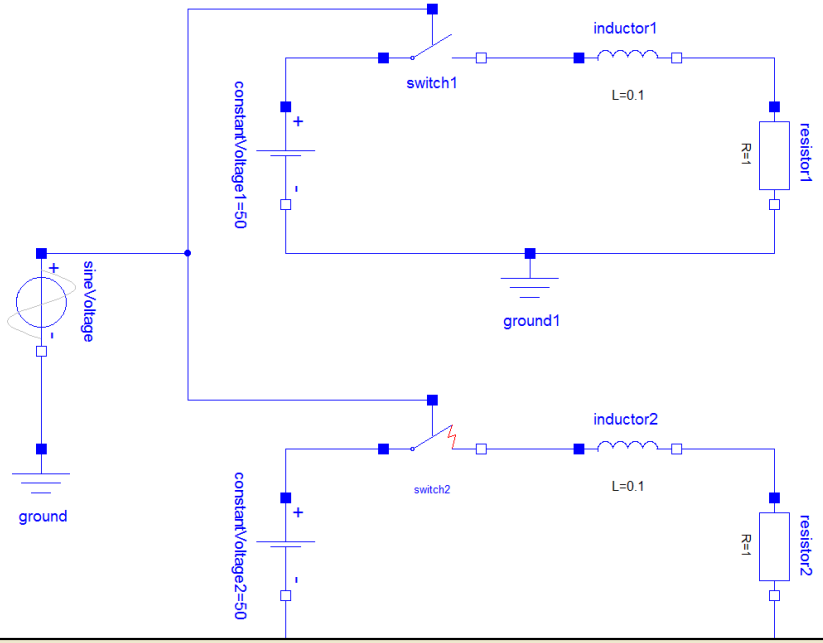
GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

# Modelica

Beschreibungssprache für technische Systeme

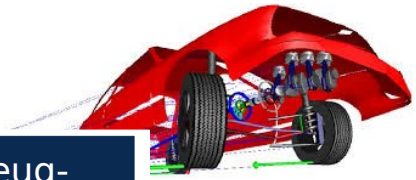
- frei verfügbar
- objektorientiert
- gleichungsbasiert
- diskrete und kontinuierliche Simulation
- freie und kommerzielle Bibliotheken
- Entwicklung durch gemeinnützigen Modelica Association




```
model Resistor "Ideal linear electrical resistor"
  parameter Modelica.SIunits.Resistance R(start=1);
  parameter Modelica.SIunits.Temperature T_ref=300.15;
  extends Modelica.Electrical.Analog.Interface.OnePort;
equation
  v = R_actual*i;
  LossPower = v*i;
end Resitor;
```

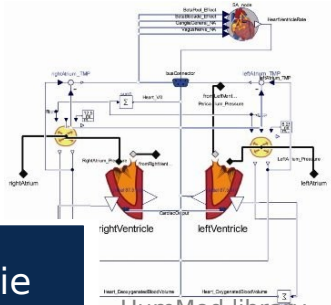
# Modelica - Anwendungsfelder

**Fahrzeug-technik**



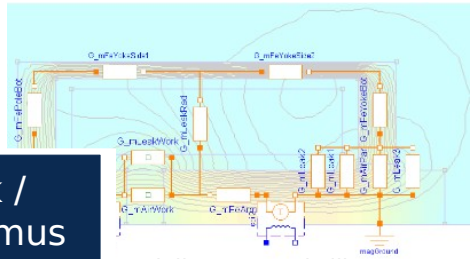
Vehicle Dynamics library

**Physiologie**



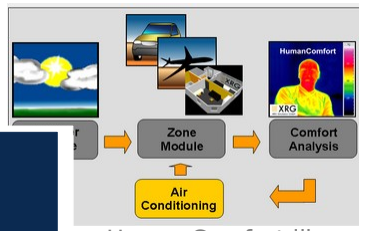
HumMod library

**Elektrik / Magnetismus**



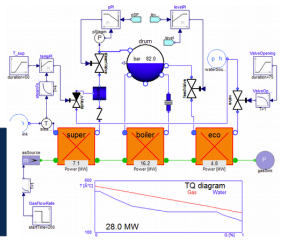
Modelica Magnetic library

**Gebäude / Klima**




HumanComfort library

**Kraftwerke**



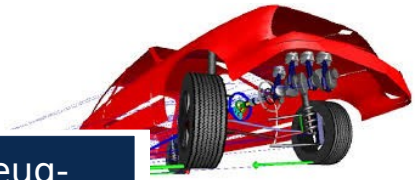

ThermoPower Library

... grundsätzlich alles,
was sich mittels Gleichungen
beschreiben lässt.

