Causalization for Dynamic Optimization

Fredrik Magnusson

Department of Automatic Control
Faculty of Engineering
Lund University, Sweden

May 8, 2015

Outline

1. Dynamic Optimization
2. Causalization
3. Causalization for Dynamic Optimization

Outline

1. Dynamic Optimization
2. Causalization
3. Causalization for Dynamic Optimization

Me

- Started February 2012
- Working on open-source framework for large-scale dynamic optimization
- Looking for ways to make dynamic optimization algorithms more
  - efficient
  - reliable
  - accessible

Dynamic optimization

- Optimization problems with differential equations as constraints
- Applications include
  - optimal control (open or closed loop)
  - parameter estimation or optimization
  - state estimation (moving horizon estimation)
  - experiment design

JModelica.org

MODELICA

Modelon

Application-oriented collaborations

- Anders Holmqvist, Postdoc at Chemical Engineering, Lund
  - Previously identification and control of atomic layer deposition reactors (for e.g. construction of semiconductors and solar cells)
  - Currently optimal control of chromatographic processes (separation of chemical compounds, e.g. for pharmaceutics)
- Roel De Coninck, PhD Student at KU Leuven and 3E, Belgium
  - Identification, state estimation, and control for heating in buildings
- Karl Berntorp/Björn Olofsson, MERL/Control, Boston/Lund
  - Vehicle maneuvers
- Kilian Link et. al, Siemens AG, Erlangen, Germany
  - MPC for power plants
- Too many more...

Optimal control

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

Nonlinear dynamics and state constraints $\implies$ probably need numerical methods

Optimal control with DAE

Differential-algebraic equation (DAE) instead of explicit ODE:

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

Why DAE?

DAEs are a more useful framework than ODEs when either

- System variables are coupled by nontrivial algebraic (static) relations, common in e.g. multibody mechanics, electrical circuits, and chemical processes
- Component-based modeling; component connections usually give rise to algebraic equations

Collocation methods

One of the most common numerical methods for dynamic optimization is direct collocation. Main idea is to approximate system trajectories by polynomials:

- Divide the time horizon $[t_0, t_f]$ into a finite number of elements
- Approximate the time-variant variables in each element by a polynomial
- Force this polynomial to satisfy all the constraints in a few (or many) points

Nonlinear program

- The result is a nonlinear program (NLP) on the form

$$
\begin{aligned}
&\text{minimize} && f(x), \\
&\text{with respect to} && x \in \mathbb{R}^n, \\
&\text{subject to} && x_L \leq x \leq x_U, \\
& && g(x) = 0, \\
& && h(x) \leq 0.
\end{aligned}
$$

- NLP solution approximates the solution to the original dynamic optimization problem
- Nonlinear dynamics $\implies$ nonlinear equality constraints $g(x) = 0$ $\implies$ nonconvex problem $\implies$ nonglobal optimization

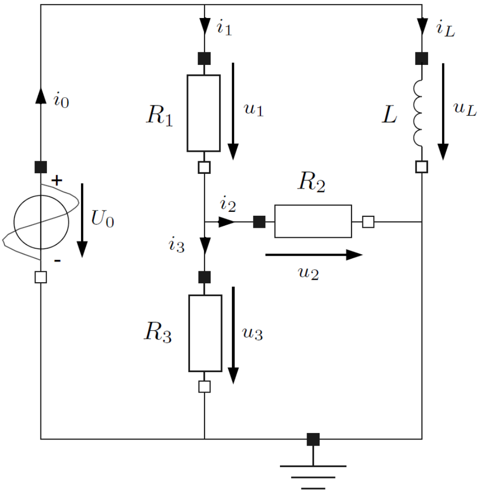
Outline

1. Dynamic Optimization
2. Causalization
3. Causalization for Dynamic Optimization

DAE simulation

- DAE systems can be simulated (numerically integrated) with specialized DAE solvers
- More common to transform the DAE to an ODE and apply ODE solvers
- This transformation consists of many steps, most notably
  - index reduction, reducing the DAE to index 1 (DAE has a unique solution for $\dot{x}$ and y)
  - causalization, eliminating or “hiding” all the algebraic variables

Example



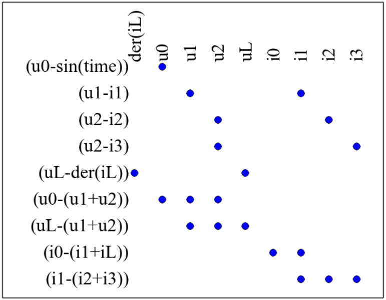
$$U_0 = \sin(t)$$
$$u_1 = R_1 \cdot i_1$$
$$u_2 = R_2 \cdot i_2$$
$$u_3 = R_3 \cdot i_3$$
$$u_L = L \cdot \frac{\mathrm{d}}{\mathrm{d}t} i_L$$
$$U_0 = u_1 + u_3$$
$$u_L = u_1 + u_2$$
$$u_3 = u_2$$
$$i_0 = i_1 + i_L$$
$$i_1 = i_2 + i_3$$

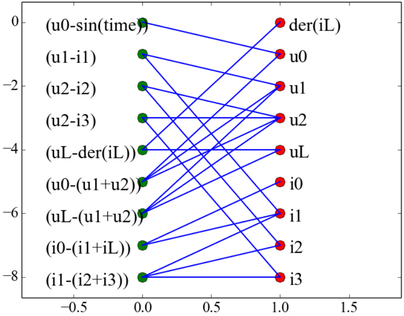
Matching

- To transform $F(\dot{x}, x, y) = 0$ into $\dot{x} = f(x)$, we want to solve the equations for $z = (\dot{x}, y)$
- We first match each variable to its own equation (possible iff DAE is index 1)

