Symbolic Transformations of Dynamic Optimization Problems

Fredrik Magnusson[a, *] Karl Berntorp[a] Björn Olofsson[a] Johan Åkesson[a,b]
[a]Department of Automatic Control, Lund University, SE-221 00 Lund, Sweden
[b]Modelon AB, Ideon Science Park, SE-223 70 Lund, Sweden

Abstract

Dynamic optimization problems involving differential-algebraic equation (DAE) systems are traditionally solved while retaining the semi-explicit or implicit form of the DAE. We instead consider symbolically transforming the DAE into an ordinary differential equation (ODE) before solving the optimization problem using a collocation method. We present a method for achieving this, which handles DAE-constrained optimization problems. The method is based on techniques commonly used in Modelica tools for simulation of DAE systems.

The method is evaluated on two industrially relevant benchmark problems. The first is about vehicle-trajectory generation and the second involves startup of power plants. The problems are solved using both the DAE formulation and the ODE formulation and the performance of the two approaches is compared. The ODE formulation is shown to have roughly three times shorter execution time. We also discuss benefits and drawbacks of the two approaches.

Keywords: dynamic optimization, symbolic transformations, causalization, collocation

1 Introduction

Industrial usage of optimization of large-scale dynamic systems has increased during the last decades. Dynamic optimization problems occur in many different fields and applications, and include parameter estimation and optimal control. Applications of optimal control include minimization of material and energy consumption during set-point transitions in power plants [1] and chemical processes [2], minimizing duration of vehicle maneuvers [3], and trajectory optimization in robotics [4].

The applications of optimal control are diverse and occur in both online and offline settings. Online optimal control is usually done in the form of Model Predictive Control (MPC). Offline applications include finding optimal trajectories, which can be used either as a reference during manual control or as nominal trajectories combined with online feedback handling deviations due to model uncertainty and disturbances. Another offline application is the identification of system bottlenecks, for example by analyzing adjoint variables.

This paper considers models described by differential-algebraic equation (DAE) systems, and investigates the benefits of applying symbolic transformations to the DAE before applying numerical optimization methods. The DAE is transformed into an ordinary differential equation (ODE). This will often lead to a drastically reduced number of system variables, as the algebraic variables are eliminated from the equation system. On the other hand, the transformed equations will also be denser and consist of more expressions of higher complexity. This transformation is common practice in simulation of DAEs in Modelica tools [5], but is traditionally not done in the context of DAE-constrained optimization, where the DAE is instead usually retained in its natural semi-explicit or implicit form.

The technique is evaluated in two case studies. The first case concerns generation of time-optimal trajectories for road vehicles. The second case concerns optimal startup of combined-cycle power plants.

The main contributions of this paper are the demonstration of how a method commonly used in Modelica tools can be applied to dynamic optimization problems and experiments indicating the potential of the method. While similar methods have been used before [6, 7], this paper studies the properties of the approach when compared to the more traditional approach that discretizes the full DAE.

The paper outline is as follows. Section 2 presents the background of transforming low-index DAEs into ODEs by causalization, the formulation of dynamic

*Corresponding author: fredrik.magnusson@control.lth.se
The authors are members of the LCCC Linnaeus Center and the ELLIIT Excellence Center at Lund University.

optimization problems and their solution, and the tools used in the implemented framework. Section 3 discusses how the DAE causalization technique is used to transform DAE-constrained optimization problems into ODE-constrained problems. Section 4 explains the case studies used to evaluate the method and Section 5 presents the corresponding results. Finally, Section 6 concludes the paper and discusses future work.

2 Background

We present the standard approach of transforming a DAE into an ODE by causalization. We then discuss the formulation of general optimization problems involving dynamic systems and commons methods for solving these. We finally present the tools used to implement the methods presented in this paper.

2.1 Causalization of DAEs

In the first step of the compilation process in a Modelica tool chain, a compiler front-end transforms Modelica source code into a flat representation, consisting essentially of lists of variables, functions, equations, and algorithms. Based on this model representation, symbolic operations such as alias elimination and index reduction are applied to reduce the size of the model and to ensure that the resulting DAE is of most index 1. In this section, we outline the steps needed to transform an implict DAE into an ODE, which is one of the key elements of the method investigated in this paper. We introduce the following notation:

- t time
- t_f final time
- x state (differentiated variables)
- $\dot{x}$ time derivative of state
- y algebraic variables
- u inputs
- p parameters without predetermined values

The initial time is assumed to be 0 and the final time t_f may or may not be predetermined, but is always finite. The variables $x, \dot{x}, y$, and u depend on time. This dependence will be implicit in certain expressions throughout the paper.

We consider nonlinear nonhybrid index-1 DAE systems of the form

$$F(t,\dot{x}(t),x(t),y(t),u(t),p) = 0, \quad t \in [0,t_f]. \tag{1}$$

From an integrator perspective, we introduce

$$z := (\dot{x}, y), \quad v := (t,x,u,p)$$

to denote the unknown and known variables of the equation system, respectively. By reordering the arguments of F, the DAE can be written

$$F(z,v) = 0. \tag{2}$$

The conceptual idea of DAE causalization commonly used in Modelica tools is to compute the inverse relationship of F:

$$z = g(v). \tag{3}$$

The DAE can then be written as the ODE

$$\dot{x} = \overline{F}(t,x,u,p), \tag{4}$$

where the algebraic variables are internal in the right-hand side function. In general, there is no closed expression for the function $\bar{F}$. Rather, iterative techniques, such as Newton's method, are employed to solve algebraic loops required for computation of z.

