Modeling Mixed-Integer Constrained Optimal Control Problems in AMPL

Christian Kirches,∗ Hans Georg Bock,∗ Sven Leyffer ∗∗

∗ Interdisciplinary Center for Scientific Computing, Ruprecht-Karls-Universität Heidelberg, Germany (e-mail: {christian.kirches, bock}@iwr.uni-heidelberg.de)
∗∗ Mathematics & Computer Science Division, Argonne National Laboratory, Argonne, IL, U.S.A. (e-mail: leyffer@mcs.anl.gov)

Abstract: Modeling languages and systems for simulation and optimization of continuous ODE/DAE systems are commonly available. For the most part, they focus on convenience of user interaction, and are tightly coupled to one or a few selected numerical methods. Control problems with discrete and hybrid controls, called mixed-integer optimal control problems (MIOCPs), have recently gained increased attention as the potential for optimization is high. The mixed-integer optimization community however most often considers problems without dynamics, and relies on symbolic modeling languages such as AMPL. Access to many advances MI(N)LP codes is provided by the NEOS Server for Optimization through the AMPL modeling language. Addressing this gap, we describe a set of extensions to the AMPL modeling language to conveniently model mixed-integer optimal control problems for ODE or DAE dynamic processes. These extensions are easily realized and do not require intrusive changes to the AMPL language standard or implementation itself. An example of an optimal control problem solver interfaced with AMPL is the “multiple shooting code for optimal control” MUSCOD-II, a direct and simultaneous method for ODE/DAE-constrained optimal control, and its extension MS-MINTOC for mixed-integer optimal control. As an example, we use the described AMPL extensions to model a heavy duty truck control problem.

Keywords: mixed-integer optimal control, differential-algebraic equations, domain specific languages

1. INTRODUCTION

This paper describes a set of extensions to the AMPL modeling language that extends AMPL’s applicability to mathematical optimization problems including dynamic processes. We describe AMPL syntax elements for convenient modeling of ordinary differential equation (ODE) and differential algebraic equation (DAE) constrained mixed-integer optimal control problems (MIOCPs), and show how our approach provides a generic way of tackling challenging mixed-integer dynamic optimization problems with the most recent and up-to-date methods developed in the MINLP community and accessible through AMPL.

Although the idea of a high-level modeling language for dynamic simulation and optimization problems is not new, most of the available development environments are designed for continuous problems, focus on the end-user’s convenience of interaction with the development environment, and are often closely tied to one or a few selected discretization schemes and solvers only. We refer to e.g. Barton and Pantelides (1993); Mattsson et al. (1997); Eldqvist and Brück (2001); Åkesson et al. (2008); Rutquist and Edvall (2010) for a few initiatives.

1.1 Problem Class

We consider the class (1) of mixed-integer optimal control problems on the time horizon [0, T]. Our goal is to minimize an objective function that includes an integral Lagrange term L on [0, T], a Mayer term E at the end point T of the time horizon. All objective terms depend on the trajectory (x(·), z(·)) : [0, T] → Rn x Rn of a dynamic process described in terms of a system of ODEs (1b) or DAEs (1b, 1c). This system is affected by continuous controls u(·) : [0, T] → Rn and integer-valued controls w(·) : [0, T] → Ωw from a finite discrete set Ωw := {w1, . . . , wnw} ⊂ RNw, |Ωw| = nw < ∞ (1g). Both control profiles are subject to optimization. Moreover, we allow the end time T (1h), the continuous model parameters p ∈ RNp, and the discrete–valued model parameters ρ ∈ Ωp := {p1, . . . , pnρ} ⊂ RNp, |Ωp| = np < ∞ (1i) to be subject to optimization as well. Control, parameters, and process trajectory may be constrained by...
inequality path constraints \( c(\cdot) \) (1d), equality constraints \( r^c(\cdot) \) and inequality constraints \( r^w(\cdot) \) coupled in time (1e), and decoupled equality and inequality constraints \( r^c(\cdot), r^w(\cdot), 1 \leq i \leq n_r \) (1f), imposed on a grid \( \{t_i\}_{i \leq n_t} \) of points on time horizon \([0, T]\). Constraints (1d-1f) include initial values and boundary values; mixed state-control constraints; periodicity constraints; and simple bounds on states, controls, and parameters. The optimal control problem is conveniently expressed as follows:

