Modelica based optimization - state of the art and future challenges

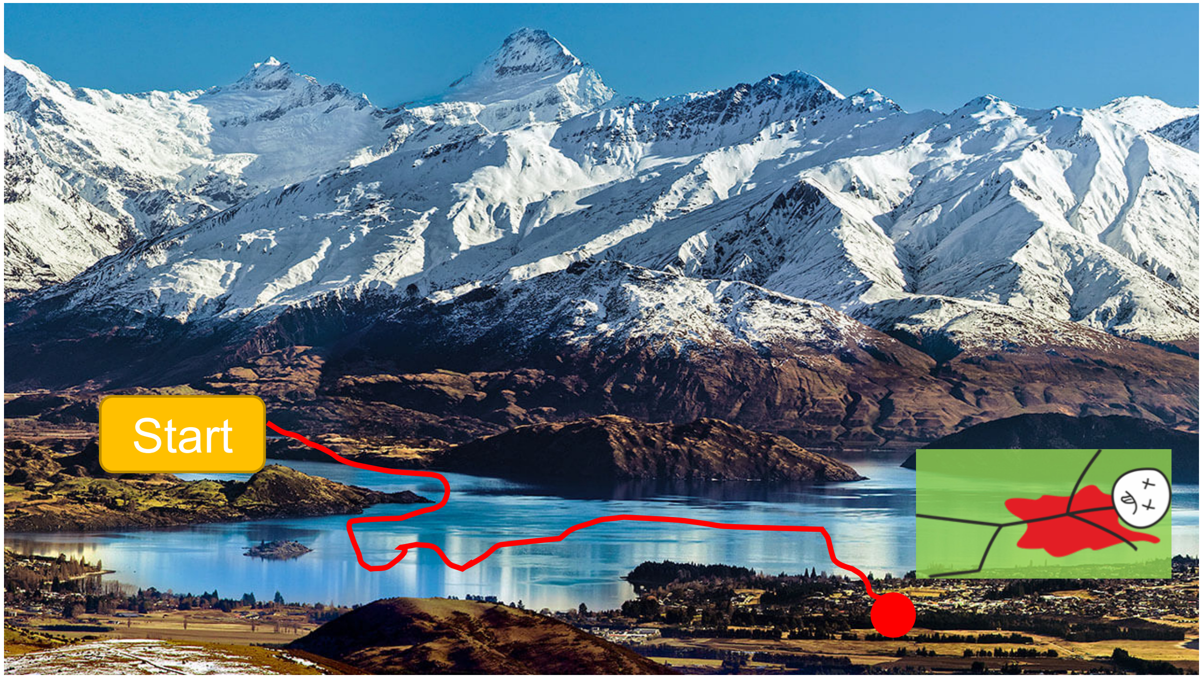
Gerald Schweiger, David Blum, Michael Wetter

Optimization is **inherently more difficult** than simulation, putting higher requirements on both **users and tool developers**

Optimization is **inherently more difficult** than simulation, putting higher requirements on both **users and tool developers**

Optimization is **inherently more difficult** than simulation, putting higher requirements on both **users and tool developers**

Optimization is **inherently more difficult** than simulation, putting higher requirements on both **users and tool developers**

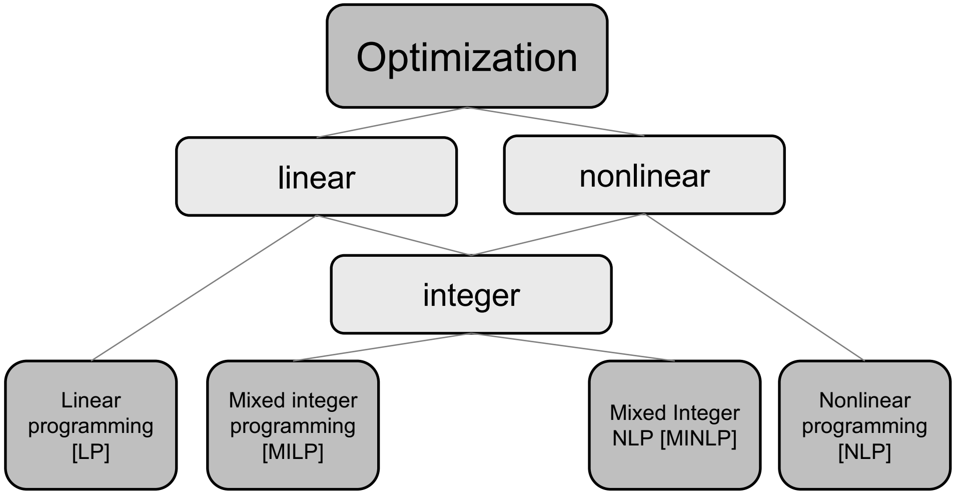


Optimization is **inherently more difficult** than simulation, putting higher requirements on both **users and tool developers**

…"in practice there are relatively few optimization problems that can be solved efficiently. In many cases we can only hope to find a good-enough local optimum in finite search time, trading off between climbing hills in one place and looking for places that might have better hills to climb"