©V. Waurich, TU Dresden

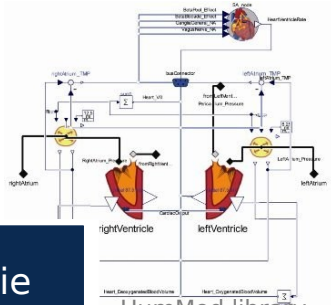
# Modelica - Anwendungsfelder

**Fahrzeug-technik**



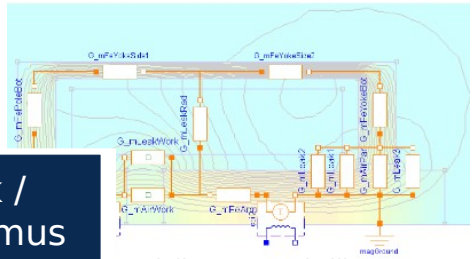
Vehicle Dynamics library

**Physiologie**



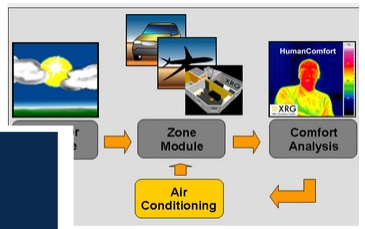
HumMod library

**Elektrik / Magnetismus**



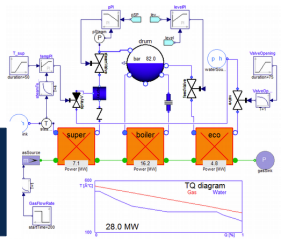
Modelica Magnetic library

**Gebäude / Klima**




HumanComfort library

**Kraftwerke**




ThermoPower Library

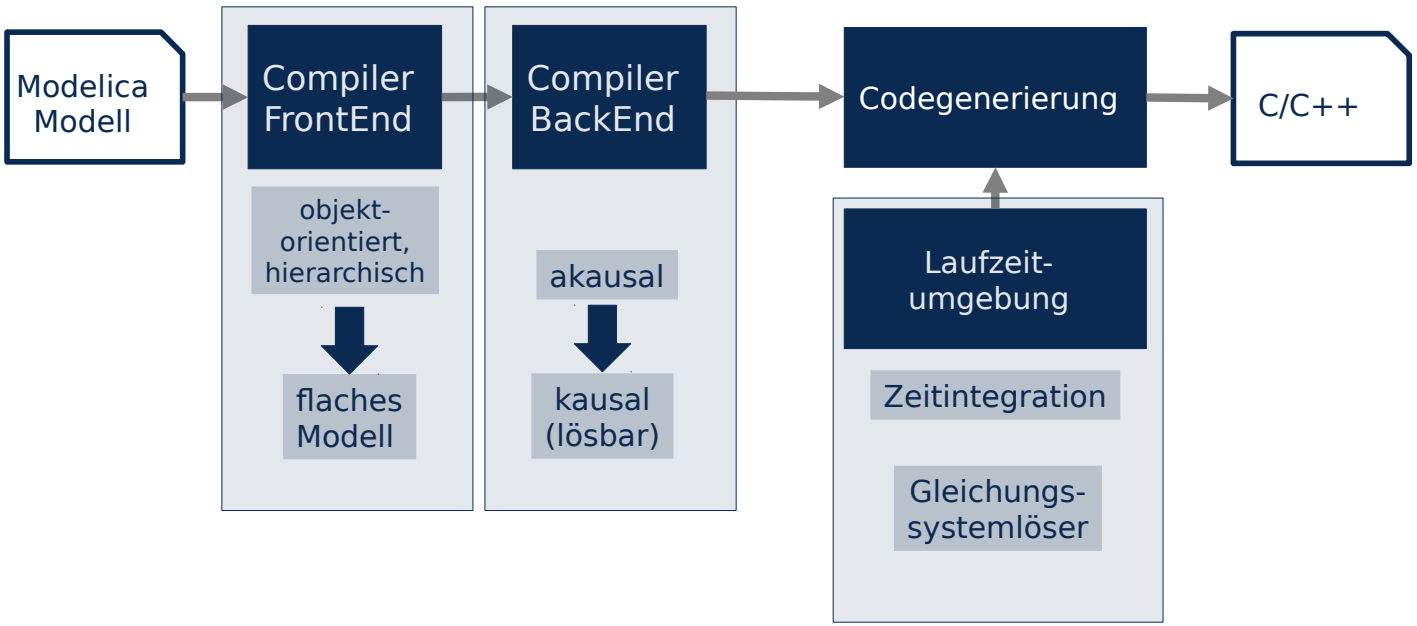
... grundsätzlich alles,
was sich mittels Gleichungen
beschreiben lässt.

©V. Waurich, TU Dresden

Verfügbare Modelica-Compiler: Dymola, SimulationX, OpenModelica, ...

# Modelica-Compiler

OpenModelica
[https://openmodelica.org/]




©V. Waurich, TU Dresden

- Daneben weitere Tools, z. B. Debugger, OMEdit
- Lizenzierung erlaubt auch gewerblichen Gebrauch
- Parallelisierung von OM-Simulationen im Projekt HPC-OM

# Ableitungsberechnung

Die betrachteten Simulationen und Optimierungen basieren grundlegend auf der effizienten Berechnung von Ableitungsinformationen.