Modelica models are typically of large scale but sparse in the sense that each model equation contains references only to a small number of equations. Graph algorithms can be employed to exploit this structure. Two commonly used algorithms that are used for this purpose are *matching algorithms*, such as the Hopcroft Karp algorithm, and Tarjan's algorithm [8] for computing *strong components*. The result of Tarjan's algorithm is used to permute the variables and equations of the DAE into Block Lower Triangular (BLT) form.

To demonstrate this procedure, let us consider an exemplary DAE system with five equations and five unknowns, where the DAE system is given by

$$\begin{aligned} F_1(z_1,z_5,v) &= 0, \\ F_2(z_3,v) &= 0, \\ F_3(z_1,z_2,z_3,z_4,v) &= 0, \\ F_4(z_1,z_3,z_5,v) &= 0, \\ F_5(z_2,z_5,v) &= 0. \end{aligned} \tag{5}$$

Note that the variable $v = (t,x,u,p)$ is known and needs not be considered in the following analysis. The dependence on the z variables can be shown in the following incidence matrix:

	z_1	z_2	z_3	z_4	z_5
F_1	$*$	0	0	0	$*$
F_2	0	0	$*$	0	0
F_3	$*$	$*$	$*$	$*$	0
F_4	$*$	0	$*$	0	$*$
F_5	0	$*$	0	0	$*$

(6)

DOI 10.3384/ECP140961027

An asterisk in (6) in row i and column j denotes that the residual function F_i contains a reference to the variable z_j. Application of the BLT procedure yields the following permuted incidence matrix:

	z_3	z_1	z_5	z_2	z_4
F_2	$*$	0	0	0	0
F_4	$*$	$*$	$*$	0	0
F_1	0	$*$	$*$	0	0
F_5	0	0	$*$	$*$	0
F_3	$*$	$*$	0	$*$	$*$

(7)

By reordering and grouping the variables according to

$$\begin{aligned}\bar{z}_1 &:= z_3,\\ \bar{z}_2 &:= (z_1, z_5),\\ \bar{z}_3 &:= z_2,\\ \bar{z}_4 &:= z_4,\end{aligned}$$

the DAE (5) can be written

$$G_1(\bar{z}_1, v) = 0, \tag{8a}$$
$$G_2(\bar{z}_1, \bar{z}_2, v) = 0, \tag{8b}$$
$$G_3(\bar{z}_2, \bar{z}_3, v) = 0, \tag{8c}$$
$$G_4(\bar{z}_1, \bar{z}_2, \bar{z}_3, \bar{z}_4, v) = 0, \tag{8d}$$

where the functions G_i are constructed from the functions F_j. For demonstrative purposes, we assume that (8a) and (8d) can be solved explicitly for $\bar{z}_1$ and $\bar{z}_2$ respectively, and that (8b) and (8c) can not be solved analogously. The DAE can then be represented by the following sequence of assignment statements and implicit equation systems:

$$\bar{z}_1 \leftarrow g_1(v), \tag{9a}$$
$$G_2(\bar{z}_1, \bar{z}_2, v) = 0, \tag{9b}$$
$$G_3(\bar{z}_2, \bar{z}_3, v) = 0, \tag{9c}$$
$$\bar{z}_4 \leftarrow g_4(\bar{z}_1, \bar{z}_2, \bar{z}_3, v). \tag{9d}$$

where (9b) and (9c) require iterative methods to be solved. It is typical for Modelica models to contain only a small number of implicit equation systems that require iteration and a large number of trivial, for example linear, equations that can be solved symbolically.

For a general DAE, the BLT procedure results in a sequence of equation systems of the form

$$\begin{aligned}\overline{G}_1(\bar{z}_1, v) &= 0,\\ &\vdots\\ \overline{G}_i(\bar{z}_1, ..., \bar{z}_i, v) &= 0,\\ &\vdots\\ \overline{G}_b(\bar{z}_1, ..., \bar{z}_b, v) &= 0,\end{aligned} \tag{10}$$

where b is the number of blocks in the BLT form and the unknown of each equation is $\bar{z}_i$. Some equations can be solved explicitly by symbolic manipulation, while the rest needs to be solved iteratively.

Given values of the known variables in v, the sequence of solved and unsolved blocks (10) allows for the computation of the corresponding state derivative and algebraic vectors contained in z. Accordingly, the DAE has been causalized into an ODE of the form (4).

We consider the class of DAEs that can be transformed into an ODE where no implicit systems of equations need to be solved. This lets us redefine each unknown $\bar{z}_i$ to be a single scalar variable and compute it explicitly as

$$\bar{z}_i = g_i(\bar{z}_1, ..., \bar{z}_{i-1}, v), \quad i = 1, \ldots, n_z, \tag{11}$$

where n_z is the total number of states and algebraic variables. While this class of systems is limited, the proposed method is trivially extendible to systems containing implicit systems by simply exposing the implicit systems to the numerical optimization method used in the end, in which case some algebraic equations will remain in the transformed DAE.

