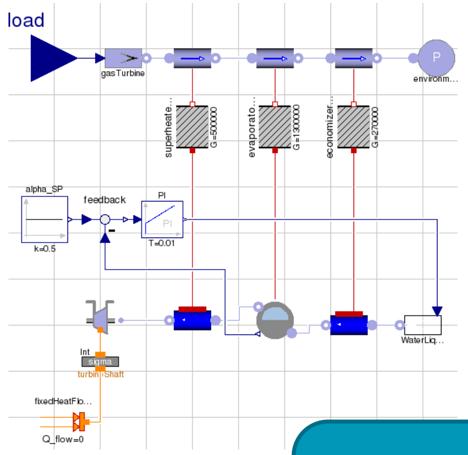
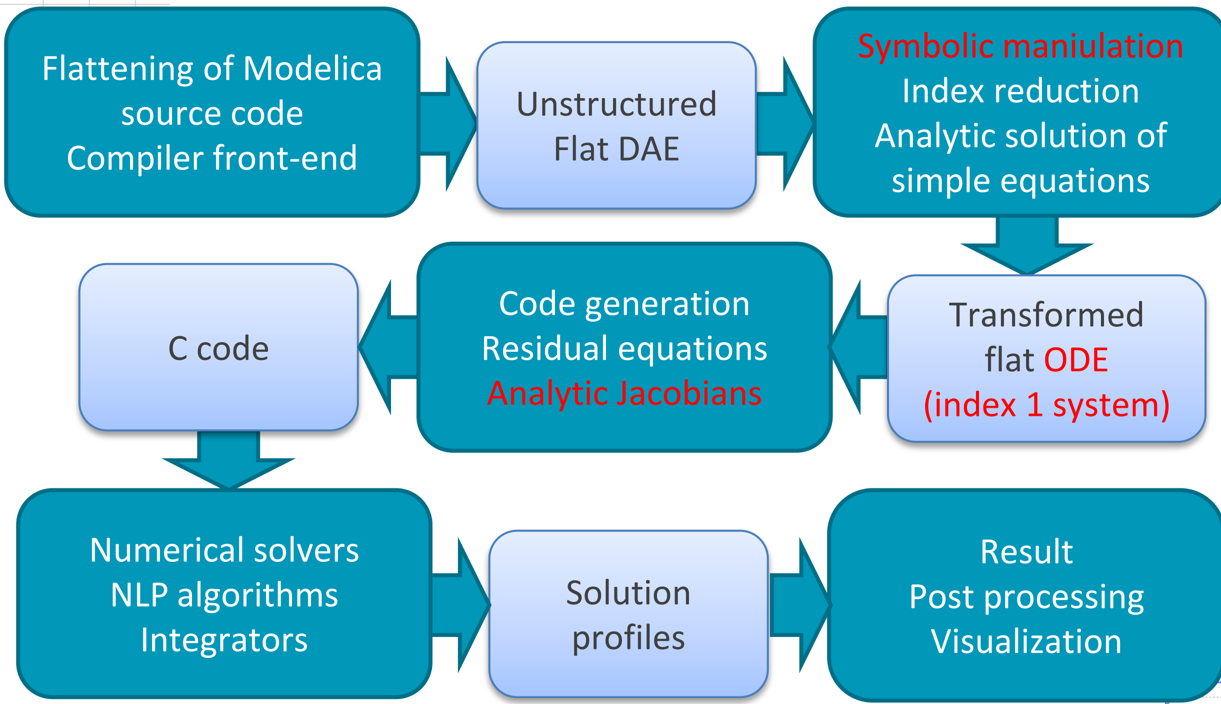
Efficient Symbolical and Numerical Algorithms for nonlinear model predictive control with OpenModelica

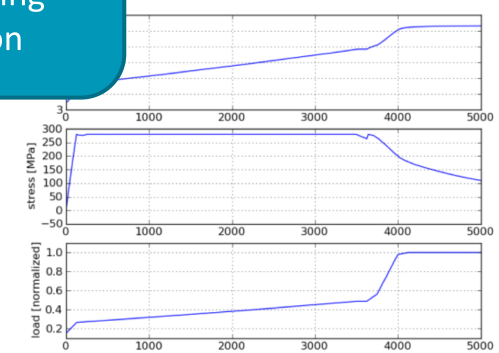
Bernhard Bachmann
Department of Engineering and Mathematics
University of Applied Sciences Bielefeld

A Modelica-based Tool Chain

(Johan Åkesson)







Outline

1. Excerpt of OpenModelica's symbolic machinery

2. Symbolically derived Jacobians
   1. Directional derivatives
   2. Sparsity pattern
   3. Coloring of the Jacobian

3. Nonlinear Optimal Control Problem
   1. General Discretization Scheme
   2. Multiple Shooting/Collocation
   3. Total Collocation
   4. Applications

4. Lessons learned & Outlook

FH Bielefeld
University of
Applied Sciences

Symbolic Machinery of OpenModelica

General representation of DAEs (continuous signals):

$$\underline{0} = \underline{f}\left(t,\ \underline{\dot{x}}(t), \underline{x}(t), \underline{y}(t), \underline{u}(t), \underline{p}\right)$$

t	time
$\underline{\dot{x}}(t)$	vector of differentiated state variables
$\underline{x}(t)$	vector of state variables
$\underline{y}(t)$	vector of algebraic variables
$\underline{u}(\mathrm{t})$	vector of input variables
$\underline{p}$	vector of parameters and/or constants

Basic Transformation Steps

Transformation to explicit state-space representation:

$$\underline{0} = \underline{f}\left(t, \underline{\dot{x}}(t), \underline{x}(t), \underline{y}(t), \underline{u}(t), \underline{p}\right)$$

$$\downarrow$$

$$\underline{0} = \underline{f}\left(t, \underline{z}(t), \underline{x}(t), \underline{u}(t), \underline{p}\right), \qquad \underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix}$$

$$\nearrow$$

$$\underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix} = \underline{g}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

$$\downarrow$$

$$\underline{\dot{x}}(t) = \underline{h}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

$$\underline{y}(t) = \underline{k}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

Implicit function theorem:

Necessary condition for the existence of the transformation is that the following matrix is regular at the point of interest:

$$\det\left(\frac{\partial}{\partial \underline{z}} \underline{f}\left(t, \underline{z}(t), \underline{x}(t), \underline{u}(t), \underline{p}\right)\right) \neq 0$$

Symbolic Transformation Algorithmic Steps

- DAEs and bipartite graph representation
  - Structural representation of the equation system

$$\underline{0} = \underline{f}\left(t, \underline{\dot{x}}(t), \underline{x}(t), \underline{y}(t), \underline{u}(t), \underline{p}\right)$$

$$\downarrow$$

- The matching problem
  - Assign to each variable exact one equation
  - Same number of equations and unknowns

$$\underline{0} = \underline{f}\left(t, \underline{z}(t), \underline{x}(t), \underline{u}(t), \underline{p}\right), \quad \underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix}$$

$$\downarrow$$

- Construct a directed graph
  - Find sinks, sources and strong components
  - Sorting the equation system

$$\underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix} = \underline{g}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

$$\downarrow$$

- Adjacence Matrix and structural regularity
  - Block-lower triangular form (BLT-Transformation)

$$\underline{\dot{x}}(t) = \underline{h}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

$$\underline{y}(t) = \underline{k}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

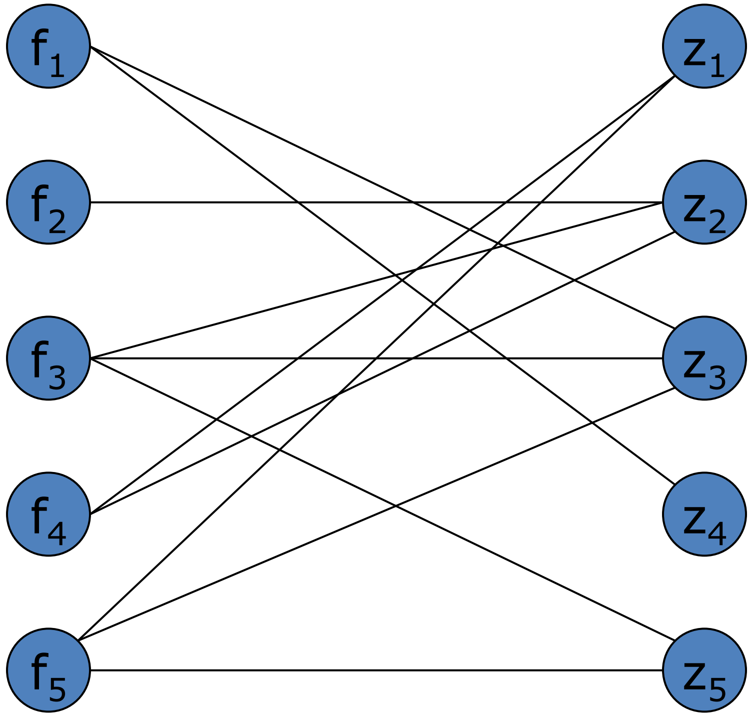
DAEs and Bipartite Graph Representation

Example of a regular DAE:

$$\underline{0} = \underline{f}\left(t, \underline{z}(t), \underline{x}(t), \underline{u}(t), \underline{p}\right), \qquad \underline{z}(t) = \begin{pmatrix} \dot{\underline{x}}(t) \\ \underline{y}(t) \end{pmatrix}$$

$$\begin{aligned}
f_1(z_3,z_4) &= 0 \\
f_2(z_2) &= 0 \\
f_3(z_2,z_3,z_5) &= 0 \\
f_4(z_1,z_2) &= 0 \\
f_5(z_1,z_3,z_5) &= 0
\end{aligned}$$

Bipartite graph



Adjacence matrix

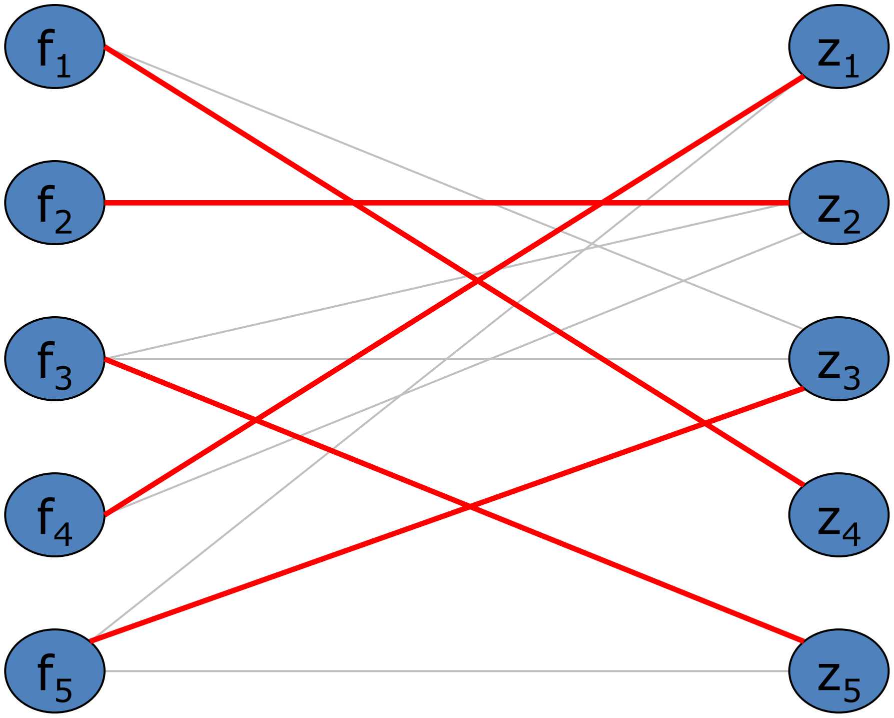
$$\begin{array}{c|c}
 & \begin{matrix} z_1 & z_2 & z_3 & z_4 & z_5 \end{matrix} \\
\begin{matrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{matrix} & \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix}
\end{array}$$