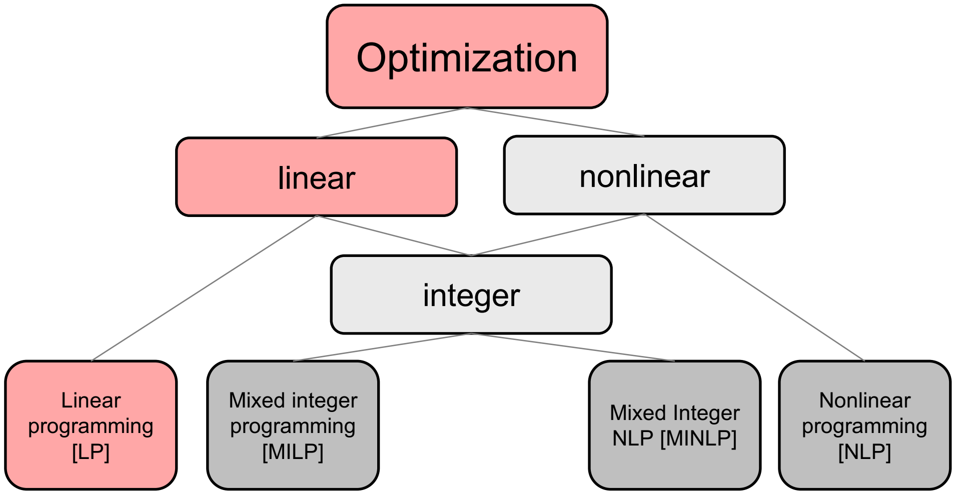
Optimization

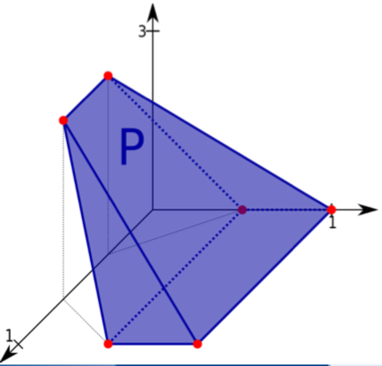
Mathematical optimization is the selection of a “best” element from some set of available alternatives.



Optimization

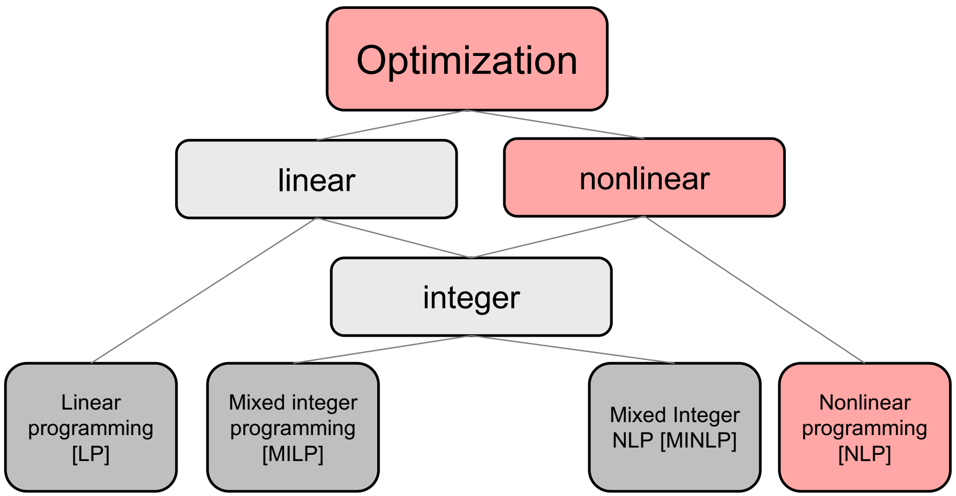
Mathematical optimization is the selection of a "best" element from some set of available alternatives.

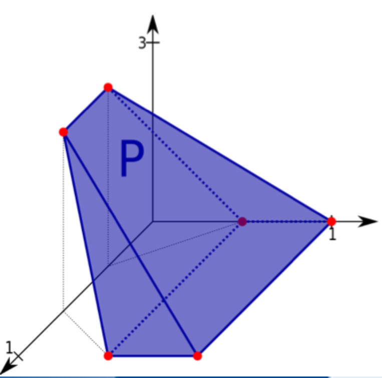


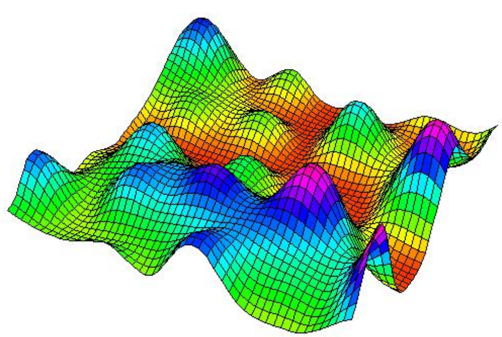


Optimization

Mathematical optimization is the selection of a “best” element from some set of available alternatives.