Incidence matrix and graph

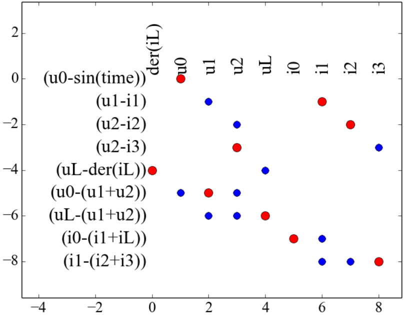
Incidences of $\dot{x}$ and y in DAE residuals

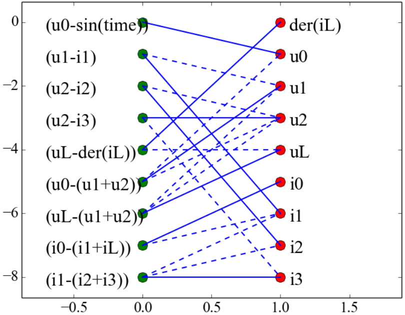




Incidence matrix and graph after matching

Matching found by applying Hopcroft-Karp on graph

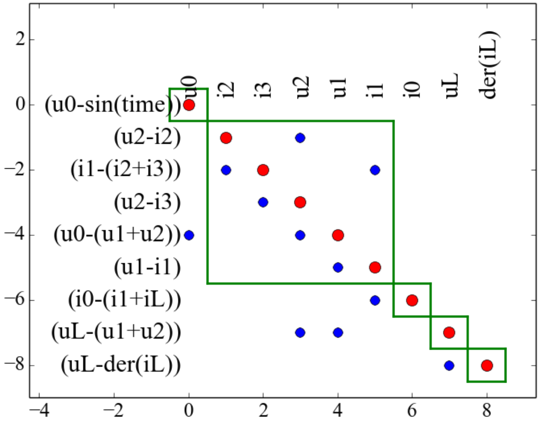




BLT

- Now we know which equation to solve for which variable
- Next step is to actually solve
- Rather than solve all equations simultaneously (like a DAE solver), permute the system to block-lower triangular (BLT) form
- Sequential solution of many, but small, equation systems
- Permutations found by applying Tarjan's algorithm on a similar graph to find strongly connected components

BLT incidence



Solving blocks

- If block equations depend nonlinearly unknowns ($\dot{x}$ and y), then Newton's method
- Then closed-form expression for ODE does not exist, but computationally behaves like an ODE (computes $\dot{x}$ given x)
- If block contains more than one equation/variable, they need to be solved simultaneously: Algebraic loop (Newton's method if nonlinear, numerical factorization if linear)
- Nonlinear and nonscalar blocks are surprisingly uncommon, even for large, nonlinear models!

The result

$$\begin{aligned} U_0 &\leftarrow \sin(t), \\ i_2 &\leftarrow U_0/3, \\ i_3 &\leftarrow U_0/3, \\ u_2 &\leftarrow U_0/3, \\ u_1 &\leftarrow 2U_0/3, \\ i_1 &\leftarrow 2U_0/3, \\ i_0 &\leftarrow i_1 + i_L, \\ u_L &\leftarrow u_1 + u_2, \\ \frac{\mathrm{d}}{\mathrm{d}t} i_L &\leftarrow u_L. \end{aligned}$$

Or simply: $\frac{\mathrm{d}}{\mathrm{d}t} i_L = \sin(t)$

Outline

1. Dynamic Optimization
2. Causalization
3. Causalization for Dynamic Optimization

Causalization for dynamic optimization

- Dynamic optimization conventionally done by exposing the full DAE to the numerical algorithm
- Let us causalize the DAE before applying e.g. direct collocation!

Initial considerations

- Goal is to eliminate algebraic variables to reduce number of variables; not to get an ODE
- Instead of nested Newton iterations for nonlinear blocks, keep them and expose to collocation and NLP solver
- May also want to not eliminate variables occurring in objective or path constraints

First results

Problem		n_x	n_y	Sol [s]
Vehicle	DAE	13	27	3.7
	Causal	13	4	2.0
CCPP	DAE	10	123	2.3
	Causal	10	1	∞
Dist.	DAE	125	1000	12
	Causal	125	2	94

What went wrong?

CCPP:

- One BLT blocks is linear, size 2, 2 state derivatives
- First state is a pressure, order of magnitude 10^8
- Second state is a volume fraction, order of magnitude 10^{-6}
- Block coefficient matrix numerically singular
- Proper solution: Scaling
- Temporary solution: Do not solve this block; instead leave it for the collocation and NLP solver

What went wrong?

Distillation column:

- Solving a block and using the solution in succeeding equations changes sparsity structure
- The full system typically becomes much smaller but denser
- No problem for typical simulation purposes; sparsity not exploited
- For optimization, trade-off between sparsity and size
- Proper solution: ?
- Envisioned solution: Analyze equation sparsity and only eliminate variables which do not majorly impact it

Results

Problem		n_x	n_y	Sol	KKT nnz	KKT nnz/row
Vehicle	DAE	13	27	3.7	9.4e4	4.7
	Causal	13	4	2.0	7.8e4	5.9
CCPP	DAE	10	123	2.3	1.7e5	3.6
	Causal	10	1	0.9	4.6e4	6.0
Dist.	DAE	125	1000	12	7.7e5	4.9
	Causal	125	2	94	1.3e6	35.9

Not just speed

Other benefits:

- Memory
- More consistent convergence
- More robust convergence (empirically)

Conclusion

Causalization for Dynamic Optimization:

- Faster and more robust convergence (if done right)
- Need to work on conditioning already during BLT stages
- Probably need to work sparsity preservation
- Future work: Nested Newton for nonlinear blocks, yay or nay?

The end

Thank you for listening!