Solve the Matching Problem

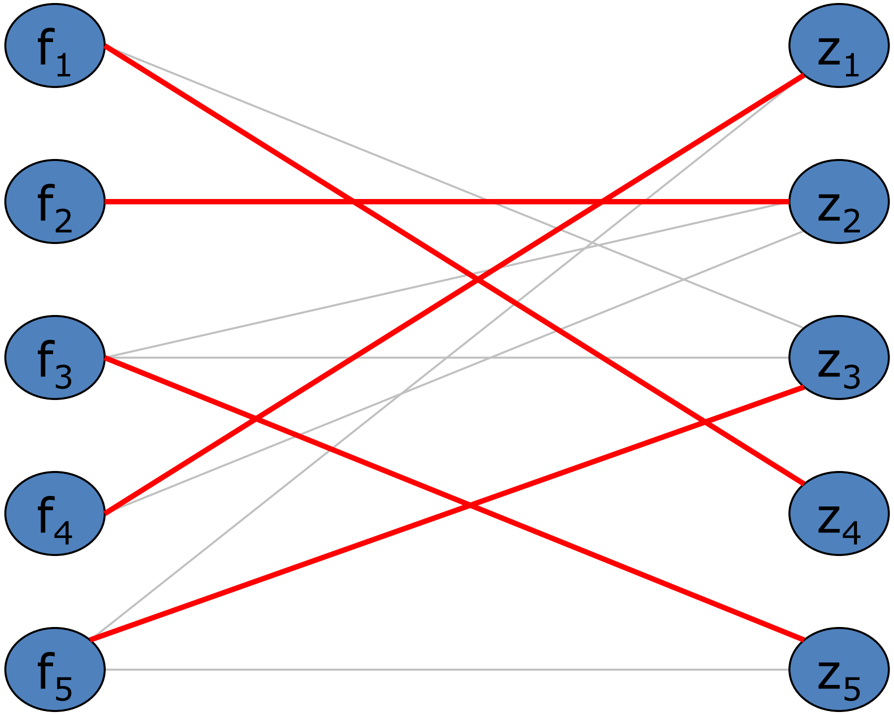
Example of a regular DAE:

$$f_1(z_3,z_4) = 0$$
$$f_2(z_2) = 0$$
$$f_3(z_2,z_3,z_5) = 0$$
$$f_4(z_1,z_2) = 0$$
$$f_5(z_1,z_3,z_5) = 0$$