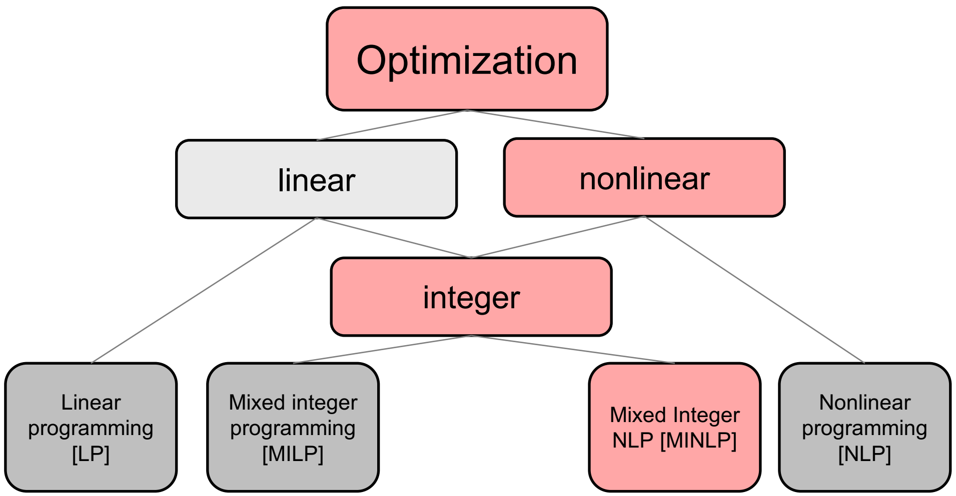


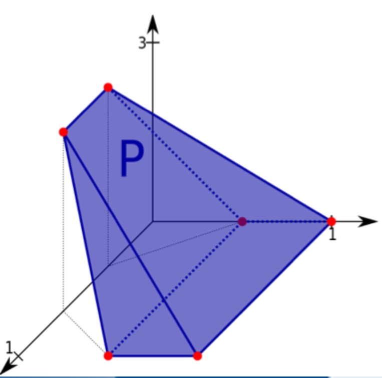


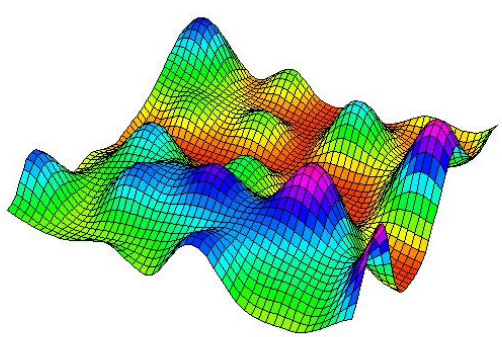


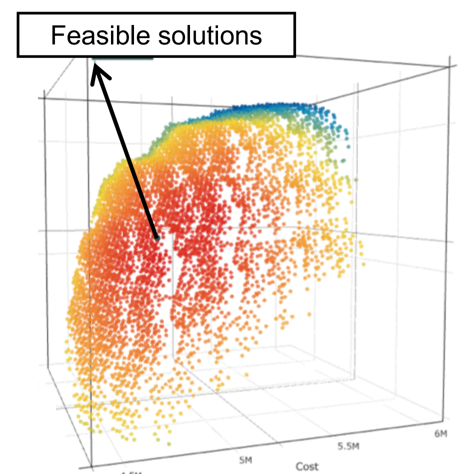
Optimization

[Mathematical optimization] is the selection of a “best” element from some set of available alternatives.



