Collocation Methods for Dynamic Optimization

Fredrik Magnusson

Department of Automatic Control
Faculty of Engineering
Lund University, Sweden

Outline

Outline

Background

Modelica is

- an equation-based language for modelling dynamical systems
- object-oriented
- declarative, rather than causal
- applicable in many different engineering domains
- used in many different tools

JModelica.org is one such tool, which originated at this department about five years ago.

JModelica.org

JModelica.org is developed by Modelon in collaboration with Lund University (Dept. of Automatic Control, Numerical Analysis and Computer Science).

At first, JModelica.org focused on dynamic optimization rather than simulation (slightly different situation today).

The main optimization algorithm in JModelica.org is implemented in C using CppAD - suffers from efficiency issues and an inflexible implementation.

My work

To remedy the issues of the old optimization algorithm, I started implementing a new algorithm. The new algorithm is

- based on the same mathematical theory (soon to be the topic of discussion)
- uses Python and CasADi instead of C and CppAD

This started out as a master's thesis at Modelon in 2011. Today I am the main developer of optimization-related algorithms in JModelica.org.

Outline

Dynamic optimization

In our course in Nonlinear Control, we study optimal control problems of the form

$$
\begin{aligned}
\text{minimize} \quad & \phi(t_f, x(t_f)) + \int_0^{t_f} L(x(t), u(t))\,\mathrm{d}t, \\
\text{with respect to} \quad & t_f, x, u, \\
\text{subject to} \quad & \dot{x} = f(x(t), u(t)), \\
& x(0) = x_0, \\
& g_e(t, u(t)) = 0, \\
& g_i(t, u(t)) \leq 0, \\
& \psi(x(t_f)) = 0, \\
& \forall t \in [0, t_f].
\end{aligned}
$$

Pontryagin's maximum principle

- Using Pontryagin's maximum principle, this results in a boundary value problem.
- Not feasible to solve analytically in many practical applications
- One possible approach is to solve the boundary value problem numerically. This is called an indirect approach, and was the state of the art until the 1970s.
- Two major weaknesses
  - the switching structure of the inequality constraints can be difficult to find
  - sensitive to initial guesses of adjoint states

Direct approach

Another approach is to discretize the problem before establishing optimality conditions.

- Results in a mathematical program
- Optimality conditions given by the Karush–Kuhn–Tucker conditions.
- This is called a direct approach, and is most commonly used today.
- Two major categories of direct approaches: collocation (today's topic) and multiple shooting

System dynamics

System dynamics modeled by a differential algebraic equation (DAE) system of the form

$$F(t, \dot{x}(t), x(t), u(t), w(t), p) = 0.$$

The introduction of free parameters p enables formulation of parameter identification problems.

Let $z := (\dot{x}, x, u, w)$, giving us

$$F(t, z(t), p) = 0.$$

DAE system is assumed to be of most index one (JModelica.org compiler transforms it to this form).

Generalized constraints

Generalized constraints

- Initial conditions: $F_0(z(t_0), p) = 0$
- Path constraints: $g_e(t, z(t), p) = 0$
  $g_i(t, z(t), p) \leq 0$
- Point constraints: $\psi_e(z(T_1), \ldots, z(T_m), p) = 0$
  $\psi_i(z(T_1), \ldots, z(T_m), p) \leq 0$

Generalized dynamic optimization

The result is the dynamic optimization problem (DOP), which covers both parameter identification and optimal control

$$
\begin{aligned}
&\text{minimize} && \phi(t_0, z(t_0), t_f, z(t_f), p) + \int_{t_0}^{t_f} L(t, z(t), p)\,\mathrm{d}t, \\
&\text{with respect to} && t_0, t_f, z, p, \\
&\text{subject to} && F(t, z(t), p) = 0, \\
& && F_0(z(t_0), p) = 0, \\
& && g_e(t, z(t), p) = 0, \\
& && g_i(t, z(t), p) \leq 0, \\
& && \psi_e(z(T_1), \ldots, z(T_m), p) = 0, \\
& && \psi_i(z(T_1), \ldots, z(T_m), p) \leq 0, \\
& && \forall t \in [t_0, t_f].
\end{aligned}
$$