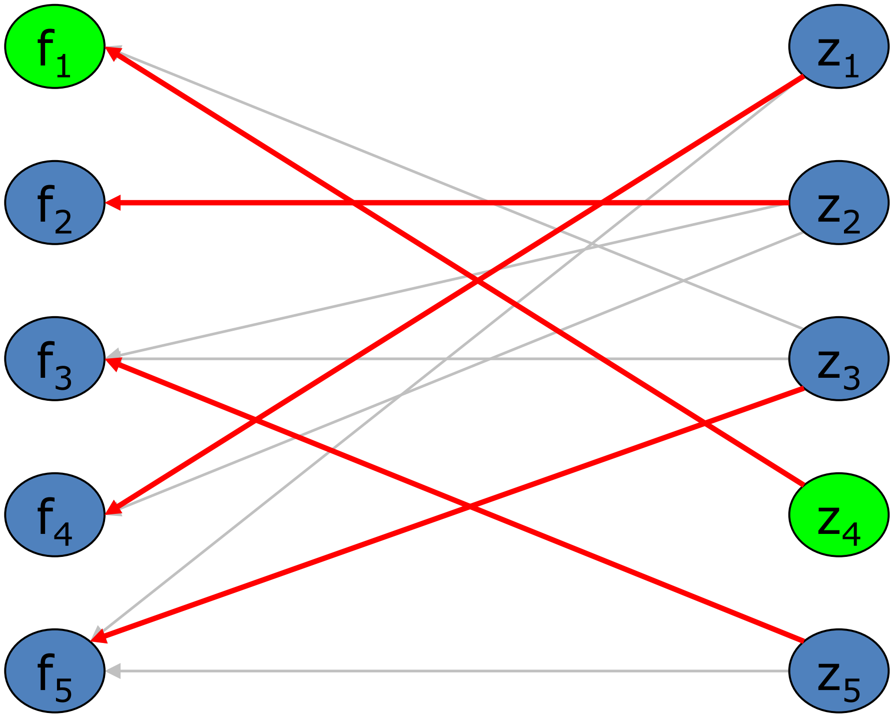
Bipartite graph



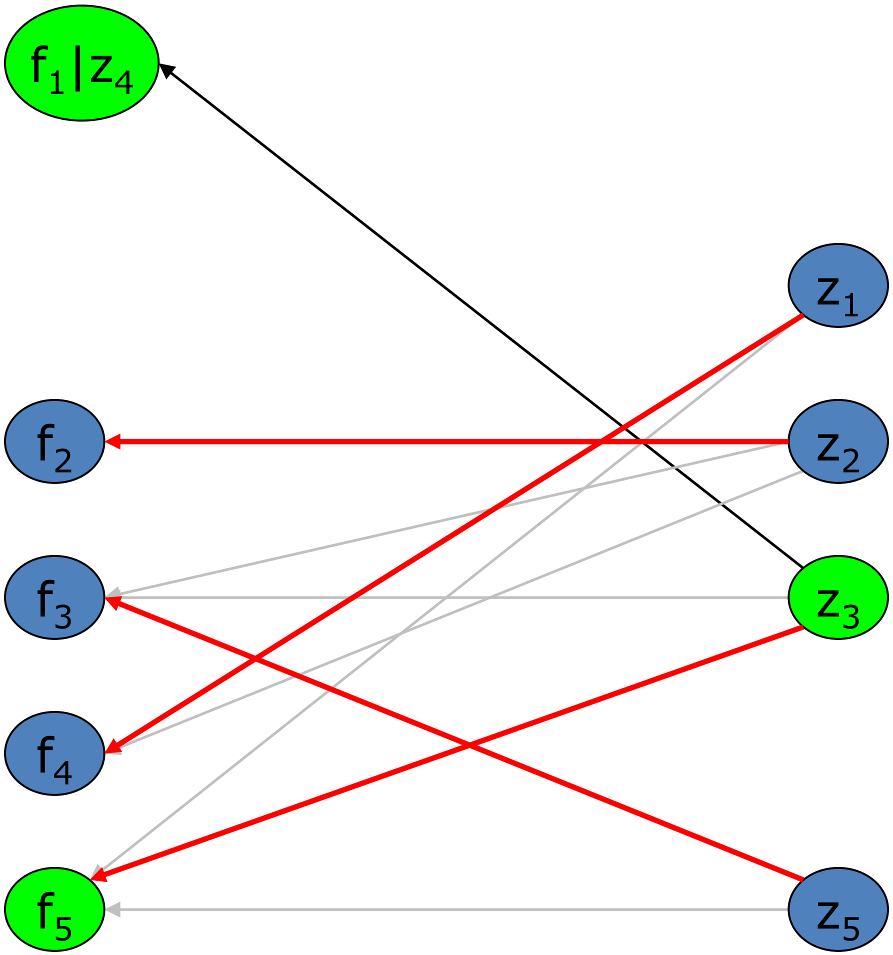
Construct a Directed Graph



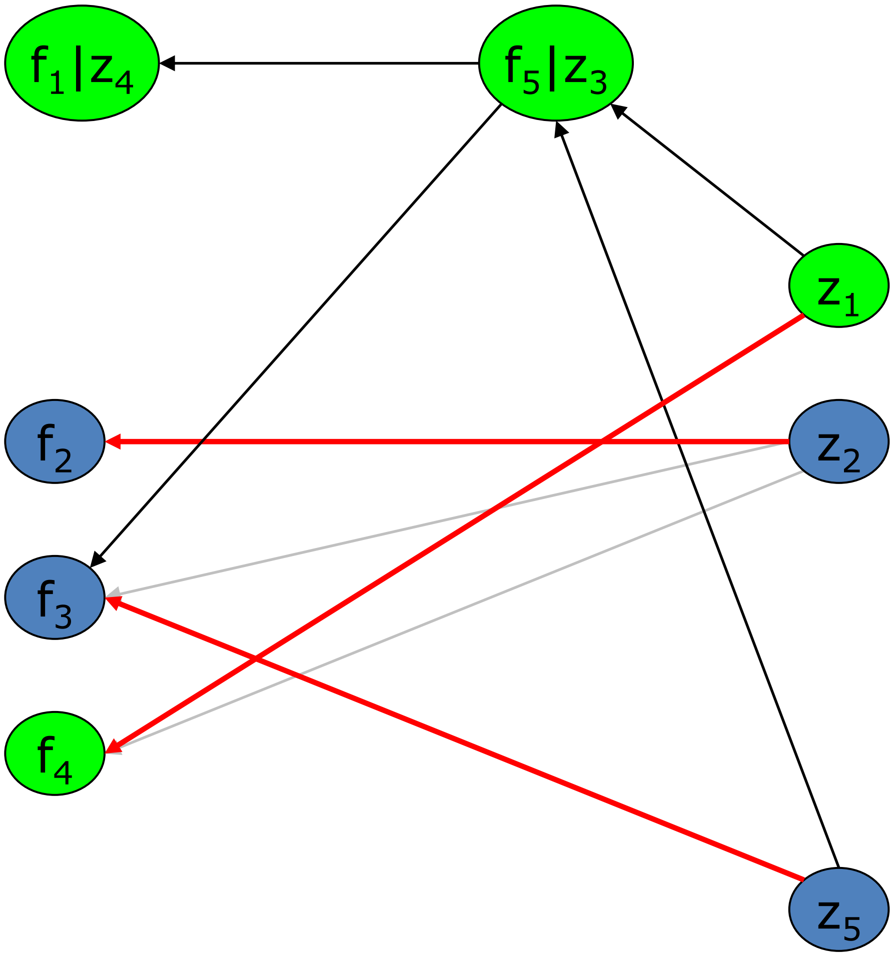
Construct a Directed Graph



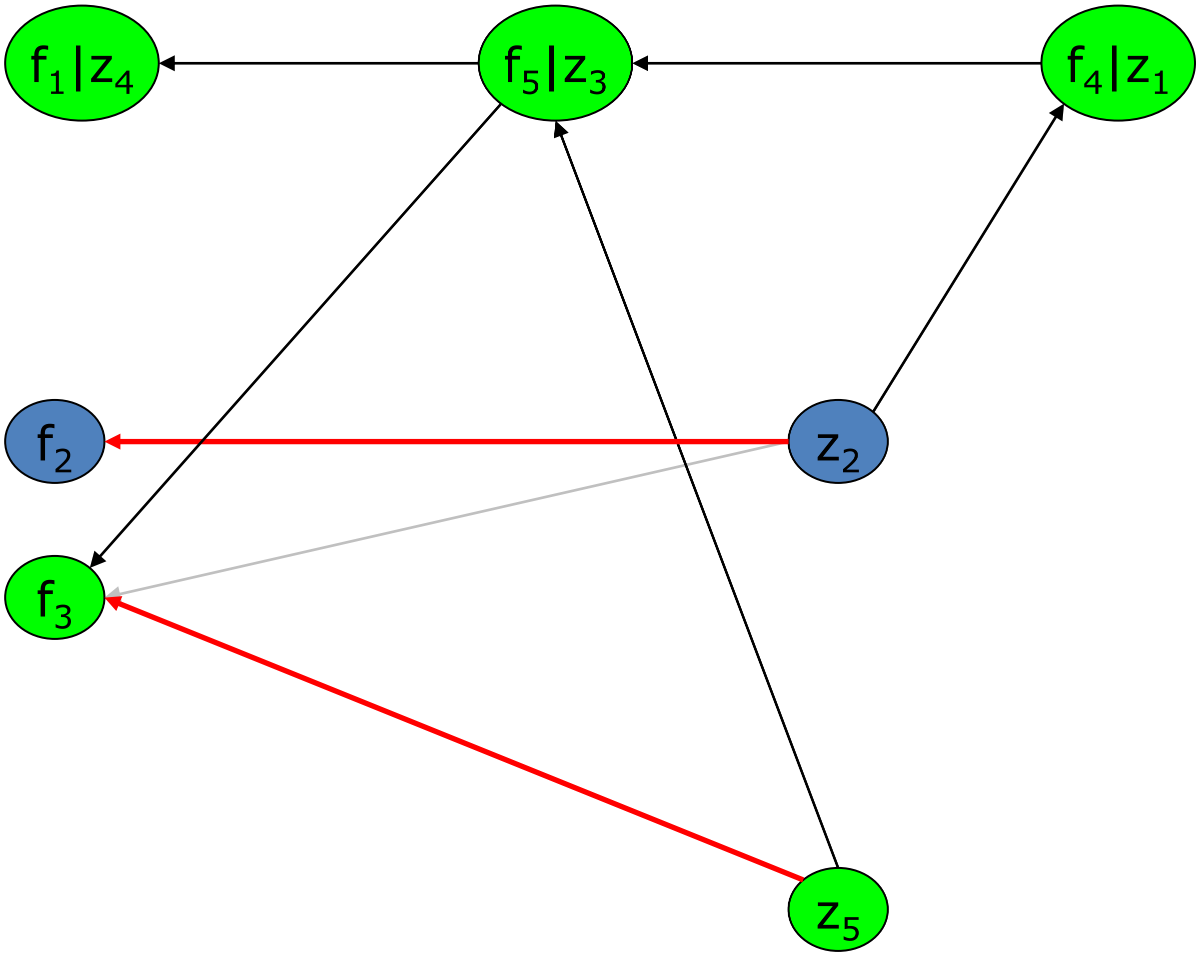
Construct a Directed Graph



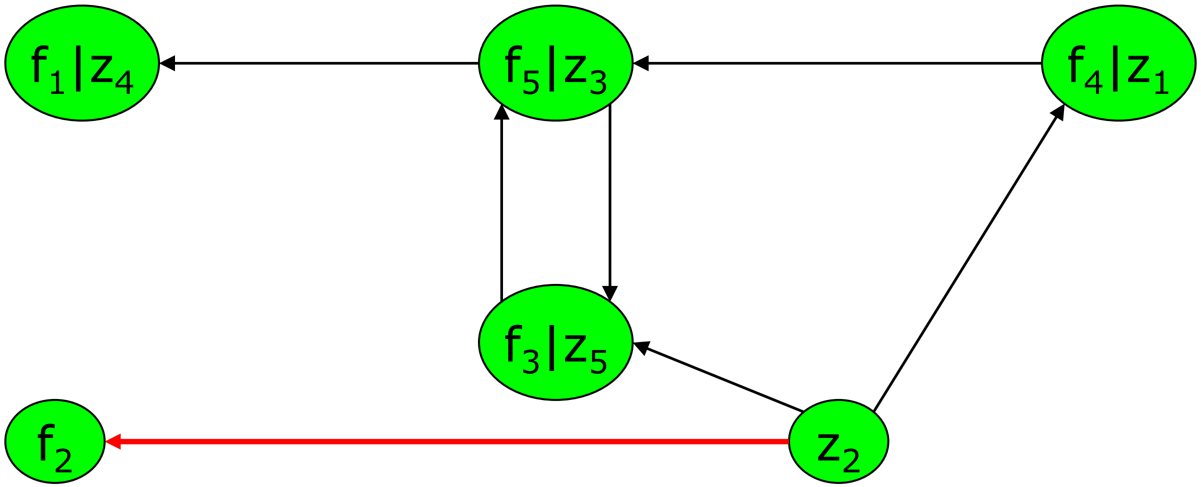
Construct a Directed Graph



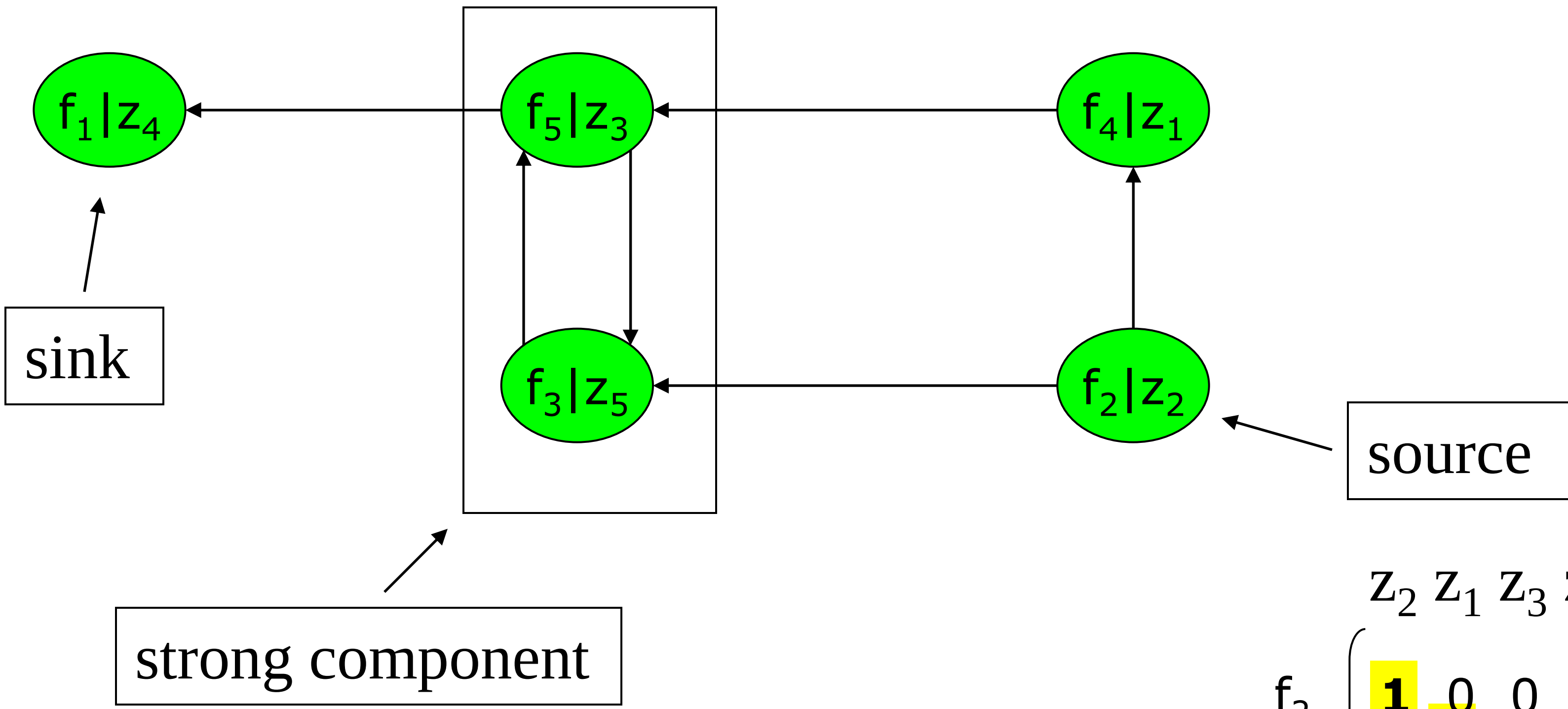
Construct a Directed Graph



Construct a Directed Graph



Construct a Directed Graph



$\Longrightarrow$ Tarjan's algorithm

$$
\begin{array}{c|c}
 & \begin{matrix} z_2 & z_1 & z_3 & z_5 & z_4 \end{matrix} \\
\begin{matrix} f_2 \\ f_4 \\ f_3 \\ f_5 \\ f_1 \end{matrix} &
\begin{pmatrix}
\mathbf{1} & 0 & 0 & 0 & 0 \\
1 & \mathbf{1} & 0 & 0 & 0 \\
1 & 0 & \mathbf{1} & \mathbf{1} & 0 \\
0 & 1 & \mathbf{1} & \mathbf{1} & 0 \\
0 & 0 & 1 & 0 & \mathbf{1}
\end{pmatrix}
\end{array}
$$

Further Efficiency Issues - Dummy-Derivative Method

- Matching algorithm fails
  - System is structurally singular
  - Find minimal subset of equations
    - more equations than unknown variables
  - Singularity is due to equations, constraining states

- Differentiate subset of equations
  - Static state selection during compile time
    - choose one state and corresponding derivative as purely algebraic variable
      - so-called dummy state and dummy derivative
    - by differentiation introduced variables are algebraic
    - continue matching algorithm
    - check initial conditions
  - Dynamic state selection during simulation time
    - store information on constrained states
    - make selection dynamically based on stability criteria
    - new state selection triggers an event (re-initialize states)

Further Efficiency Issues – Algebraic Loops

- Solution of linear equation systems
  - Advanced solver packages (e.g. LAPACK) are used
  - Calculate LU-Decomposition for constant matrices
    - Small systems are inverted symbolically
- Solution of nonlinear systems
  - Advanced solver packages are used
    - Performance is depending on good starting values
  - Analytical Jacobian is provided symbolically
- Tearing systems of equations
  - Reducing the iteration variables dramatically
- Analytical Jacobians of the overall system
  - Minimize simulation/integration time needed

Outline

1. Excerpt of OpenModelica's symbolic machinery

2. Symbolically derived Jacobians
   i. Directional derivatives
   ii. Sparsity pattern
   iii. Coloring of the Jacobian

3. Nonlinear Optimal Control Problem
   i. General Discretization Scheme
   ii. Multiple Shooting/Collocation
   iii. Total Collocation
   iv. Applications

4. Lessons learned & Outlook

FH Bielefeld
University of
Applied Sciences

MODELICA

Fast Simulation of Fluid Models with Colored Jacobians

Willi Braun, Bernhard Bachmann
Department of Engineering and Mathematics
University of Applied Sciences Bielefeld

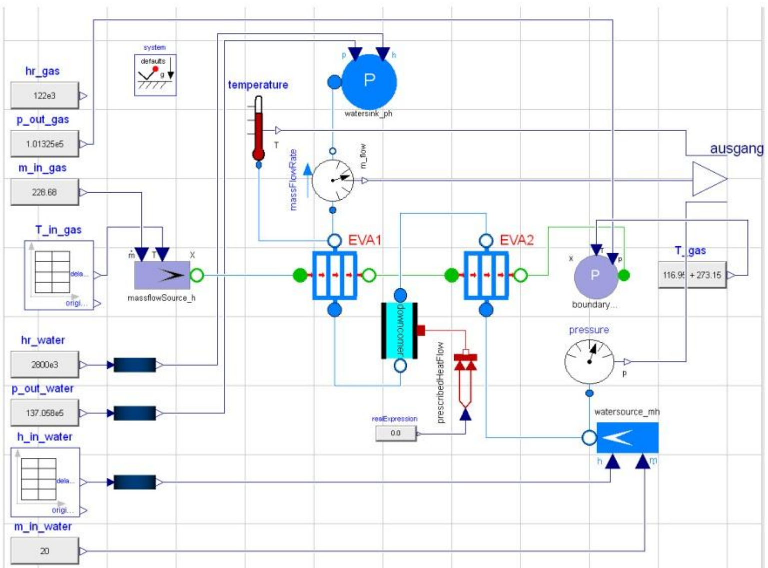
Stephanie Gallardo Yances, Kilian Link
Siemens AG, Energy Section
Erlangen

(see 9th International Modelica Conference)

FH Bielefeld
University of
Applied Sciences

Symbolically Generation of Jacobians

How is simulation time effected by Jacobians?



Fluid Test Model

States	231
Equations	942
Simulation time	10.8
J evaluations	111
J evaluation time	9.7

The evaluation of Jacobians effects the simulation time a lot!