Wie erhält man Ableitungsinformationen?

- Per Hand
  - Analytische Ausdrücke für Ableitungen in Source-Code
- Finite Differenzen
  - Approximation der Ableitung durch Differenzenquotienten, z. B. $f'(x) \approx \frac{f(x+h)-f(x)}{h}$
- Symbolische Differenziation
  - Verwendung von Algebra-Systemen

# Ableitungsberechnung

Die betrachteten Simulationen und Optimierungen basieren grundlegend auf der effizienten Berechnung von Ableitungsinformationen.

Wie erhält man Ableitungsinformationen?

- Per Hand
  - Analytische Ausdrücke für Ableitungen in Source-Code
- Finite Differenzen
  - Approximation der Ableitung durch Differenzenquotienten, z. B. $f'(x) \approx \frac{f(x+h)-f(x)}{h}$
- Symbolische Differenziation
  - Verwendung von Algebra-Systemen

Gut bekannte Vor- aber auch Nachteile.

Geht es besser?

# Ableitungsberechnung

Die betrachteten Simulationen und Optimierungen basieren grundlegend auf der effizienten Berechnung von Ableitungsinformationen.

Wie erhält man Ableitungsinformationen?

- Per Hand
  - Analytische Ausdrücke für Ableitungen in Source-Code
- Finite Differenzen
  - Approximation der Ableitung durch Differenzenquotienten, z. B. $f'(x) \approx \frac{f(x+h)-f(x)}{h}$
- Symbolische Differenziation
  - Verwendung von Algebra-Systemen

Gut bekannte Vor- aber auch Nachteile.

Geht es besser?

**Yes, we can!**

# Algorithmisches Differenzieren

Grundlegende Idee

- Beliebige Funktion $f$ als $f = f_1 \circ f_2 \circ \ldots f_n$
- $\frac{\partial f_i}{\partial x}$ sind bekannt
- $\frac{\partial f}{\partial x}$ via Kettenregel

# Algorithmisches Differenzieren

Grundlegende Idee

- Beliebige Funktion $f$ als $f = f_1 \circ f_2 \circ \ldots f_n$
- $\frac{\partial f_i}{\partial x}$ sind bekannt
- $\frac{\partial f}{\partial x}$ via Kettenregel

Beispiel: $y = \sin(x_1 x_2) x_3$

| | | |
|---|---|---|
| $v_1$ | $=$ | $x_1 x_2$ |
| $v_2$ | $=$ | $\sin(v_1)$ |
| $v_3$ | $=$ | $v_2 x_3$ |
| $y$ | $=$ | $v_3$ |

# Algorithmisches Differenzieren

Grundlegende Idee

- Beliebige Funktion $f$ als $f = f_1 \circ f_2 \circ \ldots f_n$
- $\frac{\partial f_i}{\partial x}$ sind bekannt
- $\frac{\partial f}{\partial x}$ via Kettenregel

Beispiel: $y = \sin(x_1 x_2) x_3$

$$
\begin{array}{lcl}
v_1 & = & x_1 x_2 \\
v_2 & = & \sin(v_1) \\
v_3 & = & v_2 x_3 \\
\hline
y & = & v_3
\end{array}
$$

$$
\begin{array}{lcl}
\dot{v}_1 & = & \dot{x}_1 x_2 + x_1 \dot{x}_2 \\
\dot{v}_2 & = & \cos(v_1)\dot{v}_1 \\
\dot{v}_3 & = & \dot{v}_2 x_3 + v_2 \dot{x}_3 \\
\hline
\dot{y} & = & \dot{v}_3
\end{array}
$$

# Algorithmisches Differenzieren

Beispiel: $y = \sin(x_1 x_2) x_3$

- Gesucht ist $\frac{\partial y}{\partial x_1}$ im Punkt $(1, 3, 7)$
- $x_1$ ist einzige unabhängige Variable, d. h. $\dot{x}_1 = 1$, $\dot{x}_2 = 0$ und $\dot{x}_3 = 0$

| | | | | |
|---|---|---|---|---|
| $v_1$ | $=$ | $x_1 x_2$ | $=$ | $3.0$ |
| $v_2$ | $=$ | $\sin(v_1)$ | $=$ | $0.14112$ |
| $v_3$ | $=$ | $v_2 x_3$ | $=$ | $0.98784$ |
| $y$ | $=$ | $v_3$ | $=$ | $0.98784$ |

| | | | | |
|---|---|---|---|---|
| $\dot{v}_1$ | $=$ | $\dot{x}_1 x_2 + x_1 \dot{x}_2$ | $=$ | $3.0$ |
| $\dot{v}_2$ | $=$ | $\cos(v_1)\dot{v}_1$ | $=$ | $-2.96997$ |
| $\dot{v}_3$ | $=$ | $\dot{v}_2 x_3 + v_2 \dot{x}_3$ | $=$ | $-20.78984$ |
| $\dot{y}$ | $=$ | $\dot{v}_3$ | $=$ | $-20.78984$ |