2.2 Dynamic optimization

Consider the DAE-constrained optimization problem:

$$\begin{aligned}
\text{minimize } & \phi(t_f, \dot{x}(t_f), x(t_f), y(t_f), u(t_f), p) \\
& + \int_0^{t_f} L(t, \dot{x}(t), x(t), y(t), u(t), p)\, \mathrm{d}t, && \text{(12a)}\\
\text{w.r.t. } & t_f, \dot{x}, x, y, u, p, \\
\text{subject to } & F(t, \dot{x}, x, y, u, p) = 0, && \text{(12b)}\\
& F_0(\dot{x}(0), x(0), y(0), p) = 0, && \text{(12c)}\\
& \dot{x}_L \le \dot{x}(t) \le \dot{x}_U, && \text{(12d)}\\
& x_L \le x(t) \le x_U, && \text{(12e)}\\
& y_L \le y(t) \le y_U, && \text{(12f)}\\
& u_L \le u(t) \le u_U, && \text{(12g)}\\
& p_L \le p \le p_U, && \text{(12h)}\\
& h_e(t, \dot{x}, x, y, u, p) = 0, && \text{(12i)}\\
& h_i(t, \dot{x}, x, y, u, p) \le 0, && \text{(12j)}\\
& H_e(\dot{x}(t_f), x(t_f), y(t_f), u(t_f), p) = 0, && \text{(12k)}\\
& H_i(\dot{x}(t_f), x(t_f), y(t_f), u(t_f), p) \le 0, && \text{(12l)}\\
& \forall t \in [0, t_f].
\end{aligned}$$

The objective (12a) consists of the terminal cost ϕ and the integral of L, which is the accumulated cost. Constraint (12b) is the DAE describing the system dynam-

ics. Constraint (12c) enforces the DAE initial conditions, which are often given on the form $x(0) = x_0$. Constraints (12d)–(12h) are variable bounds. Constraints (12i) and (12j) are path constraints on equality and inequality form, which can be seen as generalizations of the variable bounds, where the functions h_e and h_i define the boundary. The variable bounds are separated from the path constraints because some solvers allow for more efficient treatment of the bounds. Constraints (12k) and (12l) are terminal constraints on equality and inequality form. These are similar to the path constraints, but instead of being enforced at all points in time they are only enforced at t_f.

There are many approaches to solving dynamic optimization problems of the form (12). Until the 1970s, problems were solved using dynamic programming or Pontryagin's maximum principle. These approaches are ill-suited for large-scale problems and problems with inequality constraints. Modern techniques often involve finding an approximate solution to the infinite-dimensional optimization problem by transcribing it into a finite-dimensional nonlinear program (NLP). These are called direct methods. The main difference among direct methods is how to handle the dynamic equations of the system. This paper employs direct local collocation. Another common approach is direct multiple shooting. See [9, 10] for overviews on direct local collocation and other direct methods.

The main idea of direct local collocation is to first divide the time horizon into a certain number of elements. Then within each element, the constraints (12b) and (12d)–(12j) are enforced at only a finite number of points, called collocation points, instead of at every point in the element. There are different schemes for choosing the placement of the collocation points with different numerical properties. The results in Section 5 have been generated using Radau points.

The constraints resulting from the collocation procedure are considered as interpolation conditions on the time-dependent variables x, y, and u. Thus the sought approximate optimal trajectories to (12) become piecewise-polynomial, where the degrees of the polynomials are determined by the number of collocation points. The state derivative $\dot{x}$ is obtained by differentiating the corresponding polynomials for the state x. The integral term in the objective (12a) is approximated as a sum using quadrature. See [11] for a complete description of the used collocation method, and also possible generalizations of (12). Once the discretization procedure is completed, the infinite-dimensional dynamic optimization problem (12) has been transformed into an NLP of the following general form:

$$\begin{aligned}
&\text{minimize} && \tilde{f}(\tilde{x}), && \text{(13a)}\\
&\text{with respect to} && \tilde{x} \in \mathbb{R}^{n_{\tilde{x}}},\\
&\text{subject to} && \tilde{x}_L \le \tilde{x} \le \tilde{x}_U, && \text{(13b)}\\
& && \tilde{g}(\tilde{x}) = 0, && \text{(13c)}\\
& && \tilde{h}(\tilde{x}) \le 0. && \text{(13d)}
\end{aligned}$$

The number of discretization points is affected both by the number of elements and the number of collocation points within each element. An increase in either of these directly corresponds to an increase in the number of variables and constraints in the NLP (13).

In this work, (13) is solved using a gradient-based method. This requires the NLP functions f, g, and h to be twice continuously differentiable with respect to all of the NLP variables $\tilde{x}$. In particular, this requirement implies differentiability of the DAE-residual F, which excludes the possibility of solving optimization problems involving hybrid systems.