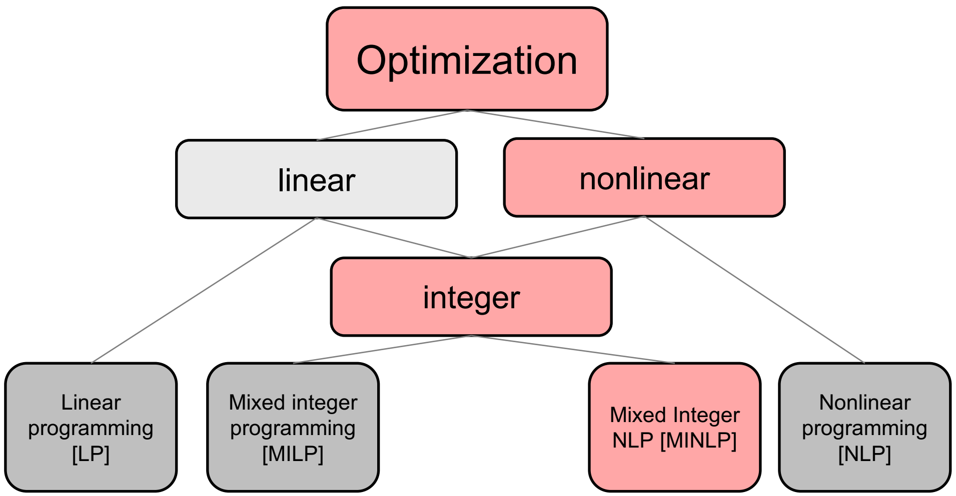


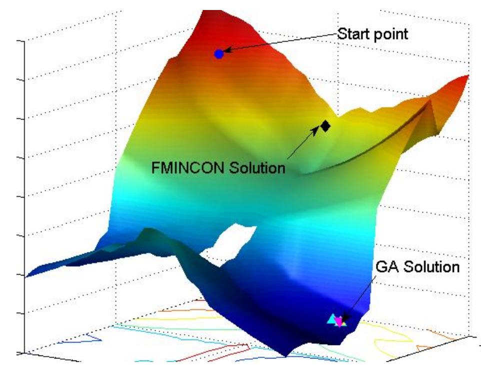


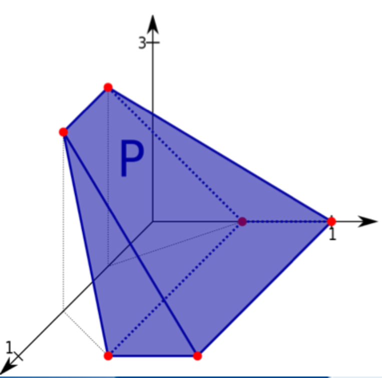


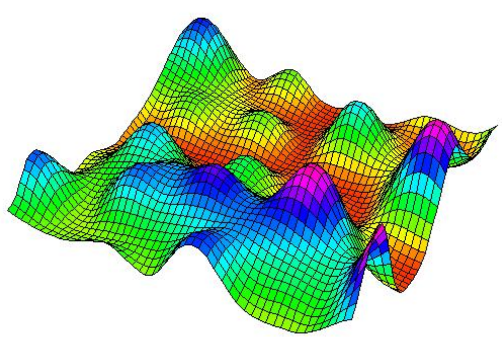
Optimization

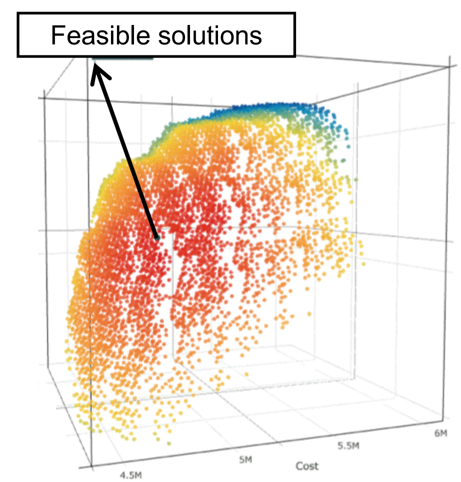
Mathematical optimization is the selection of a "best" element from some set of available alternatives.











Optimization

Static optimization:

- Finding the **optimal point** (Euclidean space) to optimize a given objective function.

Dynamic optimization:

- Finding the **optimal control trajectories** over a time horizon to optimize a given objective function.

  - Trajectory optimization
  - Optimal control (=Optimalsteuerung)
  - Dynamic optimization/programming
  - MPC (Model Predictive Control). MPC is a control method based on repeated optimal control

General form of a dynamic optimization problem

$$\underset{u(t)}{\text{minimize}} \quad \phi\left(t_f, x\left(t_f\right)\right) + \int_{t_0}^{t_f} L(t, x(t), u(t))dt$$

$$s.t. F\left(t, \dot{x}(t), x(t), u(t)\right) = 0$$
$$x(0) = x_0$$
$$\text{Path constraints: } g_i(x(t), u(t) \leq 0$$
$$\text{Point constraints: } g_e(x(t), u(t) = 0$$

Solution strategies

General form of a dynamic optimization problem

$$\underset{u(t)}{\text{minimize}} \quad \phi\left(t_f, x\left(t_f\right)\right) + \int_{t_0}^{t_f} L(t, x(t), u(t))dt$$

$$s.t. F\left(t, \dot{x}(t), x(t), u(t)\right) = 0$$

$$x(0) = x_0$$

$$\text{Path constraints: } g_i(x(t), u(t) \leq 0$$

$$\text{Point constraints: } g_e(x(t), u(t) = 0$$

- Solution strategies
  - Dynamic programming
  - Indirect methods
  - Direct methods