# Algorithmisches Differenzieren

Beispiel: $y = \sin(x_1 x_2) x_3$

- Gesucht ist $\frac{\partial y}{\partial x_1}$ im Punkt $(1, 3, 7)$
- $x_1$ ist einzige unabhängige Variable, d. h. $\dot{x}_1 = 1$, $\dot{x}_2 = 0$ und $\dot{x}_3 = 0$

| | | | | |
|---|---|---|---|---|
| $v_1$ | $=$ | $x_1 x_2$ | $=$ | $3.0$ |
| $v_2$ | $=$ | $\sin(v_1)$ | $=$ | $0.14112$ |
| $v_3$ | $=$ | $v_2 x_3$ | $=$ | $0.98784$ |
| $y$ | $=$ | $v_3$ | $=$ | $0.98784$ |

| | | | | |
|---|---|---|---|---|
| $\dot{v}_1$ | $=$ | $\dot{x}_1 x_2 + x_1 \dot{x}_2$ | $=$ | $3.0$ |
| $\dot{v}_2$ | $=$ | $\cos(v_1)\dot{v}_1$ | $=$ | $-2.96997$ |
| $\dot{v}_3$ | $=$ | $\dot{v}_2 x_3 + v_2 \dot{x}_3$ | $=$ | $-20.78984$ |
| $\dot{y}$ | $=$ | $\dot{v}_3$ | $=$ | $-20.78984$ |

AD liefert ...

- Ableitungen beliebiger Ordnung in Maschinengenauigkeit
- und mit abschätzbarem Mehraufwand

für beliebige Funktionen, die in Quelltext vorliegen.

# AD in OpenModelica

Also sollen wir AD-Funktionalität in OpenModelica implementieren?

Nein, nutze eine etabliertes, effizientes und gehärtetes Werkzeug!

ADOL-C[1] (**A**utomatic **D**ifferentiation by **O**ver**L**oading in **C**++)

- Open-Source
- Breite Nutzergemeinschaft, hohe Akzeptanz
- Umfangreiche Funktionalität (Gradienten, Jacobi- und Hessematrix, u.v.m)
- Basiert auf Operator-Überladung in C/C++
- Ableitung von OpenMP- und MPI-parallelem Code

```
class adouble {
  double val;
  double dot;
}
```

```
adouble operator* (adouble a, adouble b) {
  adouble c;
  c.val = a.val * b.val;
  c.dot = a.dot * b.val + a.val * b.dot;
  return c;
}
```

[1] https://projects.coin-or.org/ADOL-C

# OpenModelica + ADOL-C

## 2014

- Prototyp mit `adouble` im Simulationscode
- Trace wird während der Simulation erzeugt

# OpenModelica + ADOL-C

## 2014

- Prototyp mit `adouble` im Simulationscode
- Trace wird während der Simulation erzeugt

## Neu

- OMC erzeugt das *ADOL-C-Trace* direkt
- Simulationscode enthält nur ADOL-C-Treiberaufrufe, keinen Ableitungscode

Modelica-Modell:

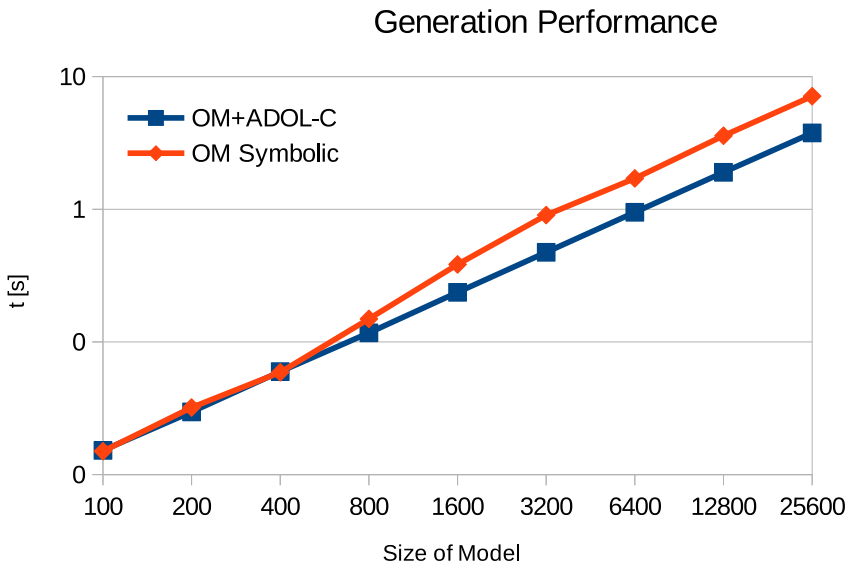
```
model A
  parameter Real a=-0.25;
  Real x,y;
equation
 der(y) = y/x + x*3.0 + a;
 der(x) = x + log(x)*(-3.0);
end A;
```

