Using Baumgarte's Method for Index Reduction in Modelica

Scott A. Bortoff[1]

[1]Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, bortoff@merl.com

Abstract

We show by example how Baumgarte's method can be used in a Modelica model to reduce the differential algebraic equation index prior to compilation. This has advantages for some constrained mechanical systems especially those with closed-chain kinematics, including improved initialization and enabling model-based control system design. We derive models for a simple pendulum, a delta robot and for elevator cable sway as case studies. The models are used for simulation and also for dynamic analysis and to design and realize feedback controllers.
Keywords: DAE, index reduction, robotics, control

1 Introduction

Modeling and simulation of some types of constrained mechanical systems, such as closed kinematic chains, can be challenging in the Modelica language. One reason is because component-oriented modeling for such systems results in a set of high-index differential algebraic equations (DAEs). Modelica compliers, such as Dymola, use the method of "dummy derivatives" (Mattsson and Söderlind, 1993; Bachmann, 2006; Cellier, 2006) to reduce the index for very good and fundamental reasons. However, for closed chains it has some disadvantages, and there are other methods (Bauchau and Laulusa, 2007), which have advantages especially for consistent initialization and use cases beside simulation, such as control system design.

In this paper we show, by example, how Baumgarte's method of index reduction (Baumgarte, 1972, 1983) can be used in Modelica to reduce the index of a constrained mechanical system *prior* to compilation. Our primary example is a delta robot, for which we derive a singularity-free, index 1 DAE. No automatic index reduction is done at compile time, and no dynamic state selection is required at simulation time. We find that the method is amenable to Modelica's object oriented modeling paradigm, and results in simulation code that can be, at least anecdotally, faster. We construct several components of a feedback controller directly from the index-1 system model, and show how consistent initial conditions can be computed in this formulation. We provide a second example, elevator cable sway, in which the method is vital to simulation and feedback control system design. Interestingly, the method can model certain types of time-varying constraints such as loss-of contact or constraint breaking.

Baumgarte's method should be considered as a viable *alternative* - not a general replacement - to the automatic index reduction algorithms that are built into Modelica compilers. It is appropriate for certain situations in which these algorithms either fail to reduce the index, or result in complex and therefore slow, simulation code. The method has been criticized in the numerical analysis literature (Bauchau and Laulusa, 2007), primarily because selection of values for its parameters, described later, is problem-dependent, and because it results in a system of equations that is of dimension larger than the number of degrees of freedom in the problem. As a result, a simulation can "drift," meaning that the algebraic constraint is not enforced exactly during a simulation. For some use cases, this could be disastrous and the method should not be used. However, for our applications, we find these criticisms to be inconsequential. The method's two parameters are easy to tune, and the drift is on the order of the solver tolerance, so it can be reduced by reducing the solver tolerance. The drift vanishes when the mechanical system is at rest.

2 Toy Pendulum Example

Consider the equations of motion of a simple pendulum expressed in Cartesian coordinates,

$$\dot{x}_1 = v_1 \tag{1a}$$
$$\dot{x}_2 = v_2 \tag{1b}$$
$$M\dot{v}_1 = -2x_1\lambda \tag{1c}$$
$$M\dot{v}_2 = -2x_2\lambda - g \tag{1d}$$
$$0 = h(x) = x_1^2 + x_2^2 - L^2 \tag{1e}$$

where M is the pendulum bob mass, L is the rod length, g is the acceleration due to gravity, $x = [x_1\ x_2]^T$ is the position in Cartesian coordinates of the pendulum bob, $v = [v_1\ v_2]^T$ is the velocity, and λ is the Lagrange multiplier which corresponds to the tension in the rod required to maintain the constraint $h(x) = 0$ in (1e). System (1) is an index-3 DAE in variables x, v and λ. Modelica code for the pendulum is (Fritzon, 2015)

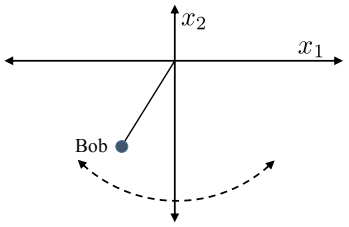

Figure 1. Pendulum.

```
der(x1) = v1;
der(x2) = v2;
M * der(v1) = -2.0 * x1 * lambda;
M * der(v2) = -2.0 * x2 * lambda - M * g;
h = x1^2 + x2^2 - L^2;
h = 0.0;
```

When this is compiled by Dymola, for example, the index is reduced using the method of "dummy derivatives," resulting in a system with two differential states and three algebraic states. However, for any single choice of differential states, there exists a kinematic configuration in which the solver Jacobian becomes singular. This means that at least two representations are required to cover the complete configuration space, and the solver switches between them. Figure 2 shows the Message window after compiling this model, indicating that two sets of two dynamics states were selected.