General form of a dynamic optimization problem

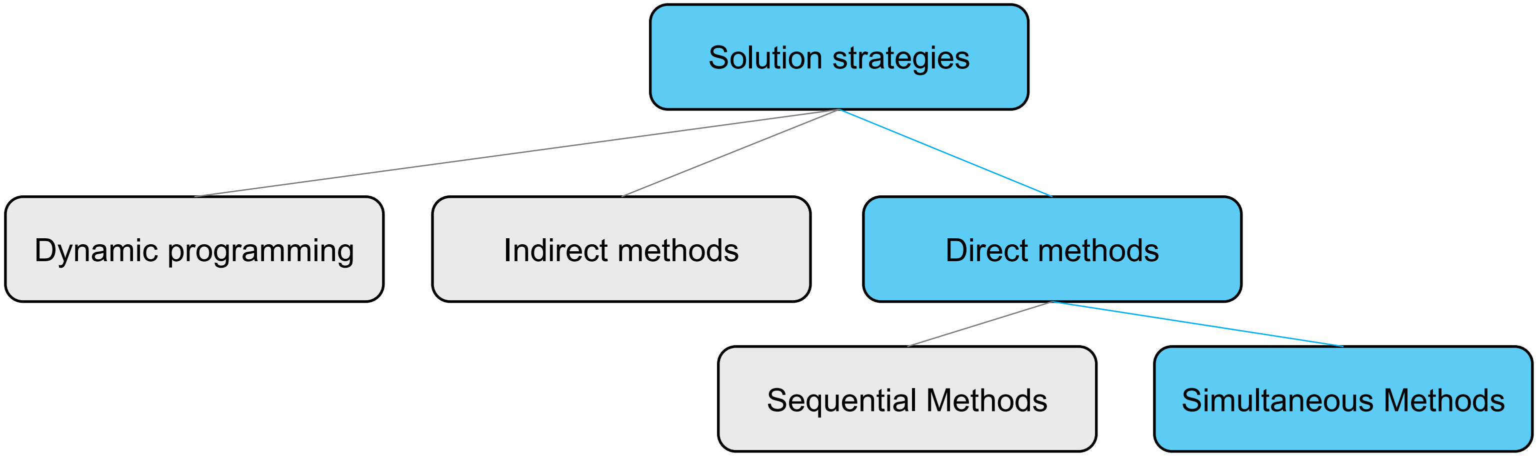
$$\underset{u(t)}{\text{minimize}} \quad \phi\left(t_f, x\left(t_f\right)\right) + \int_{t_0}^{t_f} L(t, x(t), u(t))dt$$

$$s.t. F\left(t, \dot{x}(t), x(t), u(t)\right) = 0$$

$$x(0) = x_0$$

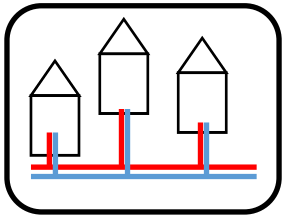
$$\text{Path constraints: } g_i(x(t), u(t) \leq 0$$

$$\text{Point constraints: } g_e(x(t), u(t) = 0$$



Direct collocation

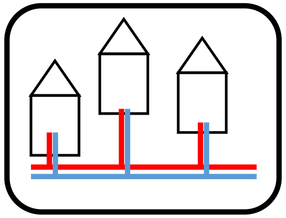
Continuous problem



$$\underset{x,y,u,p}{\text{minimize}} \quad \Phi(t_f, x(t_f)) + \int_{t_0}^{t_f} L(x(t), y(t), u(t))dt$$

Direct collocation

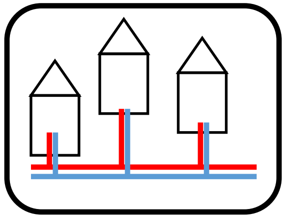
Continuous problem



$\underset{x,y,u,p}{\text{minimize}}$ **Supply temperature** $(t))dt$

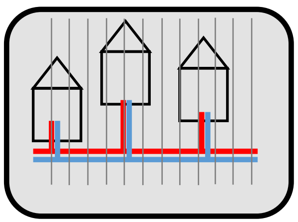
Direct collocation

Continuous problem



Discretize infinite-dimensional problem using direct collocation to get a finite time problem

[Discretization](#)



Polynomials approximate the variables of a **DAE**

The polynomials use the Lagrange basis polynomials, and they use the collocation points as the interpolation points.

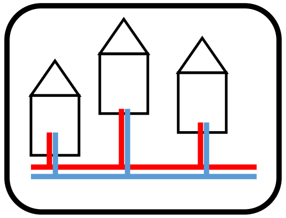
$$x_i(\tau) = \sum_{k=0}^{n_c} x_i \, \widetilde{l_k}(\tau), \quad y_i(\tau) = \sum_{k=1}^{n_c} y_i \, l_k(\tau),$$

$$\dot{x}_i(\tau) = \frac{dx_i}{d\widetilde{t}_i}(\tau) = \frac{d\tau}{d\widetilde{t}_i}\frac{dx_i}{d\tau}(\tau) = \frac{1}{h_i}\sum_{k=0}^{n_c} x_{i,k}\frac{d\widetilde{l_k}}{d\tau}(\tau)$$

$$\widetilde{l_k}(\tau) = \prod_{\substack{l=0,\dots,n_c \\ l\neq k}} \frac{\tau-\tau_l}{\tau_k-\tau_l}, \quad l_k(\tau) = \prod_{\substack{l=1,\dots,n_c \\ l\neq k}} \frac{\tau-\tau_l}{\tau_k-\tau_l}$$