Reminder: $z := (\dot{x}, x, u, w)$

No assumptions about linearity or convexity

Outline

Collocation methods

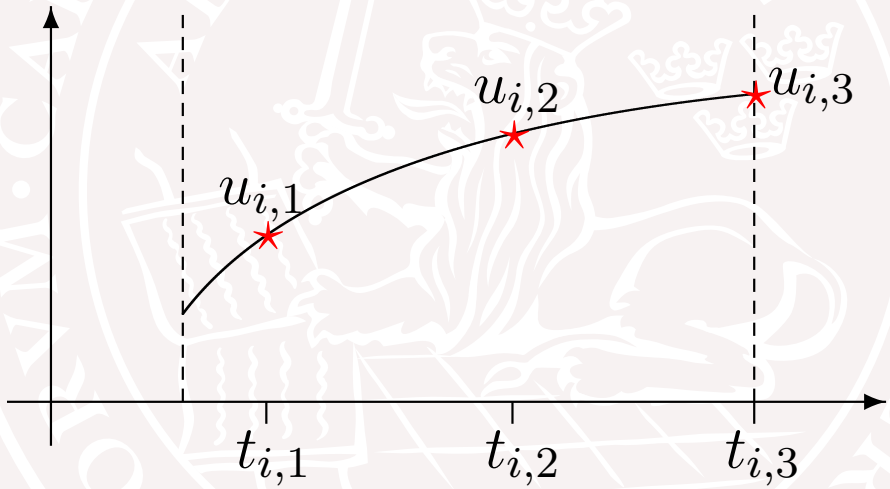
Main idea is to approximate system trajectories by polynomials:

- Discretize the time horizon $[t_0, t_f]$ into n_e elements
- Approximate the time-variant variables in each element by a polynomial
- Force this polynomial to satisfy all the constraints in n_c points
- This uniquely determines a polynomial of degree $n_c - 1$ by interpolation

Collocation polynomials

Let $t_{i,k}$ denote collocation point number k in element i and let

$$u_{i,k} := u(t_{i,k}).$$



Polynomials for x and w are constructed in a similar manner.

The choice of collocation points $t_{i,k}$ define the collocation method.

Cost function approximation

- Use collocation point values to approximate cost function

$$\phi(t_0, z(t_0), t_f, z(t_f), p) + \int_{t_0}^{t_f} L(t, z(t), p)\,\mathrm{d}t$$

- Mayer term is straightforward to approximate:

$$\phi(t_0, z(t_0), t_f, z(t_f), p) \approx \phi(t_0, z_{1,0}, t_f, z_{n_e,n_c}, p).$$

- Approximation of Lagrange term is done using quadrature:

$$\int_{t_0}^{t_f} L(t, z(t), p)\,\mathrm{d}t \approx \sum_{i=1}^{n_e} \left(h_i \cdot (t_f - t_0) \cdot \sum_{k=1}^{n_c} \omega_k \cdot L\left(t_{i,k}, z_{i,k}, p\right) \right)$$

DOP transcription

Let $Z := \{z_{i,k} \in \mathbb{R}^{n_z} : (1,1) \preceq (i,k) \preceq (n_e, n_c)\}$.

DOP

$$
\begin{aligned}
\text{min.} \quad & f(t_0, t_f, z, p), \\
\text{w.r.t.} \quad & t_0, t_f, z, p, \\
\text{s.t.} \quad & F(t, z(t), p) = 0, \\
& F_0(z(t_0), p) = 0, \\
& g_e(t, z(t), p) = 0, \\
& g_i(t, z(t), p) \leq 0, \\
& \psi_e(z(T_1), \ldots, z(T_m), p) = 0, \\
& \psi_i(z(T_1), \ldots, z(T_m), p) \leq 0, \\
& \forall t \in [t_0, t_f].
\end{aligned}
$$