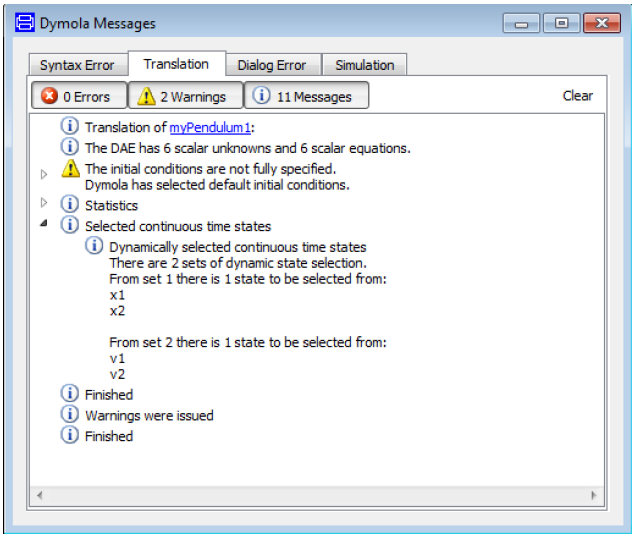

Figure 2. Message Window showing two sets of two dynamic states for the method of "dummy derivatives."

Baumgarte's method replaces (1e) with a linear combination of its first two derivatives,

$$h''(x,v,\lambda) + \alpha_1 h'(x,v) + \alpha_0 h(x) = 0\,, \qquad (2)$$

where $\alpha_i > 0$ for $i = 0, 1$, and $s^2 + \alpha_1 s + \alpha_0$ is Hurwitz (all roots in the open left-half plane). Values for α_i are tuned depending on the specific problem. Large values have smaller drift, but result in a stiff system. We find that placing the roots at locations that are on the order of the system time constant is sufficient. The resulting system (1a)-(1d) and (2) is an index-1 DAE that has the same solution as (1), which is shown below, and can be numerically integrated with an index-1 solver such as DASSL. Modelica code for the pendulum reduced via Baumgarte's method is

```
der(x1) = v1;
der(x2) = v2;
M * der(v1) = -2.0 * x1 * lambda;
M * der(v2) = -2.0 * x2 * lambda - M * g;
h0 = x1^2 + x2^2 - L^2;
h1 = der(h0);
h2 = der(h1);
0.0 = h2 + alpha1 * h1 + alpha0 * h0;
```

where we take advantage of Modelica's automatic differentiation. Figure 3 shows the message window for Baumgarte's method. We see that four static states are selected.

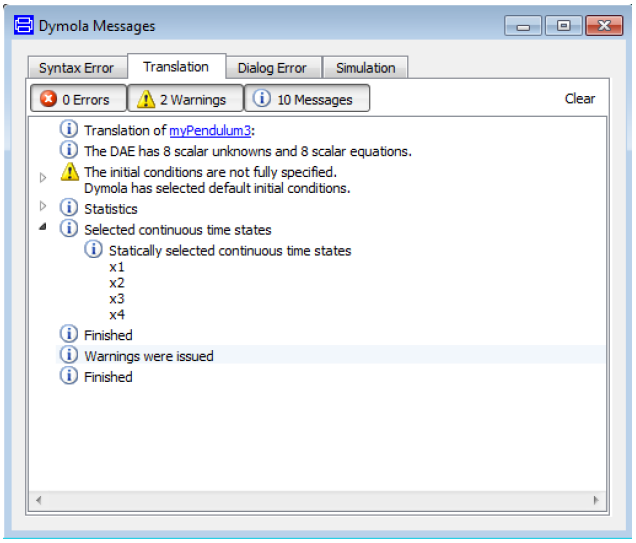

Figure 3. Message Window showing one set of four dynamic states for Baumgarte's method.