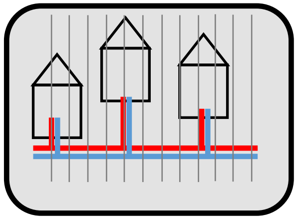
Direct collocation

Continuous problem



Discretize infinite-dimensional problem using direct collocation to get a finite time problem

Discretization



This leads to a nonlinear program (NLP)

Polynomials approximate the variables of a **DAE**

The polynomials use the Lagrange basis polynomials, and they use the collocation points as the interpolation points.

$$x_i(\tau) = \sum_{k=0}^{n_c} x_i\, \tilde{l}_k\,(\tau), \quad y_i(\tau) = \sum_{k=1}^{n_c} y_i\, l_k\,(\tau),$$

$$\dot{x}_i(\tau) = \frac{dx_i}{d\tilde{t}_i}(\tau) = \frac{d\tau}{d\tilde{t}_i}\frac{dx_i}{d\tau}(\tau) = \frac{1}{h_i}\sum_{k=0}^{n_c} x_{i,k}\frac{d\tilde{l}_k}{d\tau}(\tau)$$

$$\tilde{l}_k(\tau) = \prod_{\substack{l=0,\dots,n_c\\ l\neq k}} \frac{\tau-\tau_l}{\tau_k-\tau_l}, \quad l_k(\tau) = \prod_{\substack{l=1,\dots,n_c\\ l\neq k}} \frac{\tau-\tau_l}{\tau_k-\tau_l}$$

The polynomials are defined on a finite number of collocation points. Hence, they **convert the infinite to a finite dimensional** optimization problem, which can be solved by a NLP solver

Why Modelica?

General Trends

- Previous studies have reported that the **time spending for model development** is significantly high (up to **80%**) [1-5]
- **70% of project costs** are consumed by **model creation** and calibration [6-8]
- There is an urgent need for **automated model creation** for optimization [10-12]

General Trends

- Previous studies have reported that the **time spending for model development** is significantly high (up to **80%**) [1-5]
- **70% of project costs** are consumed by **model creation** and calibration [6-8]
- There is an urgent need for **automated model creation** for optimization [10-12]

Modelica

Modelica language

Declarative language

- Allow acausal modelling
- The order of the equations does not matter.

Multi-domain modelling

- Modelica is a multi-domain language, not geared towards any specific domain. Easily couple models containing for example, mechanical and electrical components.

Object-oriented

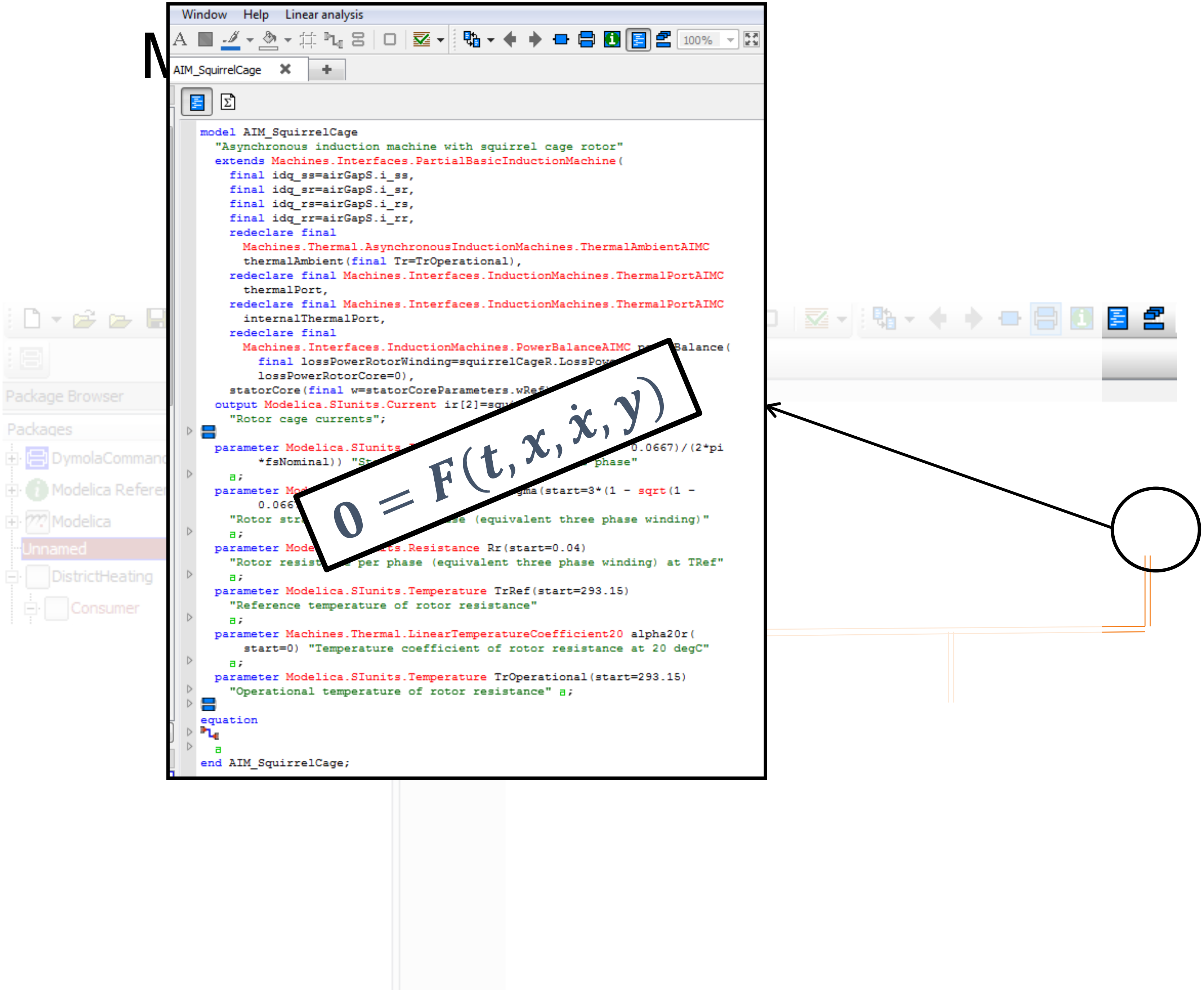
- Models are classes and thus can easily be extended using ordinary object-oriented features.