When using direct methods for dynamic optimization problems involving DAEs, the time discretization method is typically applied to the DAE in its natural semi-explicit or implicit form [9, 10]. In this paper, we instead consider causalizing the DAE as described in Section 2.1 and then eliminating the algebraic variables in the optimization problem as described in Section 3. The result is an ODE-constrained optimization problem of the following form:

$$\begin{aligned}
&\text{minimize} && \bar{\phi}(t_f, \dot{x}(t_f), x(t_f), u(t_f), p)\\
& && \quad + \int_0^{t_f} \bar{L}(t, \dot{x}(t), x(t), u(t), p)\,\mathrm{d}t, && \text{(14a)}\\
&\text{with respect to} && t_f, \dot{x}, x, u, p,\\
&\text{subject to} && \dot{x} = \overline{F}(t, x, u, p) = 0, && \text{(14b)}\\
& && \overline{F}_0(\dot{x}(0), x(0), p) = 0, && \text{(14c)}\\
& && \dot{x}_L \le \dot{x}(t) \le \dot{x}_U, && \text{(14d)}\\
& && x_L \le x(t) \le x_U, && \text{(14e)}\\
& && u_L \le u(t) \le u_U, && \text{(14f)}\\
& && p_L \le p \le p_U, && \text{(14g)}\\
& && \bar{h}_e(t, \dot{x}, x, u, p) = 0, && \text{(14h)}\\
& && \bar{h}_i(t, \dot{x}, x, u, p) \le 0, && \text{(14i)}\\
& && \overline{H}_e(\dot{x}(t_f), x(t_f), u(t_f), p) = 0, && \text{(14j)}\\
& && \overline{H}_i(\dot{x}(t_f), x(t_f), u(t_f), p) \le 0, && \text{(14k)}\\
& && \forall t \in [0, t_f].
\end{aligned}$$

The objective and constraint functions occurring in

DOI 10.3384/ECP140961027

problem (14) are defined analogously to those occurring in problem (12).

This paper investigates the possibilities of transforming problems on the form of (12) to the form of (14) and the impact this has on the solution of the NLPs resulting from collocation methods.

2.3 Tools

The proposed method has been implemented using the open-source platform JModelica.org [12]. JModelica.org is a tool targeting simulation and optimization of large-scale dynamic systems. The systems are described using Modelica, and the optimization is formulated with the Modelica extension Optimica [13].

The framework uses IPOPT [14] to solve the NLP (13). IPOPT uses a sparse primal-dual interior point method to find local optima to large-scale NLPs. To this end, it uses first- and second-order derivatives of the NLP functions f, g, and h in problem (13). The implemented framework uses CasADi [15] (**C**omputer **a**lgebra **s**ystem with **A**utomatic **Di**fferentaion) to obtain these derivatives, and also to perform the transformation from (12) to (14). CasADi is a low-level tool for efficiently computing derivatives using algorithmic differentiation (AD) while preserving sparsity, and is tailored for dynamic optimization.

3 Proposed method

This section presents the proposed method for transforming the DAE-constrained optimization problem (12) into an ODE-constrained optimization problem of the form (14). We then compare the properties of the untransformed problem to those of the transformed problem.

3.1 Problem transformation

The first step is the causalization of the DAE in (12b), as described in Section 2.1. Under the assumption that all equations in (10) can be solved explicitly by symbolic manipulations, this yields the system of equations (compare with (11))

$$\bar{z}_i = g_i(\bar{z}_1, \bar{z}_2, \ldots, \bar{z}_{i-1}, v), \quad i = 1, \ldots, n_z, \tag{15}$$

where $\bar{z}_i$ is a state derivative or an algebraic variable. The explicit solution for $\bar{z}_1, \bar{z}_2, \ldots, \bar{z}_{i-1}$ is then inlined into (15). The resulting equations are described by the following recursive relations:

$$\bar{g}_1(v) := g_1(v), \tag{16a}$$
$$\bar{g}_i(v) := g_i(\bar{g}_1(v), \bar{g}_2(v), \ldots, \bar{g}_{i-1}(v), v), \tag{16b}$$
$$\bar{z}_i = \bar{g}_i(v), \quad i = 1, \ldots, n_z. \tag{16c}$$

By expanding the variables v and z and separating the state derivatives from the algebraic variables in (16c), the equations can be written in the form

$$\dot{x} = \overline{F}(t, x, u, p), \tag{17a}$$
$$y = k(t, x, u, p), \tag{17b}$$

where each scalar component of $\overline{F}$ and k is equal to $\bar{g}_i$ for some i. Thus (17a) gives rise to the constraint in (14b). Equation (17b) is used to eliminate the algebraic variables in the objective function and remaining constraints of (12). To demonstrate, the function h_i in (12j) is transformed according to

$$\begin{aligned} &h_i(t, \dot{x}(t), x(t), y(t), u(t), p) \\ &\quad = h_i(t, \dot{x}(t), x(t), k(t, x(t), u(t), p), u(t), p) \\ &\quad =: \bar{h}_i(t, \dot{x}(t), x(t), u(t), p). \end{aligned}$$

Constraint (12f) is not possible to transform in this manner, since the algebraic variables have been eliminated. This is handled by transforming (12f) into its more general form (12j), which has consequences discussed in Section 3.2, and then transforming it as demonstrated above. By eliminating the algebraic variables in the objective and remaining constraints in the same manner, the optimization problem (12) has been transformed into the equivalent problem (14).

3.2 Method properties

A significant benefit of transforming the original DAE-constrained problem into an ODE-constrained problem is the reduction of system variables and equations, which directly leads to a smaller NLP after applying a collocation method. However, while the constraints will be fewer in number their expressions will be more complex, which may lead to expression graphs that in the end require more memory to store and more time to evaluate than the expression graphs in the original problem. On the other hand, component-based physical modeling will often lead to a great amount of trivial algebraic equations that can be solved explicitly without increasing expression complexity. It thus stands to reason that the proposed method is well suited to be used in a Modelica framework.