Simulation of this model is about 5x faster than the first system for the same simulation parameters, but it is less accurate. Figure 4 shows the constraint $h(x)$ for a portion of the simulation. Baumgarte's method drifts away from $h(x) = 0$ by an amount on the same order as the solver tolerance (1e-4). On the other hand, the method of dummy derivative implicitly enforces $h(x) = 0$ for all time, which is one reason why it is used in compilers.

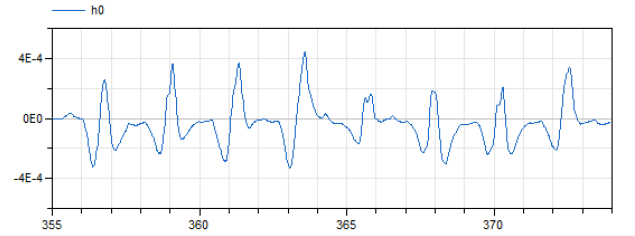

Figure 4. Constraint $h(x)$ for the pendulum example, Baumgarte's method.

It is useful to understand the geometric structure of the index-1 system (1a)-(1d) and (2). Define $z_0 = h(x)$ and $z_1 = h'(x,v)$. Following (Isidori, 1989), define $\xi = [z_0\ z_1]^T \in \mathbb{R}^2$ to be the "linear" part. Then there exist coordinates $\eta \in \mathbb{R}^2$ which are functions of x and v (after eliminating λ through algebraic manipulation) so that (1a)-(1d) and (2) can be written locally in so-called Zero Dynamics

DOI 10.3384/ecp19157333

Normal Form (Isidori, 1989),

$$\dot{\eta} = f(\eta, \xi) \tag{3a}$$
$$\dot{\xi} = A\xi, \tag{3b}$$

where the two eigenvalues of A are located at the roots of

$$s^2 + \alpha_1 s + \alpha_0 = 0, \tag{4}$$

and the two-dimensional zero dynamics

$$\dot{\eta} = f(\eta, 0) \tag{5}$$

are the dynamics of the pendulum. In other words, we simulate a four-dimensional system with state $[x\ v]^T \in \mathbb{R}^4$, but there is an attractive two-dimensional manifold in $\mathbb{R}^4$, defined by $\xi = 0$, on which the pendulum dynamics exist and evolve according to (5). The ξ-dynamics are exponentially stable, and once they converge to 0, do not affect x or v. This has two important implications. First, we may initialize the system at a state $[x_0\ v_0]^T \in \mathbb{R}^4$ nearby $\xi = 0$, and the state it will converge exponentially to the constraint manifold $\xi = 0$. This can be useful to compute consistent initial conditions by starting with an inconsistent initial condition and simulating the system until the exponentially stable part has converged. Second, if we linearize (1a)-(1b) and (2), we expect two pole-zero cancellations at the roots to (4). In a control design situation, these modes are exponentially stable, and are uncontrollable and unobservable, and may therefore be removed with a Hankel-norm model truncation (Skogestad and Postlethwaite, 2005) because their corresponding Hankel singular value is zero. The resulting reduced-order model is two-dimensional (because we started with a system of dimension four, and removed the two modes) and is equivalent to a linearization obtained otherwise, e.g., if we reduced the index using the method of dummy derivatives and then linearized it.

2.1 Breaking Pendulum

One advantage that Baumgarte's method offers is simulation of breaking constraints, which is an example of multi-mode modeling (Elmqvist et al., 2017). Consider the situation in which the pendulum rod will fracture if its tension exceeds a threshold. This situation is difficult to model using conventional methods, because the index changes from 3 to 0 when the rod breaks. It can be modeled with Baumgarte's method because the number of equations and variables remains constant before and after the break.

```
der(x1) = x3;
der(x2) = x4;
M * der(x3) = rhsX;
M * der(x4) = rhsY;
if lambda < lambdaMax then
 rhsX = -2.0 * x1 * lambda;
 rhsY = -2.0 * x2 * lambda - M * g;
 0.0 = h2 + alpha1 * h1 + alpha0 * h0;
else
 rhsX = 0.0;
 rhsY = -M * g;
 lambda = lambdaMax + epsilon;
end if;
h0 = x1^2 + x2^2 - L^2;
h1 = der(h0);
h2 = der(h1);
```

(Note that the value of `lambda` should be zero after the break, but we set it to an arbitrary `lambdaMax+epsilon` to avoid switching back after the break.) Figure 5 shows the result of a simulation.