Visual component programming

- Hierarchical system architecture capabilities.

Fully implicit DAE modelling approach

$$\mathbf{0} = \boldsymbol{F}(\boldsymbol{t}, \boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{y})$$


Window Help Linear analysis
AIM_SquirrelCage
model AIM_SquirrelCage
"Asynchronous induction machine with squirrel cage rotor"
extends Machines.Interfaces.PartialBasicInductionMachine(
final idq_ss=airGapS.i_ss,
final idq_sr=airGapS.i_sr,
final idq_rs=airGapS.i_rs,
final idq_rr=airGapS.i_rr,
redeclare final
Machines.Thermal.AsynchronousInductionMachines.ThermalAmbientAIMC
thermalAmbient(final Tr=TrOperational),
redeclare final Machines.Interfaces.InductionMachines.ThermalPortAIMC
thermalPort,
redeclare final Machines.Interfaces.InductionMachines.ThermalPortAIMC
internalThermalPort,
redeclare final
Machines.Interfaces.InductionMachines.PowerBalanceAIMC
lossPowerRotorCore=0),
"Rotor cage currents";
"Reference temperature of rotor resistance"
parameter Machines.Thermal.LinearTemperatureCoefficient20 alpha20r(
start=0) "Temperature coefficient of rotor resistance at 20 degC"
parameter Modelica.SIunits.Temperature TrOperational(start=293.15)
"Operational temperature of rotor resistance"
equation
end AIM_SquirrelCage;
$0 = F(t, x, \dot{x}, y)$
Package Browser
Packages
DymolaCommand
Modelica
Unnamed
DistrictHeating
Consumer

Modelica for optimization

Modelica is mainly intended for simulation

An extension to accommodate dynamic optimization problems is required

- Optimica: defines new syntax and semantics for specifying constraints and an objective
- Currently supported by JModelica.org, OpenModelica and IDOS

Modelica for optimization

Modelica is mainly intended for simulation

An extension to accommodate dynamic optimization problems is required

- Optimica: defines new syntax and semantics for specifying constraints and an objective
- Currently supported by JModelica.org, OpenModelica and IDOS

Modelica for optimization

Modelica is mainly intended for simulation

An extension to accommodate dynamic optimization problems is required

- Optimica: defines new syntax and semantics for specifying constraints and an objective
- Currently supported by JModelica.org, OpenModelica and IDOS

OpenModelica JMODELICA.ORG

Restrictions

- The most widely used numerical techniques for dynamic optimization are based on first-order necessary conditions for local optimality
  - The objective function and the DAE System must be twice continuously differentiable
  - Hybrid constructs are excluded!!!

$$\underset{u(t)}{\text{minimize}} \quad \int_{t_0}^{t_f} L(t, x(t), u(t))dt$$

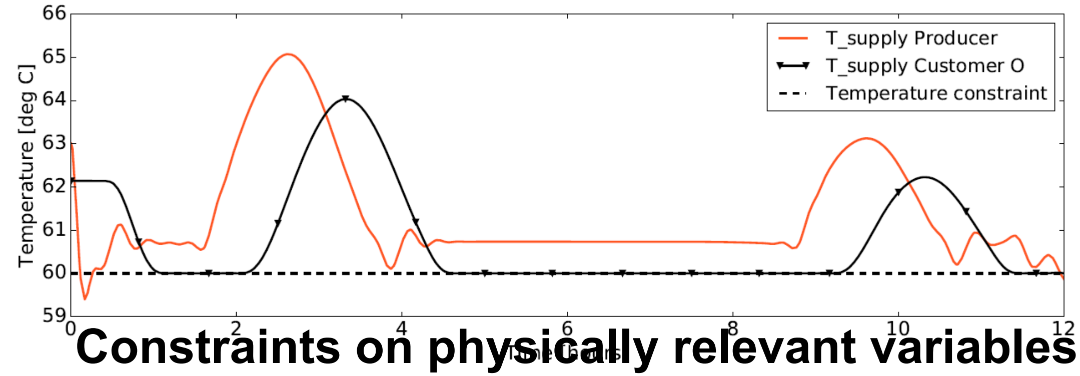
$$s.t.\, F\big(t, \dot{x}(t), x(t), y(t), u(t)\big) = 0$$