The elimination of algebraic variables will make the incidence matrix for the dynamic equation system

denser. However, the sparsity in the NLP does not mainly stem from the structure of the dynamic equations, but rather from the largely decoupled dependency of NLP variables representing system variable values in different elements. Hence in many cases, the loss of sparsity in the DAE will not have a significant impact on the solution of the NLP constructed by collocation methods.

When employing interior-point methods to solve NLPs originating from dynamic optimization problems, it is often beneficial to introduce artificial bounds on system variables to prevent the solver from leaving the domains of the involved functions. When the algebraic variables are eliminated from the problem, it is no longer possible to use these artificial bounds on algebraic variables. This can lead to difficulties in obtaining convergence in the numerical solver.

Experience has shown that seemingly trivial modifications in the original problem formulation can lead to drastically different convergence behavior in the NLP solver. By explicitly solving these trivial equations by performing the proposed transformation, increased robustness to small model modifications is attained.

The inlining step performed to obtain (16c) gives rise to a great amount of expression duplication in the expression graph for the right-hand side of (16b) because of the recursive nature of the equations. This can be avoided by storing the expressions used to evaluate $\bar{g}_i$ in a function and calling these functions when evaluating g_i. This paper does, however, not consider this proposition any further.

4 Benchmark problems

Two different optimal-control cases are presented for evaluation of the proposed transformation method: vehicle trajectory generation and startup of a combined-cycle power plant. The vehicle models are implemented in a flat manner, with condensed and complex equations, whereas the power plant model is implemented in an object-oriented fashion with a large number of simple equations.

4.1 Vehicle trajectory generation

The first considered case is a minimum-time problem, where we seek the time-optimal maneuver for an automobile in a hairpin turn, see Figure 1. A methodology for solving this kind of problems has previously been investigated in [3, 16, 17]. We use two different chassis models in the evaluation. The first is a single-track model [18], where the two wheels on each axle are lumped together. The model has three degrees of freedom: two translational and one rotational.

Figure 1: An example of a hairpin turn. Photo courtesy of RallySportLive.

The second model is a double-track model [19] with five degrees of freedom: two translational and three rotational. The suspension dynamics model is a rotational spring-damper system, and longitudinal and lateral load transfer is included.

The lateral slip α and the longitudinal slip λ are defined as in [20]:

$$\dot{\alpha}_i \frac{\sigma}{v_{x,i}} + \alpha_i := -\arctan\left(\frac{v_{y,i}}{v_{x,i}}\right), \tag{18}$$

$$\lambda_i := \frac{r\omega_i - v_{x,i}}{v_{x,i}}, \tag{19}$$

where σ is the relaxation length, r is the wheel radius, ω_i is the wheel angular velocity for wheel i, and $v_{y,i}$ and $v_{x,i}$ are the lateral and longitudinal wheel velocities for wheel i with respect to an inertial system, expressed in the coordinate system of the wheel.

The tire forces F_{x0} and F_{y0} for the longitudinal and lateral directions under pure slip conditions are computed with the Magic formula [20], given by

$$F_{x0,i} = \mu_x F_{z,i} \sin\Big(C_{x,i} \arctan\big(B_{x,i}\lambda_i - E_{x,i}(B_{x,i}\lambda_i - \arctan B_{x,i}\lambda_i)\big)\Big), \tag{20}$$

$$F_{y0,i} = \mu_y F_{z,i} \sin\Big(C_{y,i} \arctan\big(B_{y,i}\alpha_i - E_{y,i}(B_{y,i}\alpha_i - \arctan B_{y,i}\alpha_i)\big)\Big), \tag{21}$$

for each wheel $i = 1,\ldots,4$. In (20)–(21), μ_x and μ_y are friction coefficients and B, C, and E are parameters.

We have chosen two different approaches for modeling the tire forces under combined slip constraint, both of which are described next. A straightforward model of combined slip is based on the friction ellipse, and is described by the elliptical relation

$$F_{y,i} = F_{y0,i}\sqrt{1 - \left(\frac{F_{x0,i}}{\mu_{x,i}F_{z,i}}\right)^2}, \tag{22}$$

DOI 10.3384/ECP140961027

where F_{x0} is used as an input variable [21]. However, here the driving/braking torques are used as input. The main limitation with this model is that the longitudinal force does not explicitly depend on the lateral slip.