\[
\begin{align*}
\min_{p, \rho, T} & \quad \int_0^T L(t, x(t), z(t), u(t), w(t), p, \rho) dt \\
\text{s.t.} & \quad \dot{x}(t) = f(t, x(t), z(t), u(t), w(t), p, \rho), \quad 0 = g(t, x(t), z(t), u(t), w(t), p, \rho), \\
& \quad 0 \leq c(t, x(t), z(t), u(t), w(t), p, \rho), \quad 0 \leq r^c(x(0), z(0), x(T), z(T), p, \rho), \\
& \quad 0 \leq r^w(x(t_i), z(t_i), p, \rho), \quad 1 \leq i \leq N_t, \\
& \quad \min T \leq T \leq \max T, \\
& \quad \rho \in \Omega_p,
\end{align*}
\]

where \( t_i \in [0, T] \) in (1f), and (1b-1e, 1g) hold for all \( t \in [0, T] \).

### 1.2 Mixed-Integer Nonlinear Programming

Direct and simultaneous methods for tackling the class of optimization problems (1) posed in function spaces are based on the first-discretize-then-optimize approach. A discretization scheme is chosen and applied to objective (1a), dynamics (1b, 1c), and path constraints (1d) of the optimization problem in order to obtain a finite-dimensional counterpart problem accessible to mathematical programming algorithms, see e.g. Kirches (2011). This counterpart problem will usually be a high-dimensional and highly structured nonlinear problem (NLP) or mixed-integer nonlinear problem (MINLP). Thus, it readily falls into the domain of applicability of many mathematical optimization software packages already interfaced with AMPL, such as e.g. filterSQP (Fletcher and Leyffer, 1999), IPOPT (Wächter and Biegler, 2006), KNITRO (Byrd et al., 1999, 2000, 2006), MINOS (Murtagh and Saunders, 1993), SNOPT (Gill et al., 2002) and others for nonlinear programming and Bonmin (Bonami et al., 2005), Couenne (Belotti, 2009), MINLPBB (Leyffer, 1998), or FilMINT (Abhishek et al., 2010) for mixed-integer nonlinear programming.

### 1.3 The AMPL Modeling Language

AMPL is described by Fourer et al. (1990) and was designed as a mathematical modeling language for linear programming problems. It has later been extended to several more involved problem classes, such as mixed-integer nonlinear problems (MINLPs). AMPL’s syntax closely resembles the symbolic and algebraic notation used to present mathematical optimization problems and allows for fully automatic processing of optimization problems by a computer system. Significant advantages of this approach include the possibility for automatic differentiation, improved error checking, automated analysis for structural properties, and automated generation of model code for lower-level languages. The wealth of mathematical optimization codes available on the NEOS Server for Optimization (Gropp and More, 1997) that provide interfaces to read and solve problems modeled using the AMPL language underlines AMPL’s popularity and success.

### 2. NEW AMPL LANGUAGE ELEMENTS

We extend the AMPL modeling language by a facility that conveys the notion of time dependency of variables (such as e.g. \( x(t) \) and \( u(t) \) in (1)), of ODE and DAE constraints (1b, 1c), and of specially structured objective functions (1a) to an NLP or MINLP solver. To this end, we introduce the AMPL functions “\( \text{diff} \), “\( \text{eval} \), and “\( \text{integral} \), and the AMPL suffix “\( \text{type} \). We give (fictitious) examples of the syntax for all introduced AMPL constructs.

#### 2.1 ODE and DAE Constraints

We introduce an AMPL “user-function” \( \text{diff}(\text{var}, t) \) that is used in equality constraints to denote the left–hand side of an ODE (1b). The first argument \( \text{var} \) denotes the differential state variable for which a right–hand side is to be defined. The second argument \( t \) is expected to denote the independent time variable. The appearance of a call to \( \text{diff} \) in an AMPL constraint expression allows us to distinguish an ODE constraint from other constraint types.

\[
\text{ode}_1: \text{diff}(x1, t) = x1^2 + 2*x1 - u0;
\]

An equality constraint on all of \([0, T]\) involving state or control variables is understood as a DAE constraint (1c) if it is not an ODE constraint.

\[
\text{dae}_2: 0 = x1^2 + x2*z1^2 - x3;
\]

Most DAE solvers will expect the DAE system to be of index 1, i.e., the number of DAE constraints (1c) must match the number of algebraic states \( z(t) \) and the Jacobian w.r.t. these must be regular in a neighborhood of the DAE trajectory. While the first requirement can be readily checked, the burden of ensuring regularity is put on the modeler creating the AMPL model. It can be verified locally by a DAE integrator at runtime.