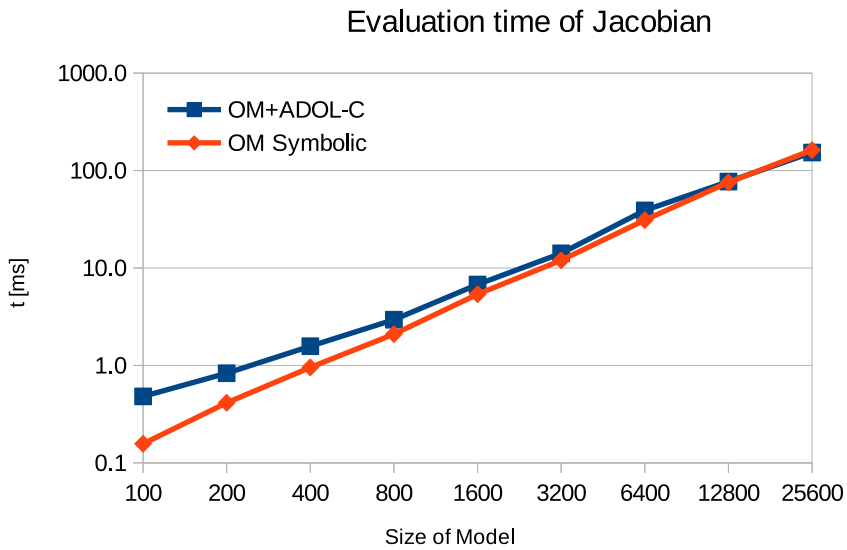
ASCII-Trace:

```
// allocation of used variables
{ op:assign_d_zero loc:0 }
{ op:assign_d_zero loc:1 }
{ op:assign_d_zero loc:2 }
{ op:assign_d_zero loc:3 }
// define independent -> x, y
{ op:assign_ind loc:0 }
{ op:assign_ind loc:1 }
// operations
{ op:div_a_a loc:1 loc:0 loc:4 }
{ op:mult_d_a loc:0 loc:5 val:3.0 }
{ op:assign_p loc:1 loc:6 }
{ op:plus_a_a loc:5 loc:6 loc:7 }
{ op:plus_a_a loc:4 loc:7 loc:3 }
{ op:log_op loc:0 loc:4 }
...
```

# OpenModelica + ADOL-C

Messungen[1] mit dem Modell
*ScalableTestSuite.Elementary.SimpleODE.Models.CascadedFirstOrder*