Another approach to tire modeling, which is inspired by the Magic Formula, is to scale the nominal forces (20)–(21) with weighting functions $G_{\alpha,i}$ and $G_{\lambda,i}$, which depend on α and λ [20]. The relations in the longitudinal direction are

$$H_{\alpha,i} = B_{1,i}\cos(\arctan(B_{2,i}\lambda_i)), \tag{23}$$
$$G_{\alpha,i} = \cos(C_{\alpha,i}\arctan(H_{\alpha,i}\alpha_i)), \tag{24}$$
$$F_{x,i} = F_{x0,i}G_{\alpha,i}, \tag{25}$$

and the corresponding relations in the lateral direction are given by

$$H_{\lambda,i} = B_{1,i}\cos(\arctan(B_{2,i}\alpha_i)), \tag{26}$$
$$G_{\lambda,i} = \cos(C_{\lambda,i}\arctan(H_{\lambda,i}\lambda_i)), \tag{27}$$
$$F_{y,i} = F_{y0,i}G_{\lambda,i}. \tag{28}$$

To summarize, four different model configurations were investigated for the vehicle trajectory generation, all having three control inputs. The combination of single-track chassis model and friction ellipse for tire modeling (ST-FE) has 13 states and 13 algebraic variables. The corresponding numbers for the combination of single-track chassis model and weighting functions (ST-WF) are 13 states and 23 algebraic variables. Considering the double-track model with friction ellipse model (DT-FE), it has 21 states and 36 algebraic variables. For the combination of double-track model and weighting functions (DT-WF), the corresponding numbers are 21 states and 56 algebraic variables.

For each model configuration, the time-optimal trajectories in the hairpin turn are to be determined. An initialization procedure based on a driver model presented in [18] is used. The optimization problem is formulated over the time horizon $t \in [0, t_f]$ and the objective of the optimization is to minimize the final time t_f of the maneuver. Accordingly, the dynamic optimization problem to be solved can be written as:

$$\begin{aligned}
\text{minimize} \quad & t_f, \\
\text{subject to} \quad & T_{i,\min} \le T_i \le T_{i,\max},\ i \in \{1,2,3,4\}, \\
& |\dot{T}_i| \le \dot{T}_{i,\max},\ i \in \{f,r\}, \\
& |\delta| \le \delta_{\max}, \quad |\dot{\delta}| \le \dot{\delta}_{\max} \\
& x(0) = x_0, \\
& x(t_f) = x_{t_f}, \quad y(t_f) = y_{t_f}, \\
& \Gamma(X_p, Y_p) \le 0, \\
& F(\dot{x}, x, y, u) = 0,
\end{aligned} \tag{29}$$

where x_0 are the initial conditions for the state variables, x_{t_f} and y_{t_f} are the desired values at the final time $t = t_f$, and (X_p, Y_p) is the position of the center-of-gravity of the vehicle. The wheel driving and braking torques $T = \begin{pmatrix} T_f & T_r \end{pmatrix}$ of the front and rear wheel axles, as well as the steer angle δ of the front wheels are considered as inputs. The inputs are equally distributed between the wheels at the respective axle, that is, $T_1 = T_2 = T_f/2$ and $T_3 = T_4 = T_r/2$ for the double-track chassis model. In practice, the terminal constraints are only applied to a subset of the model variables. Further, $\Gamma(X_p, Y_p)$ is a mathematical description of the road constraint for the center-of-gravity of the vehicle for the maneuver. These constraints in the geometric two-dimensional XY-plane are formulated as super-ellipses.

Figure 2 shows the geometric path and the time-optimal control inputs obtained for ST-WF. For more details about the solution method and the model parameters used, see [3, 17].

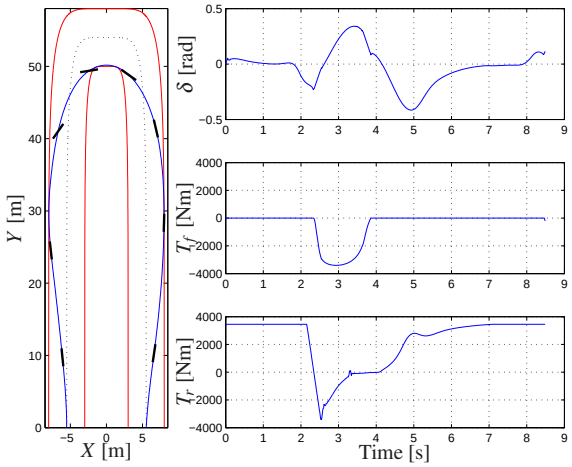

Figure 2: The geometric path and control inputs are shown for the time-optimal hairpin maneuver when using a single-track model with the weighting functions. The black bars in the left plot indicate the direction of the car every second.

4.2 Combined-cycle power plant startup

The second considered case concerns optimal startup of combined-cycle power plants (CCPP). The model used is described in [1]. The model has 9 states, 128 algebraic variables, and 1 control variable. The task is to minimize the time required to perform a warm startup of the power plant. This problem has become highly industrially relevant during the last years, due to an increasing need to improve power-generation flexibility. The startup process is considered finished when

the normalized load input signal u to the steam turbine, starting at 15 %, has reached 100 % and the evaporator pressure p, which is a state with an initial value of approximately 3.47 MPa, has reached 8.35 MPa.

In order to reduce the wear and tear on the steam turbine, which is one of the most expensive parts of the power plant, the thermal stress in the turbine σ, which is an algebraic variable, may not exceed 260 MPa. This is the main limiting factor in the startup process. Another imposed constraint is that the derivative of the load input signal u may not be negative and may not exceed $0.1/60$ s^{-1}. Since these bounds are applied to the derivative of the control variable, which is not supported by the current framework, we introduce the control variable $\dot{u}$ and add the equation

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \dot{u}.$$

This converts the control variable u into a state, giving us a total of 10 states, and the control variable is now instead $\dot{u}$, on which we can impose the discussed bounds. The model diagram is displayed in Figure 3.