$\longrightarrow$

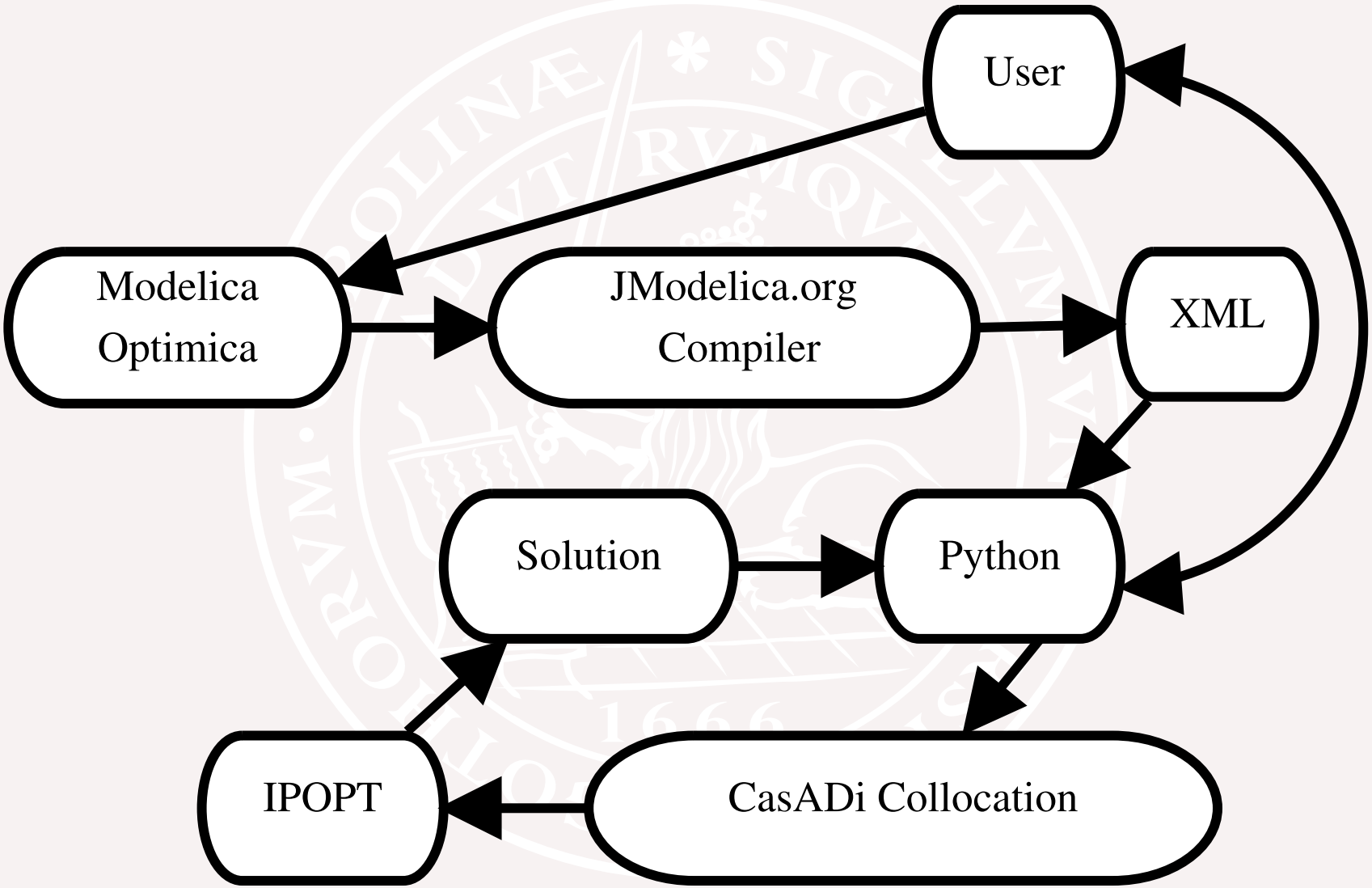
Nonlinear program (NLP)

$$
\begin{aligned}
\text{min.} \quad & \tilde{f}(t_0, t_f, Z, p), \\
\text{w.r.t.} \quad & t_0, t_f, Z, p, \\
\text{s.t.} \quad & F(t_{i,k}, z_{i,k}, p) = 0, \\
& F_0(z_{1,0}, p) = 0, \\
& g_e(t_{i,k}, z_{i,k}, p) = 0, \\
& g_i(t_{i,k}, z_{i,k}, p) \leq 0, \\
& \psi_e(Z, p) = 0, \\
& \psi_i(Z, p) \leq 0, \\
& \dot{x}_{i,k} = \frac{\mathrm{d}x_i}{\mathrm{d}t}(t_{i,k}) = \frac{1}{h_i \cdot (t_f - t_0)} \sum_{l=0}^{n_c} \alpha_{k,l} \cdot x_{i,l}, \\
& \forall (i,k) \in \mathbb{Z}^2 : (1,1) \preceq (i,k) \preceq (n_e, n_c).
\end{aligned}
$$

Solving the NLP

- Find approximate solution to the original DOP by solving the approximative NLP.
- JModelica.org uses the tool IPOPT (**I**nterior **P**oint **OPT**imizer) to solve NLPs.
- IPOPT needs first- and possibly second-order derivative information. The new collocation algorithm uses the tool CasADi (**C**omputer **a**lgebra **s**ystem with **A**utomatic **Di**fferentaion) to derivatives while preserving sparsity.

Algorithm workflow



New vs. old

Pros:

- Applies automatic differentiation (AD) to the entire NLP, resulting in faster iterations
- Supports analytic second-order derivatives (by AD), instead of BFGS, resulting in
  - even faster iterations
  - good initial guess $\Rightarrow$ faster and more robust convergence
  - bad initial guess $\Rightarrow$ possibly slower and less robust convergence
- More flexible implementation

Con:

- Only supports basic Modelica code (no functions and no records)

Application-oriented collaborations

- Optimal start-up of combined cycle power plants, with Francesco Casella from Politecnico di Milano
- Optimal vehicle maneuvers, with Björn & Karl
- Parameter identification of models for energy consumption in buildings, with Roel De Coninck from KU Leuven
- Parameter identification of atomic layer deposition reactor models (used in photovoltaics), with Anders Holmqvist from the Department of Chemical Engineering
- Future industrial collaboration: optimal start-up of combined cycle power plants, with Siemens AG

Future work

- Manipulate the index-one DAE system using graph algorithms before sending it to the collocation algorithm. In particular
  - solve equations analytically when possible
  - permute DAE system to block lower triangular form, to further exploit sparsity
  - apply tearing to reduce the number of iteration variables
- These techniques are commonly used by Modelica tools when simulating large-scale systems
- Can these techniques also be successfully applied to large-scale dynamic optimization?

The end

Thank you for listening!

The End