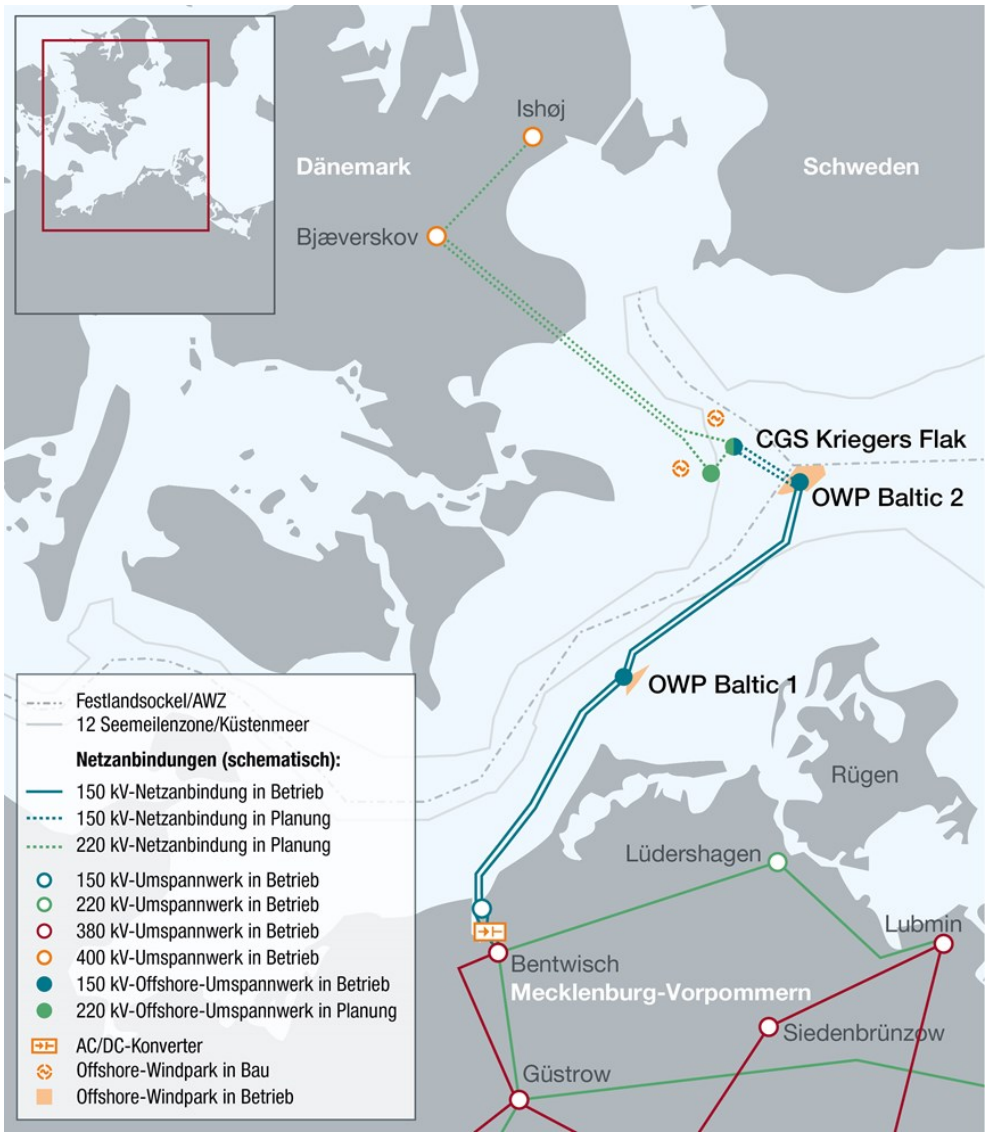



- OM Symbolic Module[2]: nur Jacobimatrix
- OM+ADOL-C: Tuning offen

[1] W. Braun et al.: *Towards Adjoint and Directional Derivatives in FMI utilizing ADOL-C within OpenModelica*, Modelica Conference 2017
[2] W. Braun et al.: *Generic Differentiation Module and its Application within the OMC Backend*, OpenModelica Workshop 2014

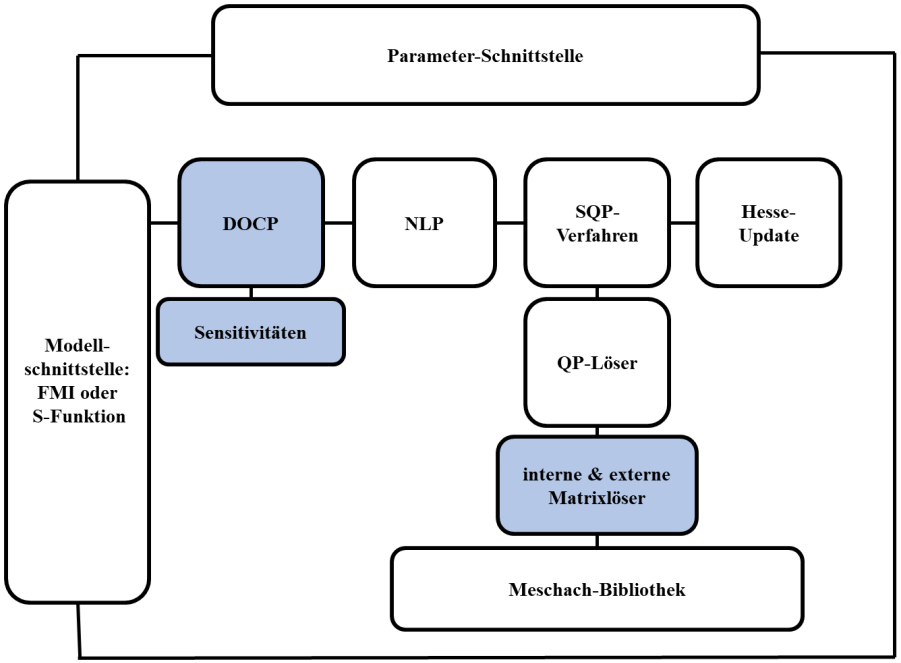
# Kriegers Flak




©50Hertz Transmission GmbH

- Teil des weltweiten ersten Offshore-Elektrizitätsnetzes
- Übertragungskapazität von 400 MW, Leistung von 600 MW
- Interkonnektor soll die Windparks Kriegers Flak (Dänemark) and Baltic 2 (Deutschland) verbinden
- Erneuerbare Energie an europäische Verbraucher, Stärkung des regionalen Energiemarktes und erhöhte Liefersicherheit durch mehr Kapazität

# Optimierungslöser HQP[1]




©C. Grindler, ABB

- Innere-Punkte basiertes SQP-Verfahren für große dünnbesetzte Probleme + Mehrfachschießverfahren (MSV)
- Statische und dynamische Optimierung
- Neu:
  - Externe, parallele Gleichungssystemlöser
  - Parallelisierung des MSV und Ausnutzung der Dünnbesetztheit