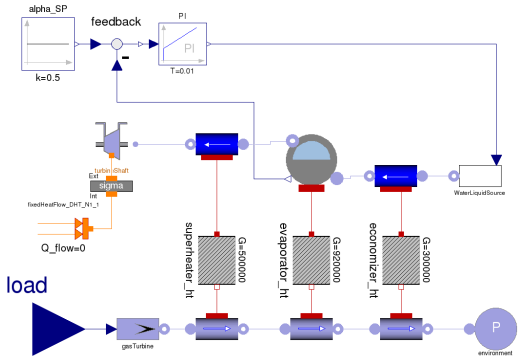

Figure 3: Power plant model diagram

The objective function is the weighted square deviation of the load input signal and the evaporator pressure from their respectively desired values, given by

$$f(z) = \int_0^{t_f} \left(10^{-12} \cdot \left(p(t) - 8.35 \cdot 10^6\right)^2 + 0.5 \cdot \left(u(t) - 1\right)^2 \right) \mathrm{d}t.$$

The final time is chosen to be $t_f = 4000$ s. The obtained solution is displayed in Figure 4.

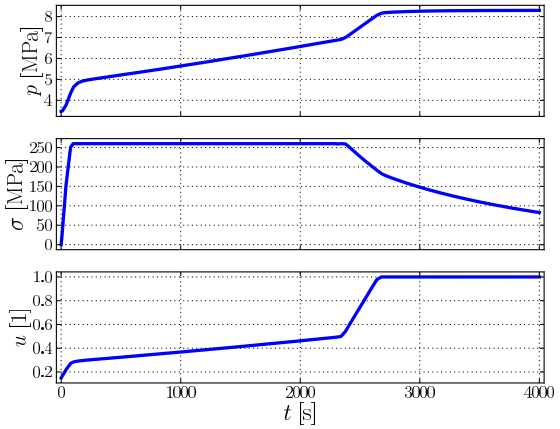

Figure 4: Optimal power plant startup

5 Results

The problems in the respective cases described in Section 4 were solved with the implementation described in Section 2.3. The solutions were obtained using JModelica.org revision [5625] and IPOPT version 3.11.3 with the linear solver MA57 [22]. The collocation discretization was done using 150 elements with 3 collocation points for each vehicle maneuver problem and 40 elements with 4 collocation points for the power plant startup problem. The solution procedure consists of the following three steps:

1. Model compilation, where the compiler of JModelica.org generates XML code that describes the system equations and optimization formulation. This code is then parsed by CasADi, which then creates symbolic representations of all the problem expressions.

2. Offline NLP setup, where the symbolic expressions created by CasADi in the previous step are used to construct the corresponding NLP by collocation. First- and second-order derivatives are computed by algorithmic differentiation while preserving sparsity.

3. Online NLP solution, where IPOPT solves the NLP constructed in the previous step. This is the only part of the solution procedure that would need to be performed in an online setting, such as MPC.

All problems were solved both with the model dynamics on DAE form and transformed to ODE form.

DOI 10.3384/ECP140961027

If the DAE is transformed into an ODE, the transformation takes place in the first step and is performed by CasADi. The respectively obtained solutions for the DAE and ODE formulations are the same up to tolerances.

For comparison of the two different strategies to solving the optimization problems, the time spent in step 2 and 3 of the solution procedure were measured separately for each of the optimization runs. The number of iterations required by IPOPT and the number of NLP variables have also been recorded. Table 1 displays the resulting numbers, where times are presented in seconds. The time spent in step 1 of the solution procedure is between 1 and 2 seconds for all of the problems and is largely unaffected by whether the ODE transformation is performed. These times are thus not presented.

Table 1: Solution times [s] for the considered model configurations with DAE and ODE form in the optimal control problem, respectively. In addition, the number of iterations required to solve the NLP and the total number of NLP variables are shown.

Problem		Offline	Online	Iter	NLP
ST-FE	DAE	3.6	10.6	112	20880
	ODE	3.5	5.0	83	15017
ST–WF	DAE	4.2	17.6	102	25390
	ODE	3.7	5.1	77	15017
DT–FE	DAE	9.0	152.2	303	39661
	ODE	10.5	46.0	151	23425
DT–WF	DAE	9.1	229.6	364	48681
	ODE	10.8	116.4	322	23425
CCPP	DAE	3.5	5.4	109	23574
	ODE	1.8	1.4	79	3771

There is no clear trend for the offline execution time in the four vehicle problems, whereas it is halved for the power plant. Both the number of iterations and the online execution times are shorter for the transformed problem in all compared scenarios. For the power plant, the difference in online execution time is approximately a factor of 4, whereas the vehicle examples exhibit a factor of between 2 and 3. This indicates that models containing a large amount of simple equations gain more from the proposed method, for reasons discussed in Section 3.2, but also that models with mainly complex algebraic equations gain speedups in the online NLP solution from the transformation.

6 Conclusions

We have presented a method for symbolically transforming a broad class of DAE-constrained optimization problem into an ODE-constrained optimization problem. The approach has been evaluated by measuring execution times for benchmark problems involving time-optimal trajectory generation for vehicles and startup of combined-cycle power plants.

The considered problems have been solved with both a traditional approach where the DAE is left intact in an implicit form, and with the presented method where the DAE system is symbolically transformed into an ODE before applying discretization techniques. Significant speedups have been observed in the solution of the NLPs resulting from a direct collocation discretization method.