Symbolically Generation of Jacobians

State-Space Equations

$$\begin{pmatrix} \dot{\underline{x}}(t) \\ \underline{y}(t) \end{pmatrix} = \begin{pmatrix} h(\underline{x}(t), \underline{u}(t), \underline{p}, t) \\ k(\underline{x}(t), \underline{u}(t), \underline{p}, t) \end{pmatrix}$$

Simulation

- Many integration algorithms need "the Jacobian" : $A(t) = \frac{\partial h}{\partial \underline{x}}$
- Integrator DASSL

Jacobian matrices

- $A(t) = \frac{\partial h}{\partial \underline{x}}$
- $B(t) = \frac{\partial h}{\partial \underline{u}}$
- $C(t) = \frac{\partial k}{\partial \underline{x}}$
- $D(t) = \frac{\partial k}{\partial \underline{u}}$

Symbolically Generation of Jacobians

Jacobian

$$J_A = \frac{\partial \underline{h}}{\partial \underline{x}} = \begin{pmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial x_1} & \cdots & \frac{\partial h_n}{\partial x_n} \end{pmatrix}$$

Full Symbolic Jacobian

Generation of the full symbolic jacobian requires n-times differentiation of every equation.

Symbolically Generation of Jacobians

Jacobian

$$J_A = \frac{\partial \underline{h}}{\partial \underline{x}} = \begin{pmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial x_1} & \cdots & \frac{\partial h_n}{\partial x_n} \end{pmatrix}$$

Generic Directional Derivative

$$J_A = \frac{\partial \underline{h}}{\partial z}(\underline{e}_k)$$

$\underline{e}_k \in R^n := k -$ th coordinate vector

Symbolically Generation of Jacobians

Example

```
model twoflattankmodel
  Real h1, h2;
  Real F1, F2;
  input Real F;
  parameter Real A1=2, A2=0.5;
  parameter Real R1=2, R2=1;
equation
  der(h1) = (F/A1) - (F1/A1);
  der(h2) = (F1/A2) - (F2/A2);
  F1 = R1 * sqrt(h1 - h2);
  F2 = R2 * sqrt(h2);
end twoflattankmodel;
```

Jacobian

$$J_A = \frac{\partial \underline{h}}{\partial z}\left(\frac{\partial \underline{x}}{\partial z}\right) = \begin{pmatrix} \frac{\partial der(h1)}{\partial z}\left(\frac{\partial h1}{\partial z}, \frac{\partial h2}{\partial z}\right) \\ \frac{\partial der(h2)}{\partial z}\left(\frac{\partial h1}{\partial z}, \frac{\partial h2}{\partial z}\right) \end{pmatrix}$$

$$\frac{\partial der(h2)}{\partial z} = \frac{\frac{\partial F1}{\partial z} * A2 - F1 * \frac{\partial A2}{\partial z}}{A2^2} - \frac{\frac{\partial F2}{\partial z} * A2 - F2 * \frac{\partial A2}{\partial z}}{A2^2}$$

$$\frac{\partial F1}{\partial z} = R1 * \frac{\left(\frac{\partial h1}{\partial z} - \frac{\partial h2}{\partial z}\right)}{2 * \sqrt{h1 - h2}}$$

$$\frac{\partial F2}{\partial z} = R2 * \frac{\frac{\partial h2}{\partial z}}{2 * \sqrt{h2}}$$

$$\frac{\partial der(h1)}{\partial z} = \frac{\frac{\partial F}{\partial z} * A1 - F * \frac{\partial A1}{\partial z}}{A1^2} - \frac{\frac{\partial F1}{\partial z} * A1 - F1 * \frac{\partial A1}{\partial z}}{A1^2}$$

Symbolically Generation of Jacobians

Numerical

$$\frac{\partial \underline{h}}{\partial \underline{x}} = \frac{(\underline{h}(\underline{x} + \delta \underline{e}_k) - \underline{h}(\underline{x}))}{\delta}$$

$\underline{e}_k \in R^n := k -$ th coordinate vector

Calculate the Jacobian numerical needs $n + 1$ call of the ODE-Block $\underline{h}$.

Symbolical

$$J_A = \frac{\partial \underline{h}}{\partial z}(\underline{e}_k)$$

$\underline{e}_k \in R^n := k -$ th coordinate vector

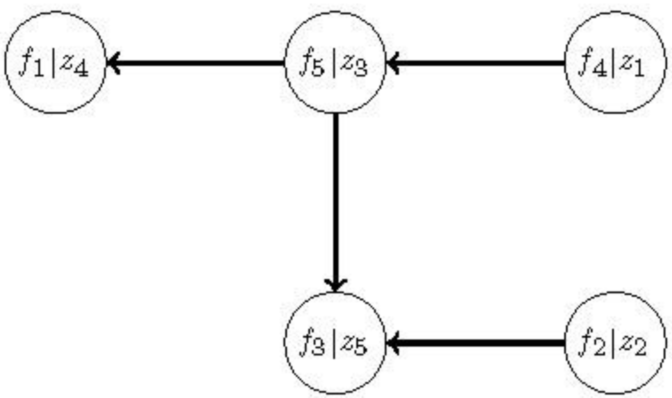
Evaluate the Jacobian symbolical needs n calls of Directional Derivative.

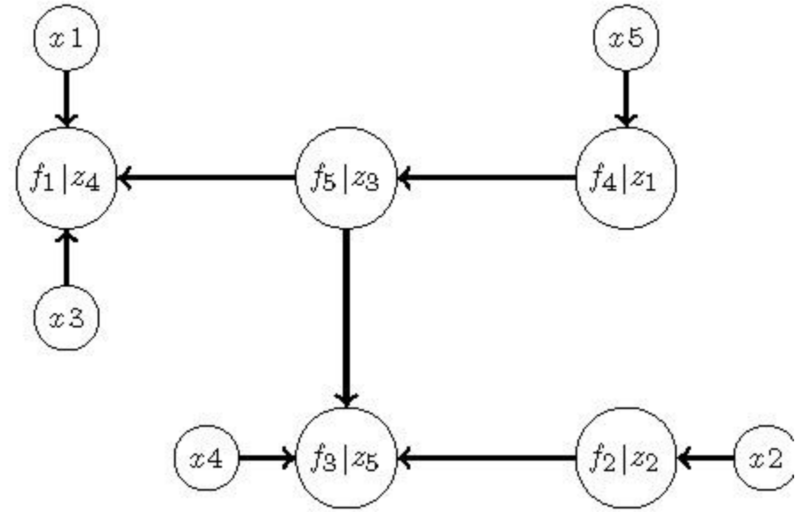
The amount of calls could be reduced by exploiting the **sparsity pattern** and partitioning the columns by colors.

Compute sparsity pattern of the Jacobians

Example system

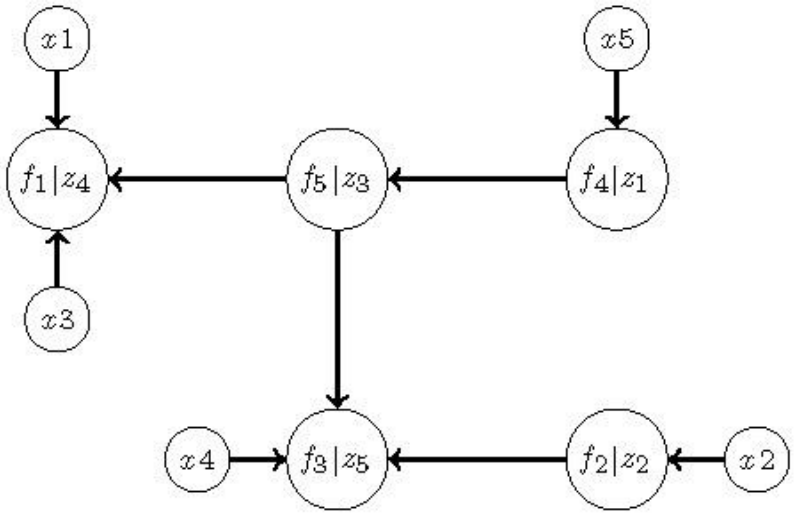
$$\underline{z}(t) = \underline{f}(\underline{x}(t), t)$$





$$J = \begin{pmatrix} 0 & 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 & * \\ * & 0 & * & 0 & * \\ 0 & * & 0 & 0 & 0 \\ 0 & * & 0 & * & * \end{pmatrix}$$

Compute sparsity pattern of the Jacobians



$$J = \begin{pmatrix} 0 & 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 & * \\ * & 0 & * & 0 & * \\ 0 & * & 0 & 0 & 0 \\ 0 & * & 0 & * & * \end{pmatrix}$$

$$\begin{array}{c} \\ f4 \\ f5 \\ f1 \\ f2 \\ f3 \end{array} \begin{array}{c} \begin{array}{cccccccccc} z1 & z3 & z4 & z2 & z5 & x1 & x2 & x3 & x4 & x5 \end{array} \\ \left(\begin{array}{ccccc||ccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}\right) \end{array}$$

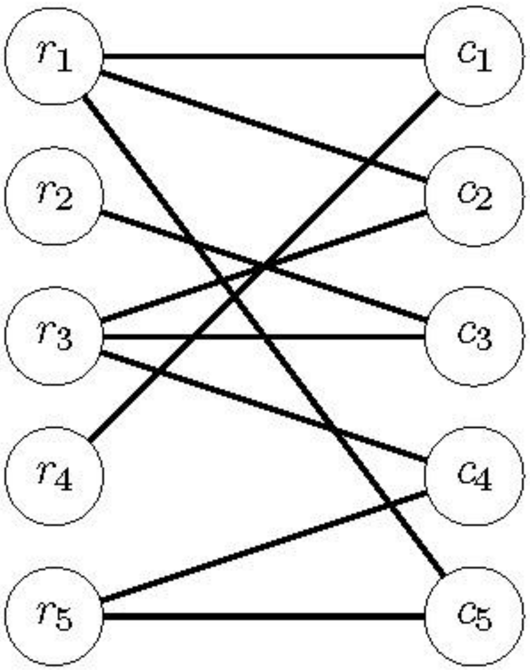
Accumulation Lists

f4: <5>
f5: <5>
f1: <1,3,5>
f2: <2>
f3: <5,2>

Utilize sparsity pattern of the Jacobians

Jacobian

$$J = \begin{pmatrix} j_{11} & j_{12} & 0 & 0 & j_{15} \\ 0 & 0 & j_{23} & 0 & 0 \\ 0 & j_{32} & j_{33} & j_{34} & 0 \\ j_{41} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & j_{54} & j_{55} \end{pmatrix}$$



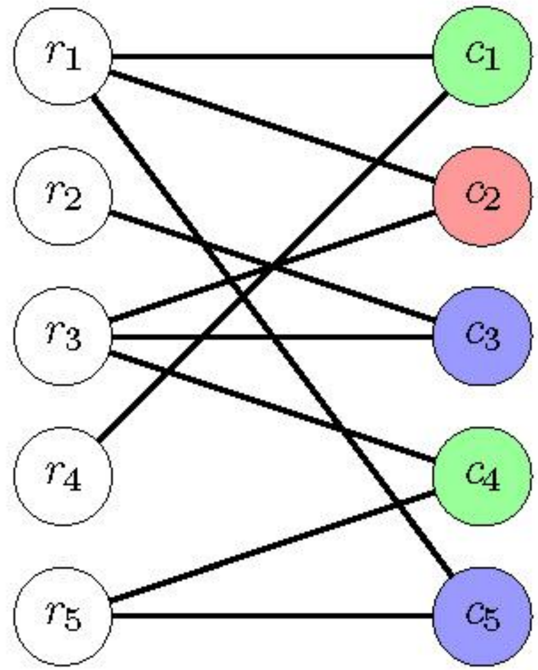
MODELICA

Utilize sparsity pattern of the Jacobians

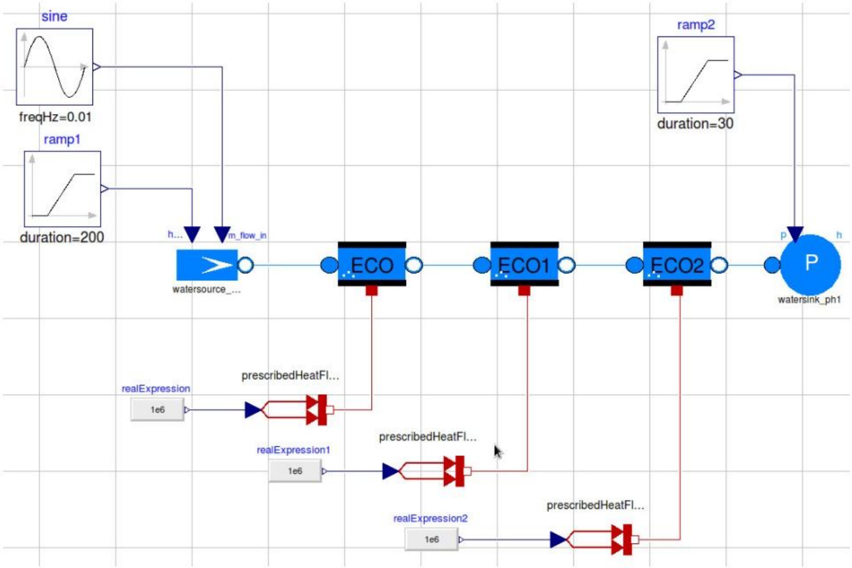
Jacobian

$$J = \begin{pmatrix} j_{11} & j_{12} & 0 & 0 & j_{15} \\ 0 & 0 & j_{23} & 0 & 0 \\ 0 & j_{32} & j_{33} & j_{34} & 0 \\ j_{41} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & j_{54} & j_{55} \end{pmatrix}$$

$$J_R = \begin{pmatrix} j_{11} & j_{12} & j_{15} \\ 0 & 0 & j_{23} \\ j_{34} & j_{32} & j_{33} \\ j_{41} & 0 & 0 \\ j_{54} & 0 & j_{55} \end{pmatrix}$$



