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

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

\[
s.t. \ F(t, \dot{x}(t), x(t), u(t)) = 0
\]
\[
x(0) = x_0
\]
Path constraints: \( g_i(x(t), u(t)) \leq 0 \)
Point constraints: \( g_e(x(t), u(t)) = 0 \)

Solution strategies
General form of a dynamic optimization problem

\[
\text{minimize } \phi \left( t_f, x(t_f) \right) + \int_{t_0}^{t_f} L(t, x(t), u(t)) dt \\
\text{s.t. } F\left(t, \dot{x}(t), x(t), u(t) \right) = 0 \\
x(0) = x_0 \\
\text{Path constraints: } g_i(x(t), u(t) \leq 0 \\
\text{Point constraints: } g_e(x(t), u(t) = 0
\]

Solution strategies

- Dynamic programming
- Indirect methods
- Direct methods
General form of a dynamic optimization problem

\[
\begin{align*}
\text{minimize} & \quad \phi \left(t_f, x(t_f)\right) + \int_{t_0}^{t_f} L(t, x(t), u(t)) dt \\
\text{s.t.} & \quad F(t, x(t), u(t)) = 0 \\
& \quad x(0) = x_0 \\
& \text{Path constraints: } g_i(x(t), u(t)) \leq 0 \\
& \text{Point constraints: } g_e(x(t), u(t)) = 0
\end{align*}
\]

Solution strategies

- Dynamic programming
- Indirect methods
- Direct methods
- Sequential Methods
- Simultaneous Methods
Direct collocation

Continuous problem

minimize \( \Phi(t_f, x(t_f)) + \int_{t_0}^{t_f} L(x(t), y(t), u(t)) \, dt \)
Direct collocation

Continuous problem

\[
\text{minimize } x, y, u, p \\
\int \text{Supply temperature } (t) dt
\]
Direct collocation

Polynomials approximate the variables of a DAE

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

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

\[ \dot{x}_i(\tau) = \frac{d x_i}{d \bar{\tau}_i}(\tau) = \frac{d \tau}{d \bar{\tau}_i} \frac{d x_i}{d \tau}(\tau) = \frac{1}{h_i} \sum_{k=0}^{n_c} x_{i,k} \frac{d \bar{l}_k}{d \tau}(\tau) \]

\[ \bar{l}_k(\tau) = \prod_{l=0}^{n_c} \left( \frac{\tau - \tau_l}{\tau_k - \tau_l} \right) \quad l \neq k, \quad l_k(\tau) = \prod_{l=1}^{n_c} \left( \frac{\tau - \tau_l}{\tau_k - \tau_l} \right) \quad l \neq k \]
Direct collocation

Polynomials approximate the variables of a DAE

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

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

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

\[
\tilde{l}_k(\tau) = \prod_{l=0, l\neq k}^{n_c} \frac{\tau - \tau_l}{\tau_k - \tau_l}, \quad l_k(\tau) = \prod_{l=1, l\neq k}^{n_c} \frac{\tau - \tau_l}{\tau_k - \tau_l}
\]

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]
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

\[ 0 = F(t, x, \dot{x}, y) \]
Modelica modelling

\[ 0 = F(t, x, \dot{x}, y) \]
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 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 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

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!!!
\[
\begin{align*}
\text{minimize} & \quad \int_{t_0}^{t_f} L(t, x(t), u(t)) dt \\
\text{s.t.} & \quad F(t, \dot{x}(t), x(t), y(t), u(t)) = 0 \\
& \quad g_i(x(t), u(t)) \leq 0 \\
& \quad g_e(x(t), u(t)) = 0
\end{align*}
\]
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

\[ \min \int_{t_0}^{t_f} (\alpha T_{\text{prod}} + \beta d p_{\text{prod}} + \gamma Q^2_{\text{prod}} + \delta d p^2_{\text{prod}}) dt, \]

s.t. model dynamics,
\[ m_{\text{Prod}}(t) \leq m^U_{\text{Prod}} \ \forall t \in [t_0, t_f], \]
\[ T^{L}_{\text{Customer}} \leq T_{\text{Customer}}(t) \ \forall t \in [t_0, t_f], \]
\[ d p^{L}_{\text{Customer}} \leq d p_{\text{Customer}}(t) \ \forall t \in [t_0, t_f], \]
### 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 optimization
Challenges for the community

- More Solver for dynamic optimization

- **Mixed-integer-optimal control problems**
Challenges for the community

- More Solver for dynamic optimization

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

- More domain specific libraries
Challenges for the community

- More Solver for dynamic optimization
- **Mixed-integer-optimal control problems**
- More domain specific libraries
- Scalability studies
Thank you for your Attention