#### 2.2 Time Dependent Variables

While an MINLP may assume a flat and uniform view of its variables, an MIQCP solver needs to distinguish between different roles AMPL variables should play in problem (1). We introduce the AMPL suffix \( \text{type} \) and describe the rules of inference.

**Independent Time and Final Time.** We expect any AMPL optimal control model to declare a variable representing independent time, denoted by \( t \) in (1), detected by its appearance in a call of the differential \( \text{diff} \). For the end point of the time horizon \([0, T]\), the user is free to either introduce a variable if \( T \) is free and subject to optimization, or to use a numerical constant if \( T \) is fixed.

\[
\text{var } t; \\
\text{var } T >= 1, <= 10;
\]
**Differential and Algebraic State Variables.** States, denoted by \( x(\cdot) \) in (1), are detected by their appearance in a call of the differential \( \text{diff} \), which defines the differential right-hand side for this state.

\[
\text{var } x1 >= -10, <= 10;
\]

Algebraic state variables are denoted by \( z(\cdot) \) in (1). Since a DAE constraint may involve multiple algebraic state variables, there is no one–one correspondence between DAE constraints and DAE variables. Hence, we require users to flag them using a new AMPL suffix named \( \text{type} \), set to the symbolic value \( "\text{dae}" \).

\[
\text{var } z2 >= -10, <= 10 \text{ suffix type "dae"};
\]

**Continuous and Integer Control Variables** Control variables, denoted by \( u_i(\cdot) \) and \( w_i(\cdot) \) in (1), also need to be flagged by setting the new AMPL suffix \( \text{type} \) to one of several symbolic values representing choices for the control discretization. Currently, \( "\text{u0}" \) (piecewise linear), \( "\text{u1c}" \) (piecewise cubic), and \( "\text{u3c}" \) (piecewise cubic continuous) are available. Integer controls are declared using the existing AMPL keywords \( \text{integer} \) and \( \text{binary} \).

\[
\text{var } u1 >= 0, <= 100 \text{ suffix type "u1"};
\]

\[
\text{var } w1 \text{ binary suffix type "u0"};
\]

**Continuous and Integer Parameters** Any AMPL variable not inferred to be independent or final time, differential or algebraic state, or control according to the rules described above is considered a model parameter \( p_i \) or \( \rho_i \) that is constant in time but may be subject to optimization, for example, in parameter estimation problems.

**2.3 Constraints on Time Dependent Variables**

In this section we show how to extend AMPL to model the various types of constraints found in problem (1).

**Path Constraints.** An inequality constraint calling neither \( \text{eval} \) nor \( \text{diff} \) becomes an inequality path constraint (1d). Equality constraints are DAE constraints (§2.1).

\[
\text{path}_1: 3*x1 + x2 >= 0;
\]

**Point Constraints.** Point constraints (1e, 1f) impose restrictions on the values of trajectories at discrete points \( t_i \in [0,T] \). We introduce a user function \( \text{eval(expr,time)} \) that denotes the evaluation of an expression involving time dependent variables in a given fixed time point in \( [0,T] \). For problems with fixed end-time \( T \), time points are absolute values (in the range \( [0,T] \)). For problems with free final time \( T \), all time points are relative times on the normalized time horizon \([0,1]\).

\[
\text{ini}_1: \text{eval}(x1, 0) = 3;
\]

\[
\text{end}_1: \text{eval}(x1^2+x2^2, T) = 1;
\]

**2.4 Lagrange and Mayer Objective**

Problem (1) uses an objective function that consists of a Lagrange-type integral term \( L \) and a Mayer-type end-point term \( E \). In addition, it is advantageous to detect least-squares structure of the integral Lagrange-term that can be exploited after discretization. We also support a point least-squares term in the objective, arising, for example, in parameter estimation problems.

We introduce an AMPL function \( \text{integral(expr,T)} \) to denote a Lagrange-type objective function. The first argument \( \text{expr} \) is the integrand, and the second one, \( T \), is expected to denote the final time variable or value \( t \).