Performance gain of implementation



Model details

States	231
Equations	1 006
JacElements	53 361
NonZero	3 032
Colors	79

Simulation statistics

method	steps	F-Eval	J-Eval	time
num	922	27184	111	10.8
numC	922	8929	94	4.5
sym	937	1539	103	8.5
symC	937	1539	103	4.3
Dymola	783	8772	90	1.6

Outline

1. Excerpt of OpenModelica's symbolic machinery

2. Symbolically derived Jacobians
   i. Directional derivatives
   ii. Sparsity pattern
   iii. Coloring of the Jacobian

3. Nonlinear Optimal Control Problem
   i. General Discretization Scheme
   ii. Multiple Shooting/Collocation
   iii. Total Collocation
   iv. Applications

4. Lessons learned & Outlook

FH Bielefeld
University of
Applied Sciences

Parallel Multiple-Shooting and Collocation Optimization with OpenModelica

Bernhard Bachmann, Lennart Ochel, Vitalij Ruge
Mathematics and Engineering
University of Applied Sciences Bielefeld

Mahder Gebremedhin, Peter Fritzson,
PELAB – Programming Environment Lab

Vaheed Nezhadali, Lars Eriksson, Martin Sivertsson
Vehicular Systems
Linköping University

(see 9th International Modelica Conference)

Nonlinear Optimal Control Problem (NOCP)

Mathematical problem formulation

- objective function

$$\min_{u(t)} J(x(t),u(t),t) = E\left(x\left(t_f\right)\right) + \int_{t_0}^{t_f} L(x(t),u(t),t)\, dt$$

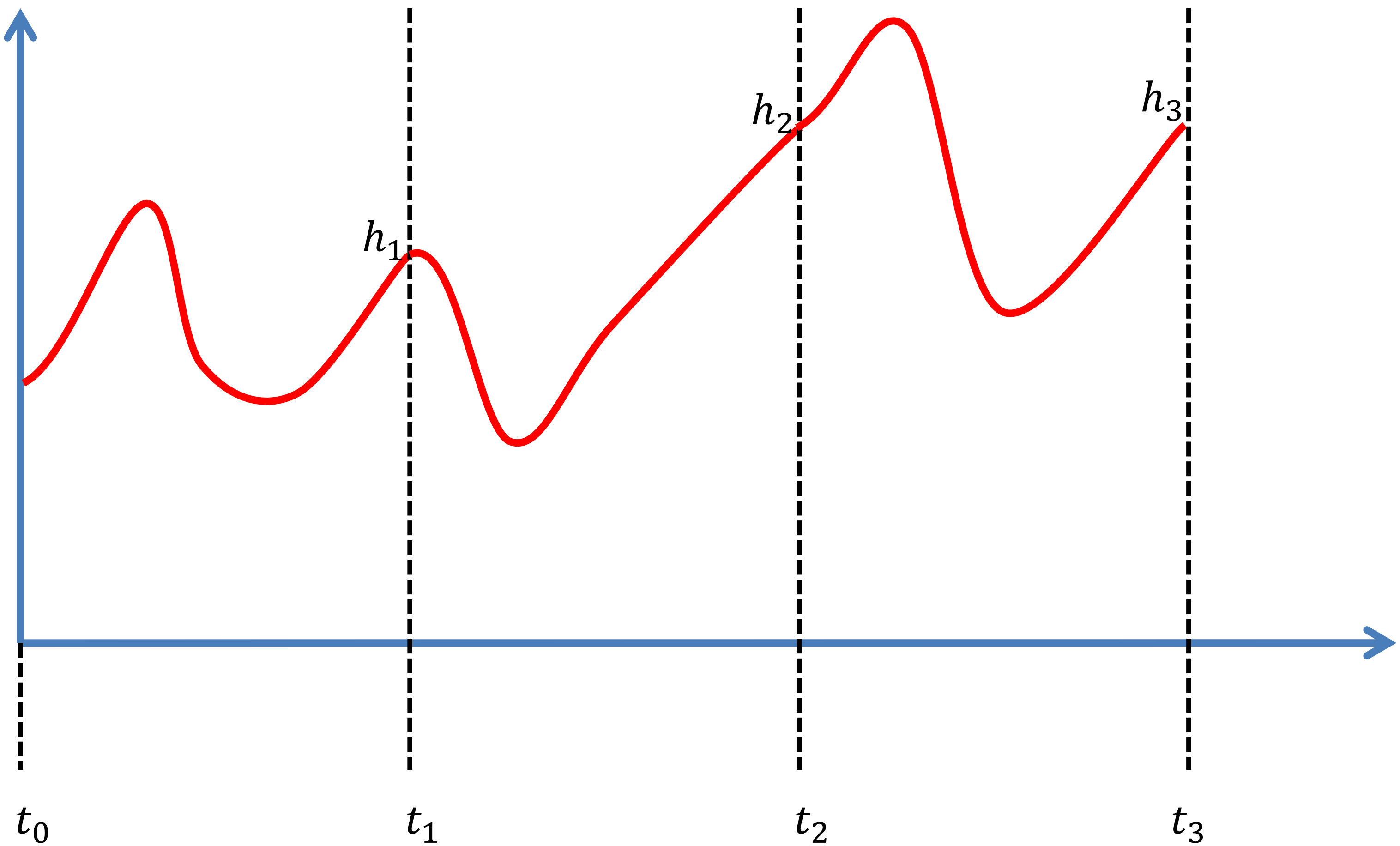
- subject to

$x(t_0)$	$=$	h_0	initial conditions
$\dot{x}(t)$	$=$	$f(x(t),u(t),t)$	DAEs, Modelica
$g(x(t),y(t),u(t),t)$	$\geq$	0	path constraints
$r\left(x\left(t_f\right),y(t_f)\right)$	$=$	0	terminal constraints

Theoretical Background
General discretization scheme

$$x_i(t_{i+1}) = h_i + \int_{t_i}^{t_{i+1}} f(x_i(t), u(t), t)\, dt$$

$$x_i(t_i) = h_i$$

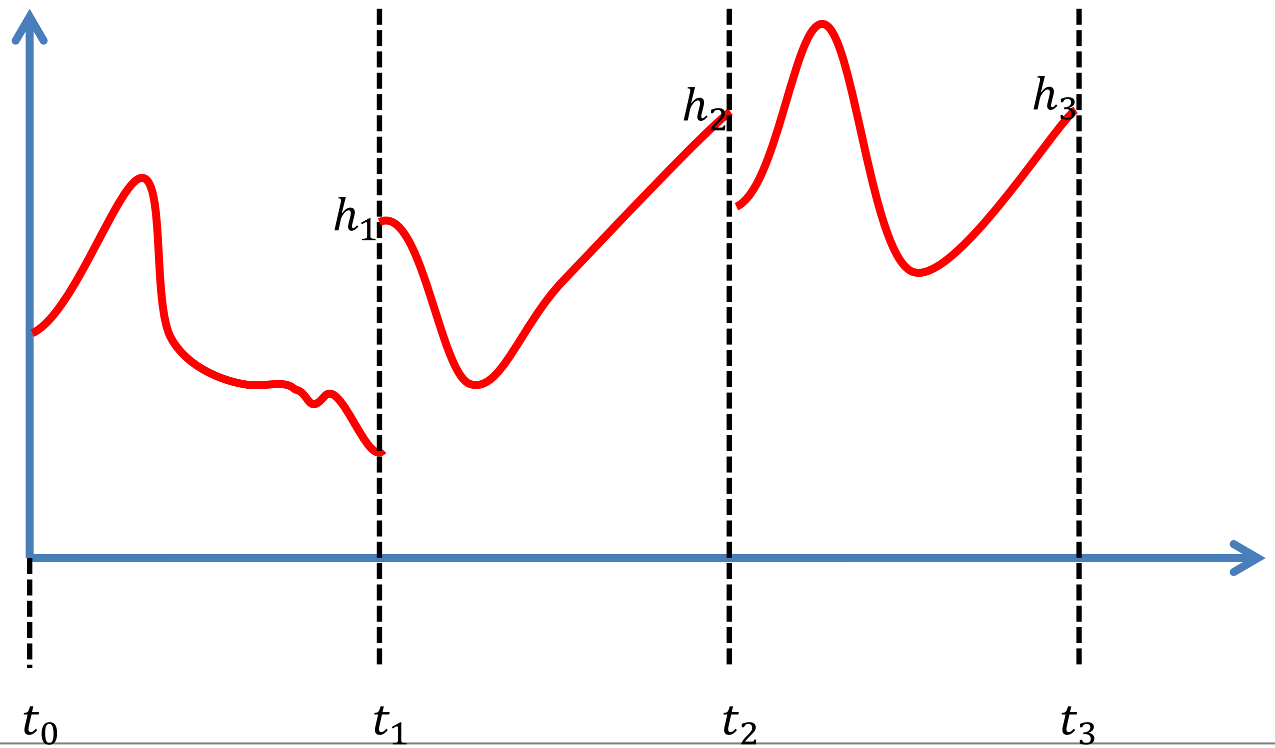


Theoretical Background

Multiple Shooting/Collocation

- Solve sub-problem in each sub-interval

$$x_i(t_{i+1}) = h_i + \int_{t_i}^{t_{i+1}} f(x_i(t), u(t), t)\, dt \approx \boxed{F(t_i, t_{i+1}, h_i, u_i)}, \qquad x_i\,(t_i) = h_i$$



Theoretical Background

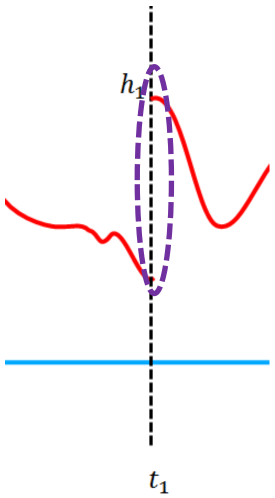
Multiple Shooting / Collocation Optimization

- Discretized Nonlinear Optimal Control Problem
  - objective function (integral approximation by trapezoidal rule)

$$\min_{u(t)} J(x(t), u(t), t) = E(h_n) + \frac{\Delta t}{2} \sum_{i=0}^{n-1} L(h_i, u_i, t_i) + L(h_{i+1}, u_i, t_{i+1})$$