[1] https://github.com/omuses/hqp

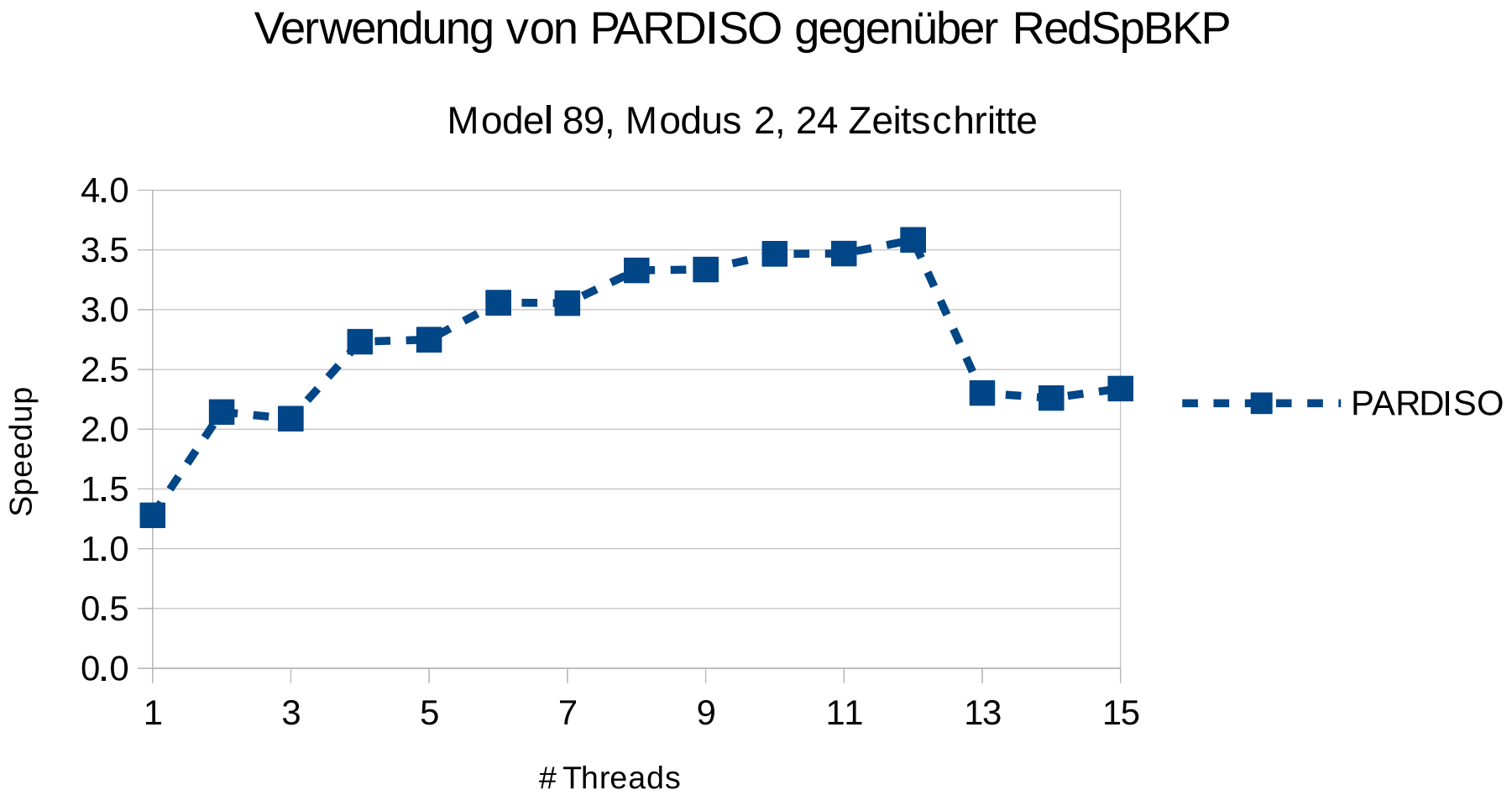
# Benchmark-Modelle - Leistungsflussrechnung

- Leistungsflussrechnung für Übertragungsnetze
  - Ermittelung aller Wirk- und Blindleistungsflüsse eines Stromnetzes aus vorgegebenen Einspeiseleistungen und Belastungen im Betrieb
- IEEE-Testsysteme
- Netzmodellierungen in OpenModelica und Erzeugung von FMU