The function \( \text{eval(expr,time)} \) already introduced can be used to model Mayer-type (\( \text{time is } T \)) objective terms.

\[
\text{of: } \text{integral(u1^2+u2^2, T) + 0.1*eval(x1^2, T)};
\]

**2.5 Benefits**

The described approach maps the components of the MIOCP (1) to the AMPL language. It thereby removes the need for tedious encoding of fixed discretization schemes, as has been done, for example, for collocation schemes in Dolan et al. (2004). This clearly improves readability and may significantly reduce the model’s code size. Adaptive choice and iterative refinement of the discretization become possible, see e.g. Kameswaran and Biegler (2008). At the same time, this idea opens up the possibility of using other and more involved discretization schemes that cannot be expressed directly in AMPL. One example is the direct multiple–shooting method (Bock and Plitt, 1984; Leineweber et al., 2003) that enables the use of state–of-the-art ODE and DAE solvers (see e.g. Albersmeyer (2010); Petzold et al. (2006)) with increased opportunities for exploiting structure and adaptivity.

**3. COUPLING TO SOLVERS: AN EXAMPLE**

The TACO “Toolkit for AMPL Control Optimization” (Kirches and Leyffer, 2011) targets developers of (MI)NLP codes and optimal control codes, and is publicly available. It implements the presented inference rules and provides convenient access to a MIOCP representation of an extended AMPL model. Given a description of a discretization method, an MINLP solver may access this MIOCP representation and apply the discretization method to time dependent functions and variables in order to obtain an (MI)NLP to work with.

**3.1 The Solvers MUSCOD-II & MS-MINTOC**

We discuss this at the example of MUSCOD-II (Leineweber et al., 2003; Diehl et al., 2001) and its extension MS-MINTOC (Sager, 2005; Sager et al., 2011; Diehl et al., 2001) for DAE-constrained mixed-integer optimal control. MUSCOD-II is based on a multiple shooting discretization in time (Bock and Plitt, 1984), and implements a direct and all-at-once approach to solving DAE-constrained optimal control problems. Therein, ODE/DAE initial value problems on the multiple shooting intervals are solved, and sensitivities of the obtained solutions are computed (see Leineweber et al., 2003).

**3.2 Evaluating AMPL Functions and Derivatives**

Implementations of MUSCOD-II optimal control problems in a compiled language like C or Fortran provide functions
for evaluating values and possibly gradients and Jacobians of all problem functions in (1). Objective and constraints are evaluated by the underlying NLP solver, while ODE and DAE constraints are evaluated by the ODE/DAE integrator. The compiled language model is replaced by a generic model provided by TACO. Its purpose is to evaluate, with the help of the AMPL solver library, the corresponding AMPL objective or constraint instead, and to provide gradients, Jacobians, and sparsity information when required. For a user of an AMPL-interfaced optimal control solver, this step is entirely transparent.

4. AN EXAMPLE AND A PROBLEM COLLECTION

In this section we model a combinatorial heavy duty truck control problem using AMPL. The aim here is to compute time/energy optimal cruise controls for a heavy duty truck including gear shift sequences, based on 3D terrain data and a detailed engine model. We study the problem’s representation as an extended AMPL model. A model-predictive variant and further details can be found in (Terwen et al., 2004; Kirches, 2011).

4.1 Dynamic Truck Model

The switched ODE model is a point-mass model comprising four differential states \( t, v, M_{\text{ind}}, M_{\text{brk}} \) and three input controls \( (R_{\text{ind}}, R_{\text{brk}}, y) \), being discrete. Traveled distance \( s \) (in meters) is chosen as the independent variable. Time \( t(s) \) is introduced as the first state, \( t(s) = 1/v(s) \). Here \( v(s) \) denotes velocity, determined from the sum of directed torques,

\[
m(v(s))v(s) = (M_{\text{acc}} - M_{\text{brake}}) \frac{i_A}{r_{\text{stat}}} - M_{\text{air}} - M_{\text{road}}.
\]

The engine and brake torques \( M_{\text{ind}} \) and \( M_{\text{brk}} \) are computed from their derivatives \( R_{\text{ind}} \) and \( R_{\text{brk}} \) which are input controls,

\[
\dot{M}_{\text{ind}}(s) = R_{\text{ind}}(s)/v(s), \quad \dot{M}_{\text{brk}}(s) = R_{\text{brk}}(s)/v(s).
\]

Remaining terms are computed from algebraic formulas: the accelerating torque \( M_{\text{acc}} \) is computed from the ratio \( \frac{i_T}{\eta_T} \) and efficiency \( \eta_T \) associated with the selected gear \( y \). Braking forces \( M_{\text{brake}} \) are due to brake controls and friction \( M_{\text{fric}} \),

\[
M_{\text{acc}}(s) := \frac{i_T}{\eta_T}(s) \eta_T(s) M_{\text{ind}}(s),
\]

\[
M_{\text{brake}}(s) := M_{\text{brk}}(s) + \frac{i_T(s)}{\eta_T(s)} M_{\text{fric}}(v(s)).
\]