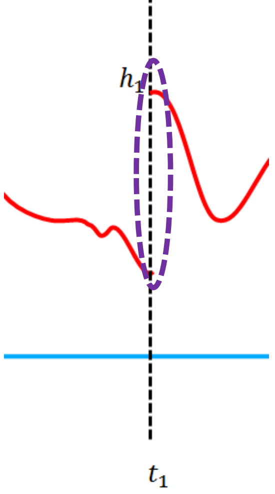
  - subject to

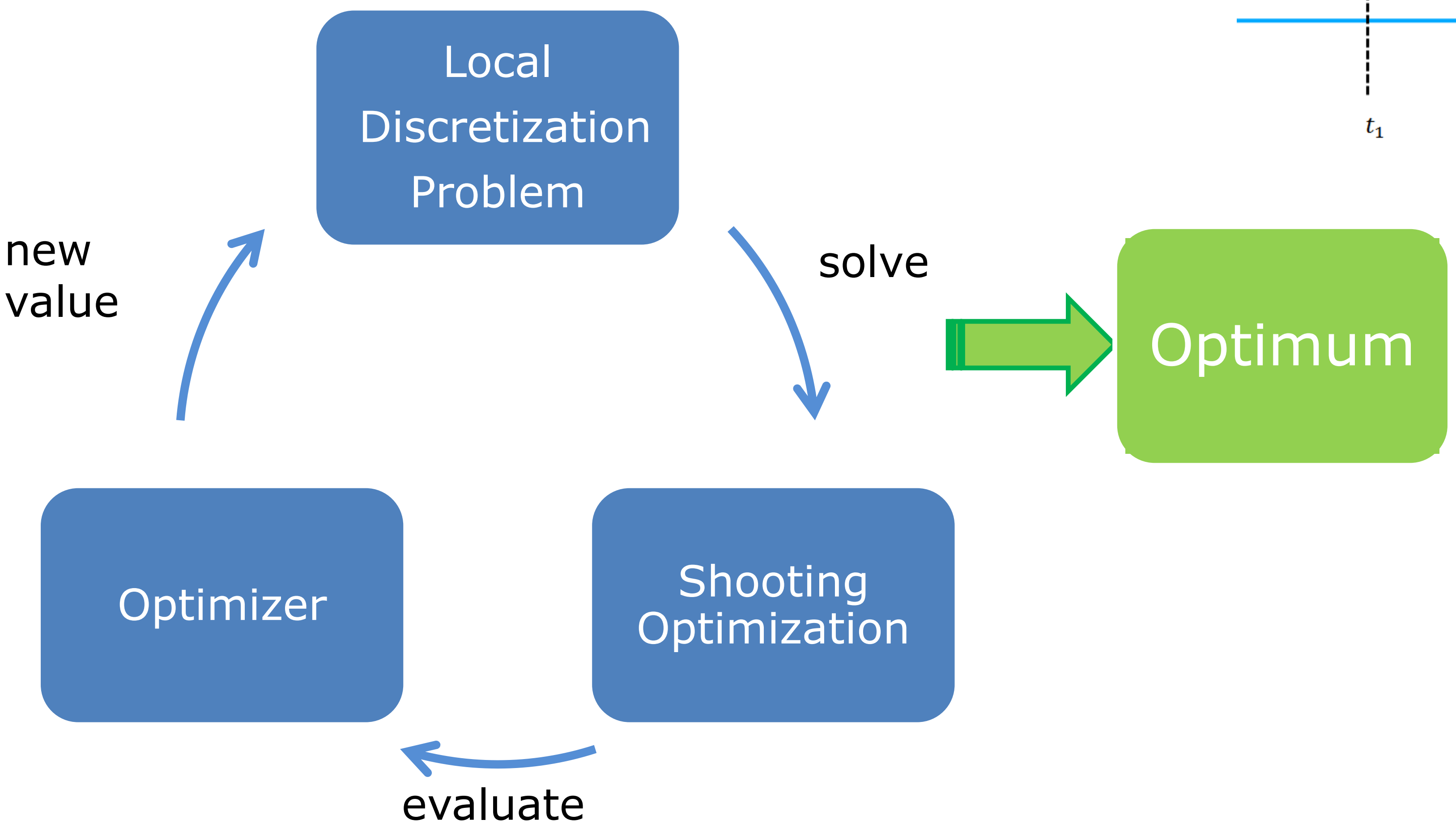
$$\begin{array}{rcl} x(t_0) & = & h_0 \\ F(t_i, t_{i+1}, h_i, u_i) & = & h_{i+1} \\ g(h_i, u_i, t_i) & \geq & 0 \\ g(h_{i+1}, u_i, t_{i+1}) & \geq & 0 \end{array}$$



Theoretical Background

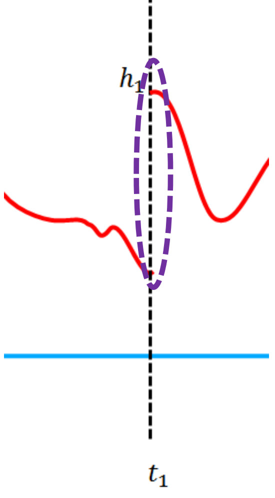
Multiple Shooting / Collocation Optimization

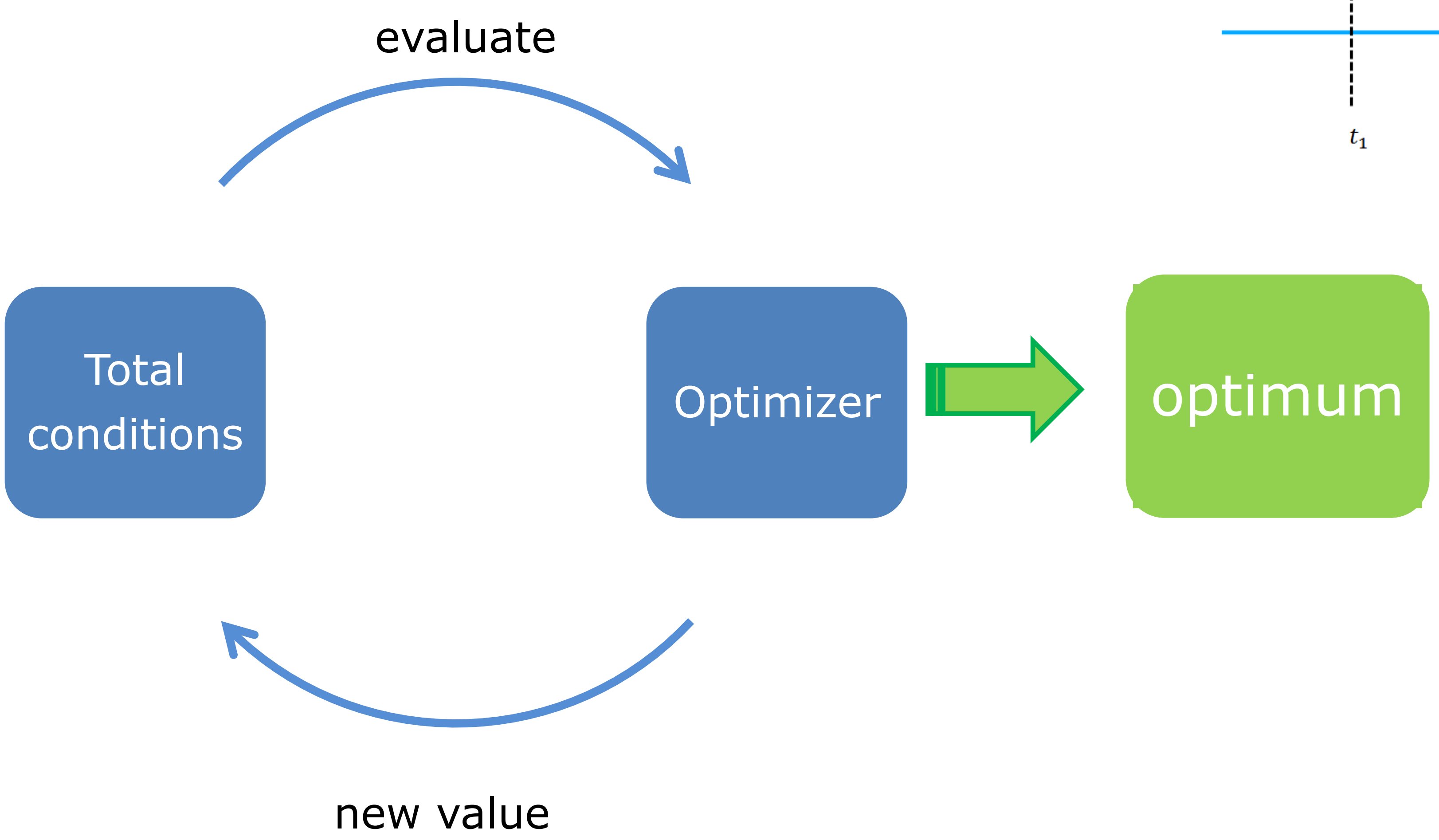




Theoretical Background

Total Collocation Optimization





Theoretical Background

Total Collocation Optimization

- Discretized Nonlinear Optimal Control Problem
  - objective function (integral approximation by Gauß quadrature)

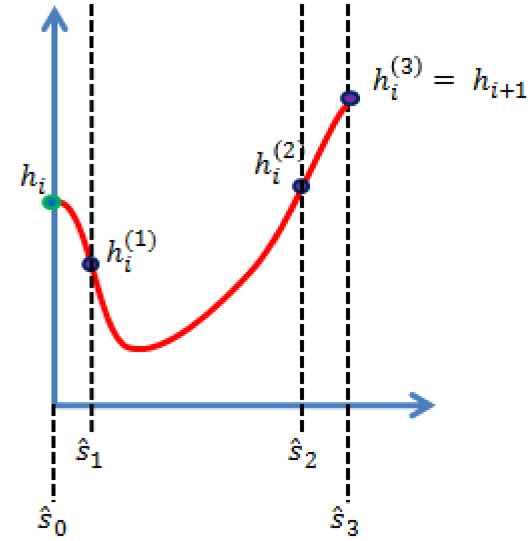
$$\min_{u(t)} J(x(t), u(t), t) = E(h_n) + \Delta t \sum_{j=0}^{m} w_j \cdot \sum_{i=0}^{n-1} L\left(h_i^{(j)}, u_i, t_i + s_j\right)$$

  - subject to

$$\begin{aligned} x(t_0) &= h_0 \\ g(h_i, u_i, t_i) &\geq 0 \\ g(h_{i+1}, u_i, t_{i+1}) &\geq 0 \end{aligned}$$

additional collocation conditions

Theoretical Background
Collocation Condition – Approximation of States



- Assumption:

  States are locally polynomial

$$x_i(t_i + \hat{s} \cdot \Delta t) = p_0(\hat{s}) \cdot h_{i-1}^{(m)} + \sum_{j=1}^{m} p_j(\hat{s}) \cdot h_i^{(j)}$$

where $x_i(t_i + \hat{s}_k \cdot \Delta t) = \delta_{k,0} \cdot h_{i-1}^{(m)} + \sum_{j=1}^{m} \delta_{k,j} \cdot h_i^{(j)} = h_i^{(k)}$

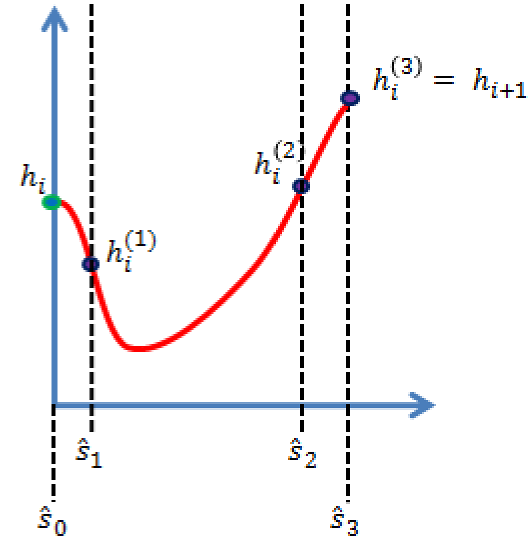
$\hat{s}_k$ are the Radau points

$p_j(\hat{s})$ are the Lagrange Basis polynomial to the nodes $\hat{s}_k$

- Collocation conditions

$$\Delta t \cdot f\left(h_i^{(j)}, u_i, t_i + \hat{s}_k \cdot \Delta t\right) = p_0'(\hat{s}_k) \cdot h_{i-1}^{(m)} + \sum_{j=1}^{m} p_j'(\hat{s}_k) \cdot h_i^{(j)}$$