$$g_i(x(t), u(t) \leq 0$$
$$g_e(x(t), u(t) = 0$$

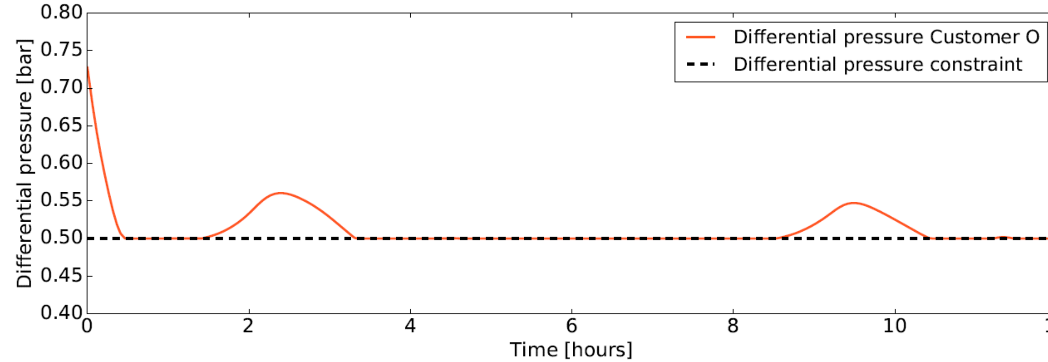
Applications in the energy sector

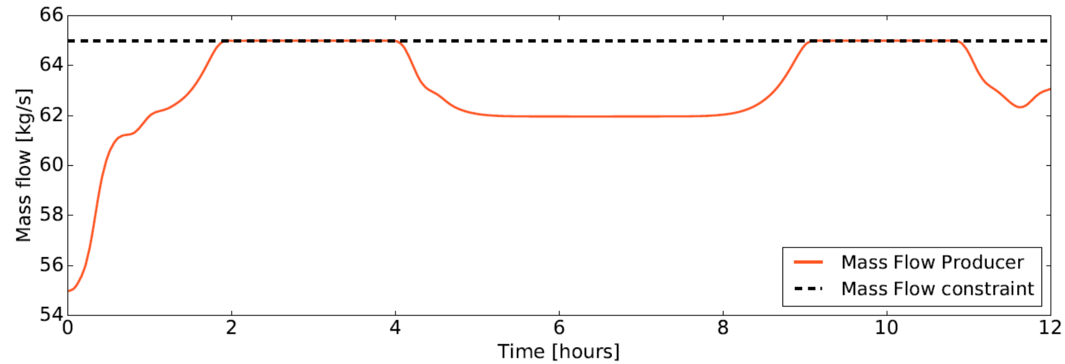
- Optimal use of an electrochemical storage tank in combination with a MV photovoltaic
- Optimal start of a steam boiler
- Model predictive control of buildings
- Model predictive control of district heating systems
- Optimal start of a combined-cycle power plant

Dynamic Optimization of a district heating system



Constraints on physically relevant variables





$$
\begin{aligned}
\text{min.} \quad & \int_{t_0}^{t_f} (\alpha T_{\text{prod}} + \beta dp_{\text{prod}} + \gamma \dot{Q}_{\text{prod}}^2 + \delta dp_{\text{prod}}^2) \mathrm{d}t, \\
\text{s.t.} \quad & \text{model dynamics}, \\
& m_{Prod}(t) \le m_{Prod}^U \quad \forall t \in [t_0, t_f], \\
& T_{Customer}^L \le T_{Customer}(t) \quad \forall t \in [t_0, t_f], \\
& dp_{Customer}^L \le dp_{Customer}(t) \quad \forall t \in [t_0, t_f],
\end{aligned}
$$

- Complex dynamical effects
  - Producer increases supply temperature in advance
  - Temperature front propagates slowly
  - Customers close to producer benefit quickly from increased temperature
  - Customers at the periphery benefit from higher flowrates

- Optimization can handle
  - Flow-dependent delays
  - Slow temperature dynamics
  - Fast pressure/flow dynamics
  - Distributed network

Challenges for the community

- More Solver for dynamic optmization

Challenges for the community

- More Solver for dynamic optmization

- **Mixed-integer-optimal control problems**

Challenges for the community

- More Solver for dynamic optmization
- **Mixed-integer-optimal control problems**
- More domain specific libraries

Challenges for the community

- More Solver for dynamic optmization
- **<u>Mixed-integer-optimal control problems</u>**
- More domain specific libraries
- Scalability studies

AEE INTEC

IDEA TO ACTION

Thank you
for your Attention