Additional braking forces due to turbulent friction \( M_{\text{air}} \) and road conditions \( M_{\text{road}} \) are taken into account,

\[
M_{\text{air}}(s) := \frac{1}{2} c_w A \rho_{\text{air}} v^2(s),
\]

\[
M_{\text{road}}(s) := m g \sin \gamma(s) + f r \cos \gamma(s).
\]

Here \( c_w \) is the shape coefficient, \( A \) denotes the flow surface, and \( \rho_{\text{air}} \) the air density. Further, \( m \) is the vehicle’s mass, gravity is denoted by \( g \), \( \gamma(s) \) denotes the road’s slope, and \( f r \) is a rolling friction coefficient. The engine speed \( n_{\text{eng}} \) depending on the gear \( y \) is obtained from velocity as follows,

\[
n_{\text{eng}}(s) := v(s) i_A \frac{i_T(y(s))}{2\pi r_{\text{stat}}},
\]

where \( i_A \) is the fixed axle transmission ratio and \( r_{\text{stat}} \) is the static tyre radius. Further details as well as parameter values and examples of pre-recorded map and engine data sets are found in Kirches (2011).

4.2 Truck Control Problem

We now focus on the mixed-integer optimal control problem formulation. We design a performance criterion and mention state and control bounds as well as path constraints.

Objective The integral cost criterion to be minimized on a horizon \([0, S]\) is composed of a weighted sum of three different objectives. The deviation of the truck’s velocity from the desired one is penalized,

\[
\Phi_{\text{dev}} := \lambda_1 \int_0^S (v(s) - v_{\text{des}}(s))^2 \, ds.
\]

Second, the fuel consumption is found from a fuel consumption rate map \( Q(n_{\text{eng}}, M_{\text{ind}}) \) and penalized,

\[
\Phi_{\text{fuel}} := \lambda_2 \int_0^S Q(n_{\text{eng}}(s), M_{\text{ind}}(s))/v(s) \, ds.
\]

Third, rapid changes of the engine torque degrade driving comfort, and are regularized,

\[
\Phi_{\text{cont}} := \lambda_3 \int_0^S (R_{\text{ind}}(s) + R_{\text{brk}}(s))^2/v(s) \, ds.
\]

The choice of \( \lambda \) allows for a gradual selection of a compromise between meeting a desired velocity and following an economic operating mode of the truck. Weighting and summing up, we obtain a combined objective function.

Constraints On the horizon, certain mechanical constraints such as velocity and engine speed limits are imposed. Beside the bounds on the truck input controls and on the system’s states, the truck’s velocity \( v(s) \) is subject to velocity limits imposed by law and by the road’s curvature,

\[
v(s) \leq v_{\text{law}}(s), \quad v(s) \leq v_{\text{curve}}(s), \quad s \in [0, S].
\]

The indicated and brake torques must respect state-dependent upper limits as specified by the engine characteristics

\[
0 \leq M_{\text{ind}}(s) \leq M_{\text{ind,max}}(n_{\text{eng}}(s)),
\]

\[
0 \leq M_{\text{brk}}(s) \leq M_{\text{brk,max}}(n_{\text{eng}}(s)), \quad s \in [0, S].
\]

Finally, the engine’s speed \( n_{\text{eng}}(s) \) depends on the velocity and the selected gear, and must stay within prescribed limits according to the engine’s specification,

\[
n_{\text{eng,min}} \leq n_{\text{eng}}(s) \leq n_{\text{eng,max}}, \quad s \in [0, S].
\]

Data for \( \gamma(s), \kappa(s), \) and \( v_{\text{law}}(s) \) is assumed to be available for the route to travel, see Terwen et al. (2004) for technical details. The bounds \( v_{\text{curve}}(s), M_{\text{ind,max}}, M_{\text{brk,max}}, n_{\text{eng,min}}, n_{\text{eng,max}} \) are vehicle- and engine-specific data sets.