Theoretical Background
Collocation Condition – Approximation of State Derivatives



- Assumption:

  State derivatives are locally polynomial

$$\Delta t \cdot f(x_i(t_i + \hat{s} \cdot \Delta t), u_i, t_i + \hat{s} \cdot \Delta t) = \sum_{j=0}^{m} p_j(\hat{s}) \cdot f_i^{(j)}$$

where $\Delta t \cdot f\left(h_i{}^{(k)}, u_i, t_i + \hat{s}_k \cdot \Delta t\right) = \sum_{j=0}^{m} \delta_{k,j} \cdot f_i{}^{(j)} = f_i{}^{(k)}$

$\hat{s}_j$ are the Lobatto points

$p_j(\hat{s})$ are the Lagrange Basis polynomial to the nodes $\hat{s}_j$

- Collocation conditions

$$h_i^{(k)} = \sum_{j=0}^{m} \int p_j(\hat{s}_k) \cdot f_i{}^{(j)} + h_{i-1}^{(m)}$$

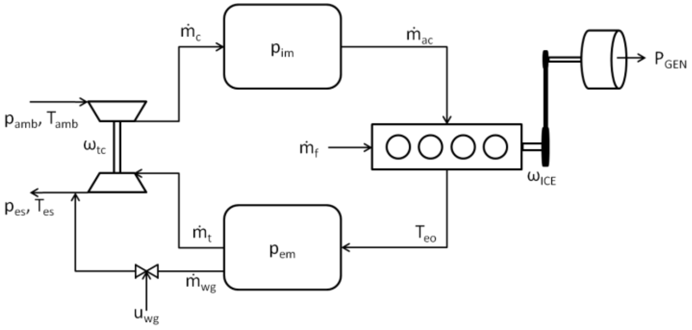
Applications – Diesel Electric Powertrain

- Find fuel optimal control and state trajectories from idling condition to a certain power level

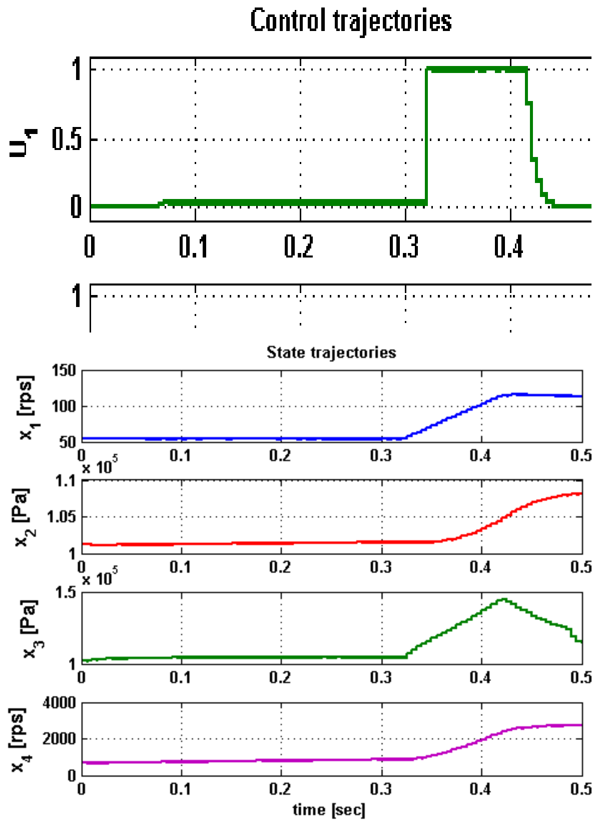
- Nonlinear mean value engine model

- Only diesel operating condition

- Mathematical problem formulation:
  - 2 inputs (u_f, u_{wg})
  - 4 states $(\omega_{ice}, p_{im}, p_{em}, \omega_{tc})$
  - 32 algebraic equations



Applications – Diesel Electric Powertrain



- Mathematical problem formulation
  - Object function

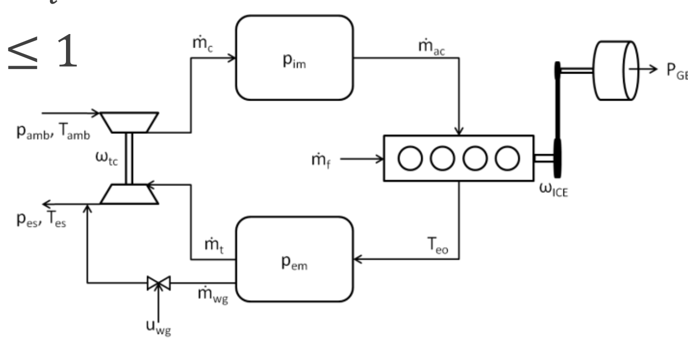
$$\min_{u(t)} \sum_{i=1}^{4} \left(x_i(t_f) - x_i^{ref}\right)^2 + \int_0^T \dot{m}_f \, dt$$

  - subject to

$$\begin{aligned}
\dot{x}_1 &= f_1(x_2, x_3, u_1) \\
\dot{x}_2 &= f_2(x_1, x_2, x_4) \\
\dot{x}_3 &= f_3(x_1, x_2, x_3, u_1, u_2) \\
\dot{x}_4 &= f_4(x_2, x_3, x_4, u_2)
\end{aligned}$$

$$x_{lb_i} \leq x_i \leq x_{ub_i}, \; i = 1, \dots, 4$$

$$0 \leq u_1, u_2 \leq 1$$



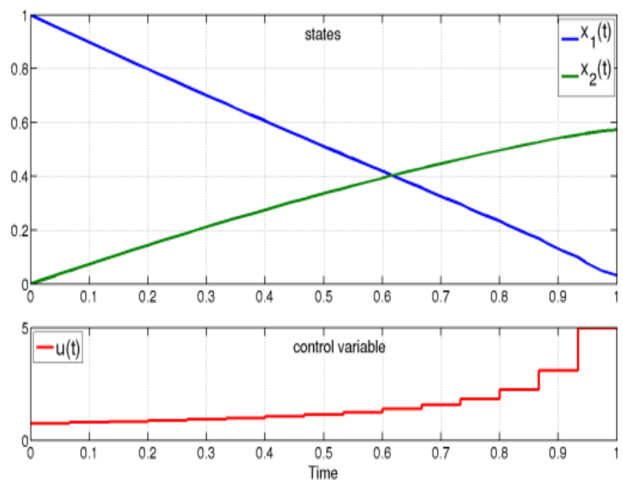
Engine is accelerated only near the end of the time interval to meet the end constraints while minimizing the fuel consumption

Implementation Details – Current Status

- Realization with OpenModelica Environment
- Optimica prototype implementation is available
- Using Ipopt for solution process
- Necessary derivatives are numerically calculated
  - Gradients, Jacobians, Hessians, …
- **But:** Complete tool chain not yet implemented

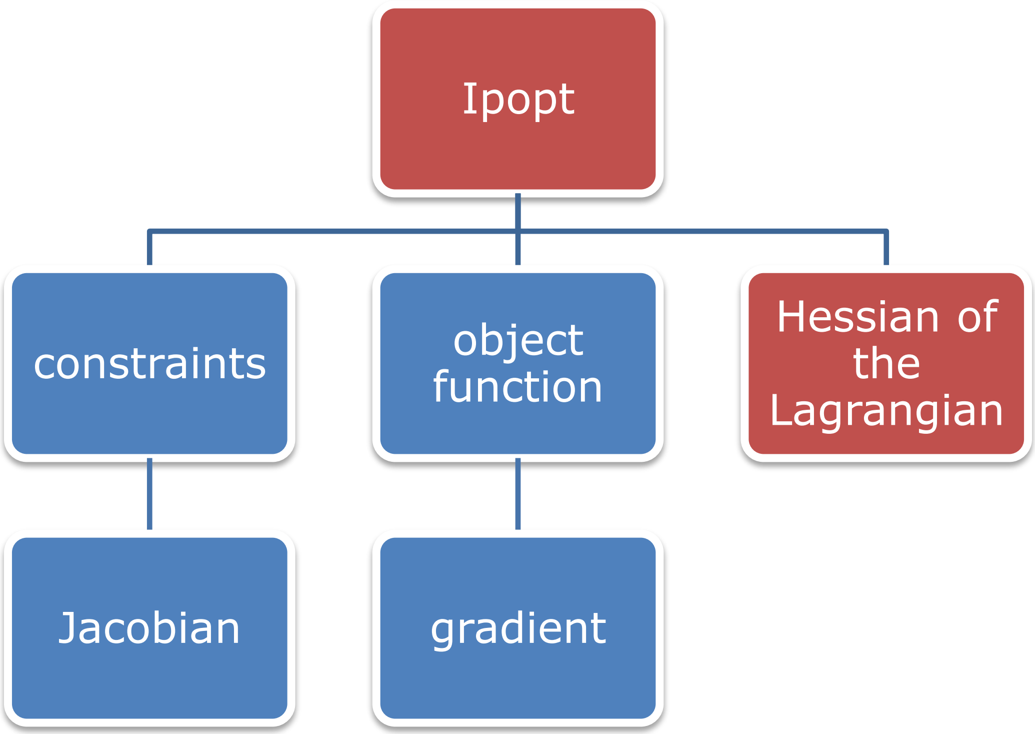
Test Environment

- Processor:
- 2xIntel Xeon CPU E5-2650
  16 cores @ 2.00GHz
- OpenMP



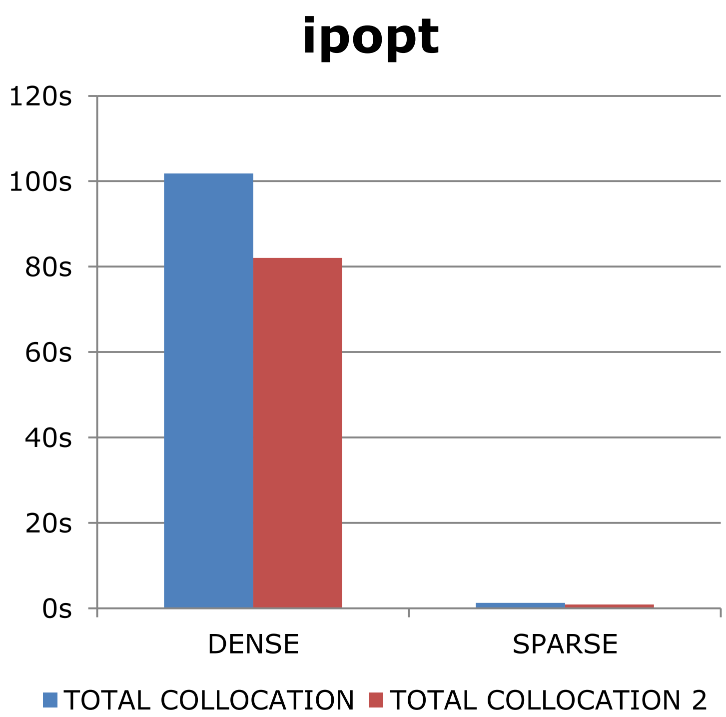
Implementation Details – Ipopt & Parallelization