Konfiguration

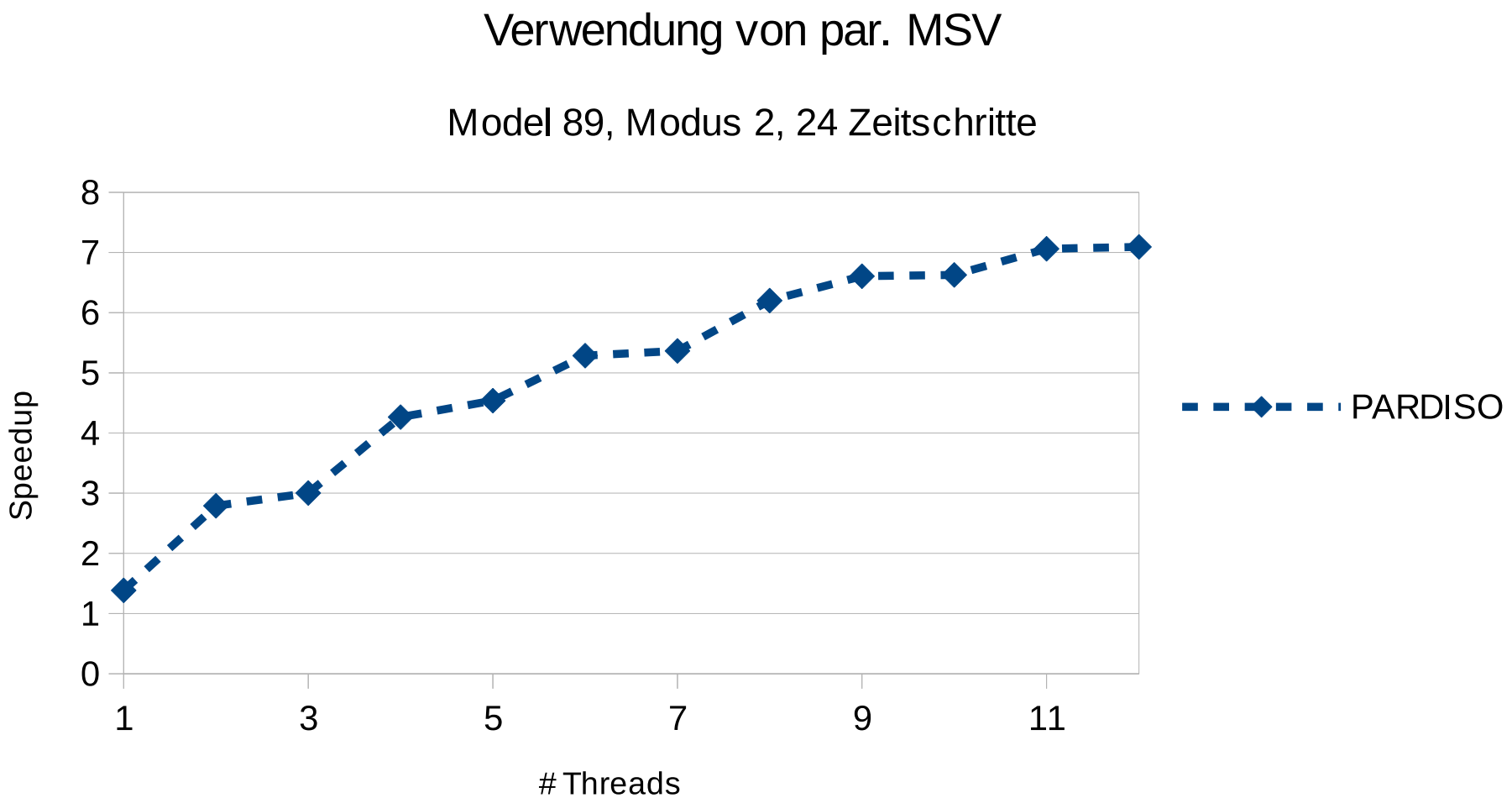
- Intel Compiler 2017
- Intel MKL 2017
- HPC-System Taurus: 1 Haswell-Knoten mit 2x Intel(R) Xeon(R) CPU E5-2680 v3 mit je 12 Kernen @ 2,50GHz, Hyper-Threading aus

# HQP - Parallele Gleichungssystemlöser




- Deutliche Beschleunigung gegenüber bisherigem default Löser
- Gewinn ist abhängig vom Modell

# HQP - Paralleles MSV



# Zusammenfassung und Ausblick

Zusammenfassung

- Kopplung von OpenModelica und ADOL-C
- Parallele Gleichungssystemlöser in HQP
- Paralleles Mehrfachschießverfahren in HQP

# Zusammenfassung und Ausblick

Zusammenfassung

- Kopplung von OpenModelica und ADOL-C
- Parallele Gleichungssystemlöser in HQP
- Paralleles Mehrfachschießverfahren in HQP

Ausblick

- Parallelisierung der Ableitungsberechnung in ADOL-C
- Funktionalität der ADOL-C-Trace-Generierung aus OpenModelica Compiler
- Show-Cases, die alle Komponente zusammenbringen

# Zusammenfassung und Ausblick

Zusammenfassung

- Kopplung von OpenModelica und ADOL-C
- Parallele Gleichungssystemlöser in HQP
- Paralleles Mehrfachschießverfahren in HQP

Ausblick

- Parallelisierung der Ableitungsberechnung in ADOL-C
- Funktionalität der ADOL-C-Trace-Generierung aus OpenModelica Compiler
- Show-Cases, die alle Komponente zusammenbringen

Vielen Dank für Ihre Aufmerksamkeit.