4.3 The Truck Control Problem Modeled in AMPL

From sections §4.1 and §4.2 we come up with the following AMPL representation of the heavy duty truck control problem:

# independent variable
var s;

# position [m]
var v := 0; # time [s]

# differential states and bounds
var v >= 0, <= 40; # velocity [m/s]
var Mind >= 0, <= 3500; # engine torque [Nm]
var Mbrk >= 0, <= 8000; # brake torque [Nm]
# controls and bounds

var Rind := -500, <= 750;  # engine torque rate [Nm/s]
var Rbrk := -1000, <= 500;  # brake torque rate [Nm/s]
var y(1..16) binary;  # selectors for 16 gears

# constant parameters

param l3 := 1.0e-4; # of weight: comfort
param l2 := 1.0;    # of weight: fuel consumption
param l1 := 0.1;    # of weight: desired velocity
param etaT{i in 1..16}; # gearbox efficiencies
param iT{i in 1..16}; # gearbox transmission ratios
param nengmax := 2100; # maximum engine speed [1/min]
param nengmin := 700; # minimum engine speed [1/min]
param rho := 1.293; # density of air [kg/m^3]
param cw := 0.54; # aerodynamic shape coefficient
param pi := 3.14159;
param fr := 0.006; # coefficient of rolling friction
param rstat := 0.501; # static tyre radius [m]
param iA := 2.801; # rear axle transmission ratio
param g := 9.81; # gravity [N]
param m := 40000; # vehicle mass [kg]

# defined variables (automatically substituted)

var Mroad := m * g * (sin(SLOPE(s)) + fr*cos(SLOPE(s))); # road resistance
var Mair := 0.5 * cw * A * rho * v^2; # air resistance
var Mbrake := Mbrk + sum{i in 1..16}(y[i]*iT[i])*MFRIC(neng); # brake action

# # objective

minimize

of: integral (l1*(v-VDESIRE(s))^2 + l2*FUEL(neng,Mind)/v + l3*(Rind + Rbrk)^2/v, s);

# differential equations

subject to

ode_v: diff(v,s) = ((Macc-Mbrake)*iA/rstat-Mair-Mroad)/(m*v);
ode_t: diff(t,s) = 1/v;

# path constraints

subject to

v_law_limit: v <= VLAW(s);
v_curve_limit: v <= VCURVE(s);
engine_limit: Mind <= MINDMAX(neng);
brake_limit: Mbrk <= MBRKMAX(neng);

# several possible formulations exist for this constraint

rpm_limit: nengmin <= neng <= nengmax;

# initial conditions, or embedding in predictive control

initial: eval(t,0) = 0;
initial: eval(v,0) = VCUR;
Mind_initial: eval(Mind,0) = MINDCUR;
Mbrk_initial: eval(Mbrk,0) = MBRKCUR;

# select only one gear at a time

sos: sum(i in 1..16)y[i] = 1;

5. CONCLUSIONS

We have described an extension of the AMPL modeling language beyond the domain of MINLPs that allows one to conveniently model mixed-integer DAE-constrained optimal control problems in AMPL. Our extension is realized through the TACO toolkit for AMPL control optimization, which is publicly available and serves as an interface between AMPL and an optimal control solver.

We have used this new toolkit to implement an AMPL interface for the optimal control software packages MUSCOD-II and its mixed-integer optimal control extension MS-MINTOC. Through AMPL and TACO, both solvers are now available on the NEOS Server for Optimization.

The modeling of a heavy duty truck control problem in AMPL using the extensions has shown the applicability of the new approach: Short model code, improved readability, and flexibility in the choice of a discretization scheme are only a few of the benefits.

The class (1) and its corresponding AMPL representation may be extended to include a number of additional features commonly encountered in modeling of dynamic processes. These include e.g. fully implicit DAE systems, implicitly switched systems, non-smooth constraints, and multistage problem setups. The solver we interfaced with AMPL relies on the direct multiple shooting discretization. In the future, we envision the implementation of a number of other schemes for evaluating ODE and DAE constraints, such as collocation schemes. This will enable users to choose from a larger subset of the NLP and MINLP solvers available on NEOS.

ACKNOWLEDGEMENTS

We thank Sebastian Sager for permission to use the mixed-integer optimal control algorithm MS-MINTOC.

REFERENCES


4.4 The mintoc.de Collection of Problems and Solutions

An extensible online library of many further continuous and mixed-integer optimal control problems has been described in Sager (2011) and is available through the web site http://mintoc.de. This site holds a wiki-based, publicly accessible library and is actively encouraging the community to contribute interesting problems modeled in various languages. So far, models in C, C++, JModelica, in AMPL using encoded collocation schemes, and in AMPL using the presented TACO extensions can be found together with best known optimal solutions and optimal control and state profiles.