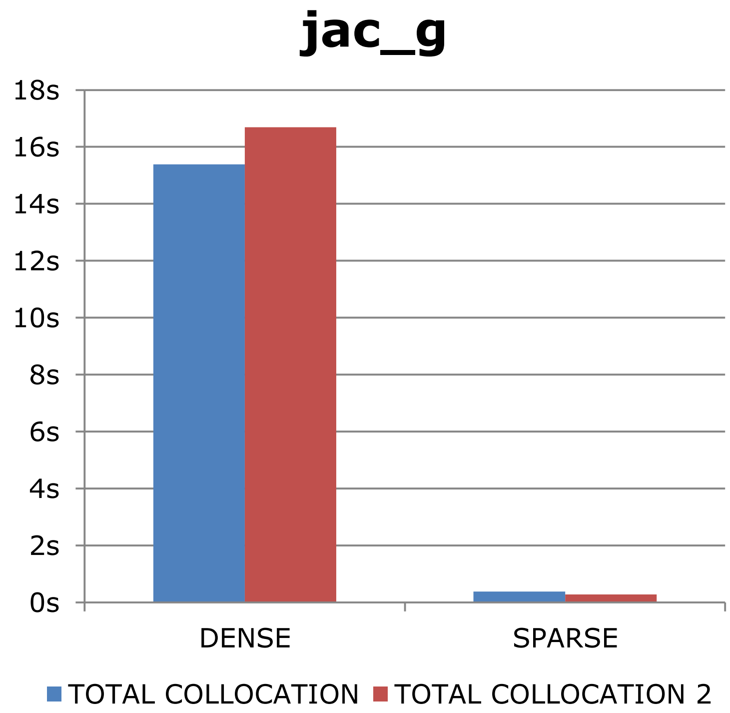
- Schematic view of the required components of Ipopt



Implementation Details - Numerical Optimization

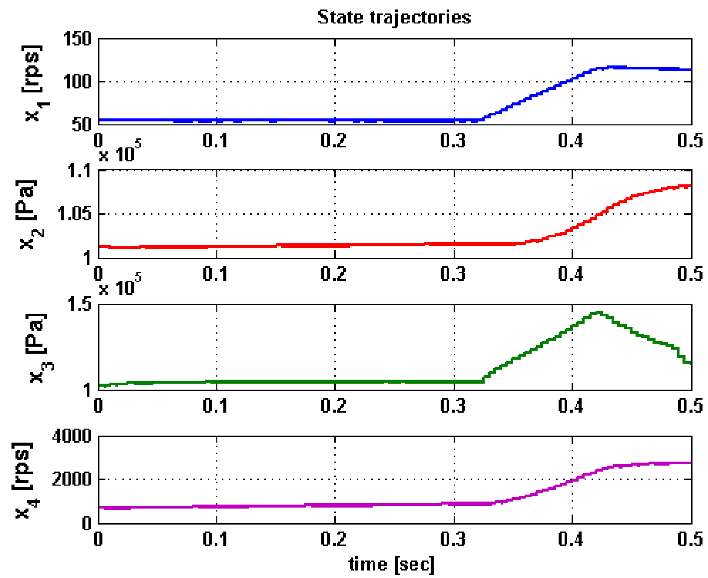
- Enormous speed-up when utilizing sparse Jacobian matrix
- Speed-up for the over-all optimization
- Sparse-structure model independent





Results - Diesel Electric Powertrain

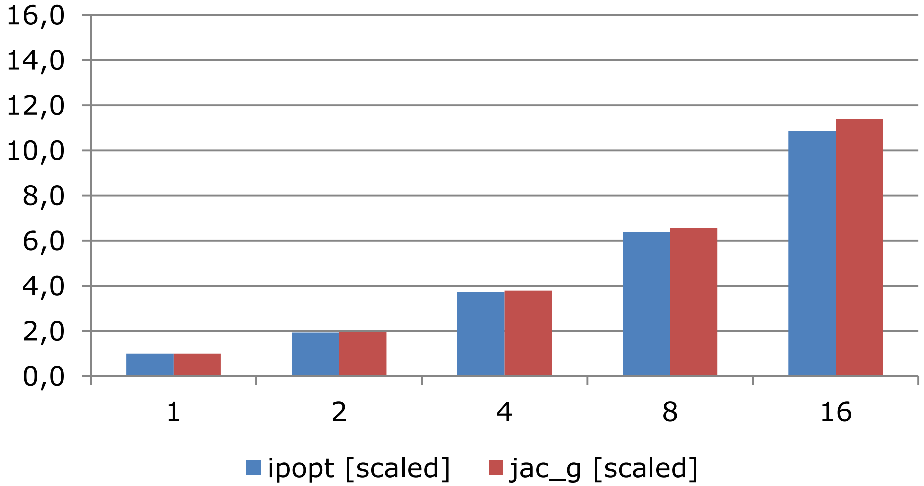
	Diesel
MULTIPLE SHOOTING	921,6s
MULTIPLE COLLOCATION	29519,8s
TOTAL COLLOCATION	9,5s
TOTAL COLLOCATION 2	15,6s



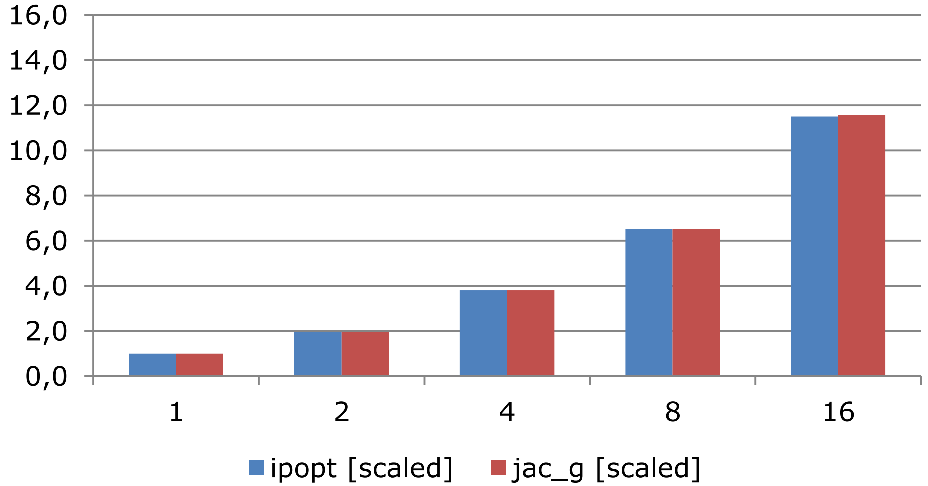
Results - Diesel Electric Powertrain

- Ipopt runs in serial mode
- Most execution time is elapsed in Jacobian calculation and solution process of the local discretization problem
- Reasonable speed-up
- Factors are non-optimal due to memory handling
  - Further investigations will be performed

MULTIPLE_SHOOTING

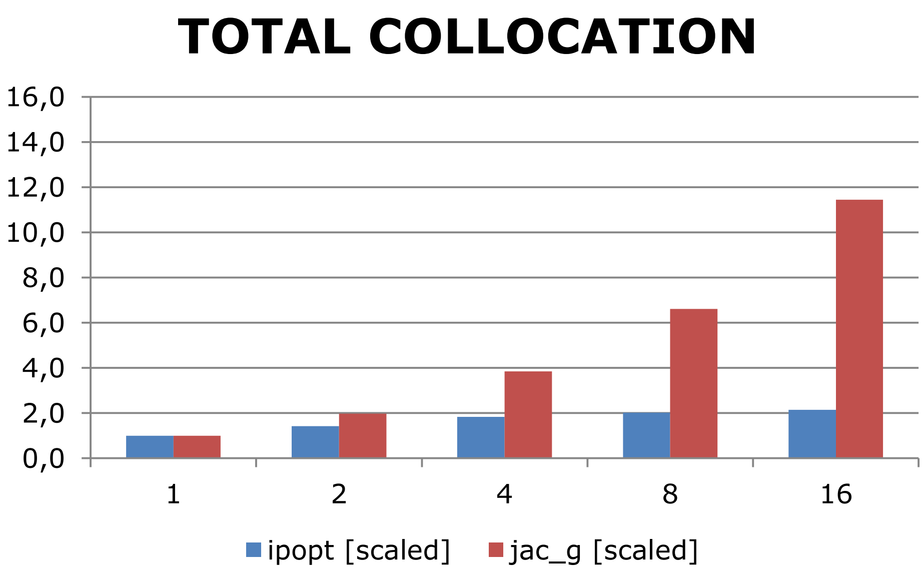


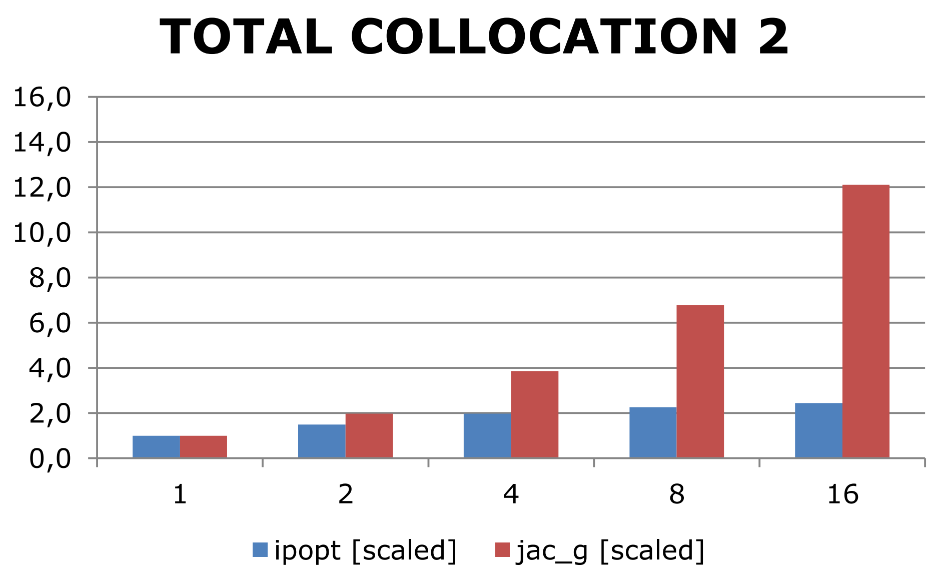
MULTIPLE_COLLOCATION



Results - Diesel Electric Powertrain

- Ipopt runs in serial mode
- Less execution time is elapsed in Jacobian calculation
- Reasonable speed-up for Jacobian calculation
- Factors are non-optimal due to memory handling
- Overall Speed-up increases with model complexity
  - Parallelizing of Ipopt necessary





Lessons learned

- Symbolic calculation of derivatives improve performance
  - Jacobian, Gradient, …
- Utilizing sparsity pattern is crucial
- In serial mode total collocation methods are superior to multiple shooting/collocation methods
- Parallelizing the algorithms performs better on multiple shooting/collocation methods
- Symbolic transformation to ODE form is a key issue for the realization of an automatic tool chain

Example 1 – From the dark side (Francesco Casella)

```
optimization Example1A(
  objective = a1/b1*(x1 - x10)^2 +
              a2/b2*(x2 - x20)^2 +
              a3/b3*(x3 - x30)^2,
...
end Example1A;
```

```
optimization Example1B(
  objective = f1 + f2 + f3,
...
equation
  f1 = a1/b1*(x1 - x10)^2;
  f2 = a2/b2*(x2 - x20)^2;
  f3 = a3/b3*(x3 - x30)^2;
...
end Example1B;
```

Example 2 – From the dark side (Francesco Casella)

```
optimization Example2A(
  parameter Real PR = 10;
...
equation
  p_in/p_out = PR "Turbine pressure ratio";
...
end Example1A;
```

```
optimization Example2A(
  parameter Real PR = 10;
...
equation
  p_in = PR*p_out "Turbine pressure ratio";
...
end Example1A;
```

Future work

- Implement complete tool chain in OpenModelica
- Automatic generation of simulation code based on Optimica
- Utilizing symbolically derived derivative information
  - Gradient, Jacobian, Hessian, …
- Further improvements with appropriate scaling
- Exploiting parallel evaluation of the optimization method
- Advanced use of OMC symbolic machinery
  - Efficient handling of model dependent algebraic loops
- Generalization of NOCP problem formulation
  - e.g. time minimal optimization, parameter estimation

- Further testing on industrial-relevant problems

Thank you

Questions?