While the method has exhibited great performance improvements, potential drawbacks have also been discussed. The most significant drawback is that while the dynamic equation system becomes smaller in size, the resulting expressions are often of much higher complexity.

Future work is the resolution of some of the discussed drawbacks of the approach, by only eliminating a suitable subset of the algebraic variables. A possible improvement in the opposite direction is to employ tearing techniques, which are often used for simulation of DAEs, to further reduce the number of variables exposed to the collocation method and the NLP solver. The interaction between the proposed method and other types of NLP solution methods, such as active set methods, is also worth investigating.

Acknowledgments

We acknowledge Francesco Casella from Politecnico di Milano for granting the usage of the combined-cycle power plant model.

References

[1] F. Casella, F. Donida, and J. Åkesson, "Object-oriented modeling and optimal control: A case study in power plant start-up," in *18th IFAC World Congress*, (Milano, Italy), Aug. 2011.

[2] P.-O. Larsson, J. Åkesson, and N. Andersson, "Economic cost function design and grade change optimization for a gas phase polyethylene reactor," in *50th IEEE Conf. Decision and*

Control and European Control Conference, (Orlando, FL), Dec. 2011.

[3] K. Berntorp, B. Olofsson, K. Lundahl, B. Bernhardsson, and L. Nielsen, "Models and methodology for optimal vehicle maneuvers applied to a hairpin turn," in *Am. Control Conf. (ACC)*, (Washington, DC), pp. 2139–2146, 2013.

[4] B. Olofsson, H. Nilsson, A. Robertsson, and J. Åkesson, "Optimal tracking and identification of paths for industrial robots," in *18th IFAC World Congress*, (Milano, Italy), Aug. 2011.

[5] F. E. Cellier and E. Kofman, *Continuous System Simulation*. New York, NY: Springer, Mar. 2006.

[6] B. Bachmann, L. Ochel, V. Ruge, M. Gebremedhin, P. Fritzson, V. Nezhadali, L. Eriksson, and M. Sivertsson, "Parallel multiple-shooting and collocation optimization with OpenModelica," in *9th Int. Modelica Conf.*, (Munich, Germany), Sept. 2012.

[7] R. Franke, "Formulation of dynamic optimization problems using Modelica and their efficient solution," in *2nd Int. Modelica Conf.*, (Oberpfaffenhofen, Germany), Mar. 2002.

[8] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Computing*, vol. 1, no. 2, pp. 146–160, 1972.

[9] L. T. Biegler, *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. MOS-SIAM Series on Optimization, Philadelphia, PA: Mathematical Optimization Society and the Society for Industrial and Applied Mathematics, 2010.

[10] J. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming: Second Edition*. Advances in Design and Control, Philadelphia, PA: Society for Industrial and Applied Mathematics, 2010.

[11] F. Magnusson and J. Åkesson, "Collocation methods for optimization in a Modelica environment," in *9th Int. Modelica Conf.*, (Munich, Germany), Sept. 2012.

[12] J. Åkesson, K.-E. Årzén, M. Gäfvert, T. Bergdahl, and H. Tummescheit, "Modeling and optimization with Optimica and JModelica.org—languages and tools for solving large-scale dynamic optimization problem," *Computers and Chemical Engineering*, vol. 34, pp. 1737–1749, Nov. 2010.

[13] J. Åkesson, "Optimica—an extension of Modelica supporting dynamic optimization," in *6th Int. Modelica Conf.*, (Bielefeld, Germany), Mar. 2008.

[14] A. Wächter and L. T. Biegler, "On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[15] J. Andersson, J. Åkesson, and M. Diehl, "CasADi – A symbolic package for automatic differentiation and optimal control," in *Recent Advances in Algorithmic Differentiation*, Lecture Notes in Computational Science and Engineering, Berlin, Germany: Springer, 2012.

[16] B. Olofsson, K. Lundahl, K. Berntorp, and L. Nielsen, "An investigation of optimal vehicle maneuvers for different road conditions," in *7th IFAC Symp. Advances in Automotive Control (AAC)*, (Tokyo, Japan), pp. 66–71, 2013.

[17] K. Lundahl, K. Berntorp, B. Olofsson, J. Åslund, and L. Nielsen, "Studying the influence of roll and pitch dynamics in optimal road-vehicle maneuvers," in *23rd Int. Symp. Dynamics of Vehicles on Roads and Tracks (IAVSD)*, (Qingdao, China), 2013.

[18] R. Rajamani, *Vehicle Dynamics and Control*. Berlin Heidelberg: Springer-Verlag, 2006.

[19] K. Berntorp, "Derivation of a six degrees-of-freedom ground-vehicle model for automotive applications," Technical Report ISRN LUTFD2/TFRT--7627--SE, Department of Automatic Control, Lund University, Sweden, Feb. 2013.

[20] H. B. Pacejka, *Tire and Vehicle Dynamics*. Oxford, United Kingdom: Butterworth-Heinemann, 2nd edition ed., 2006.

[21] J. Wong, *Theory of Ground Vehicles*. Hoboken, NJ: John Wiley & Sons, 2008.

[22] HSL, "A collection of fortran codes for large scale scientific computation." `http://www.hsl.rl.ac.uk`, 2013.