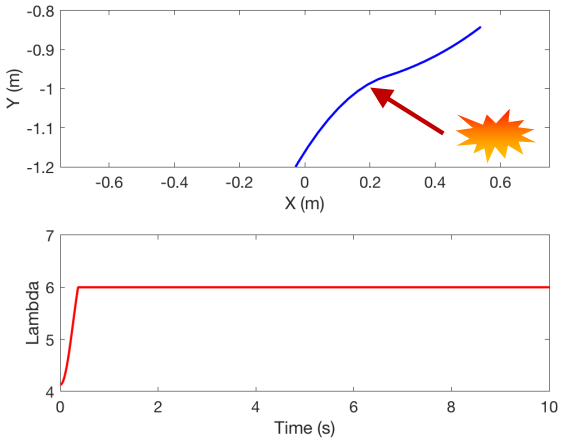

Figure 5. Simulation of breaking pendulum, in (x,y)-coordinates (top), and the Lagrange multiplier (bottom) .

3 Delta Robot Model

Next we use the same method to derive a model of a delta robot (Clavel, 1990) pictured in Figure 6, consisting of three symmetric arms constrained kinematically by universal joints at the end effector. Each arm consists of a proximal link, rigidly attached to a servomotor shaft at the proximal end, and a pair of parallel distal links that are attached to the proximal link via a pair of universal joints. The six distal links are attached to the end effector by universal joints such that the pair of arm distal links remain parallel, and the orientation of the end effector is invariant.

Delta robots are closed-chain mechanisms. Unlike the kinematics of serial chain robots (Spong and Vidyasagar, 2004), the forward kinematics of the delta robot (the function from actuated joint angles to the location of the end effector) cannot be expressed analytically (Merlet and Gosselin, 2008), making formulation of dynamic (and inverse dynamic) equations of motion more difficult (Guglielmetti, 1994; St. and C., 2003; Merlet and Gosselin, 2008; Brinker et al., 2015).

We derive the robot dynamics as in (Bortoff, 2018) first by defining the dynamics for each independent arm, assuming it is unconstrained, and then adding the holonomic coupling constraint representing the end effector. The resulting index-3 DAE is stabilized using Baumgarte's method, giving an index-1 DAE.

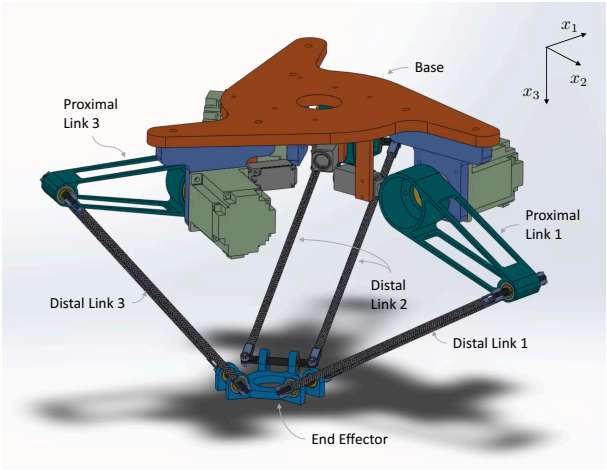

Figure 6. Delta robot.

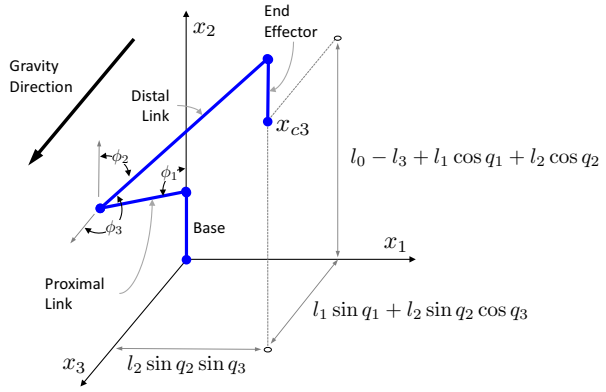

Figure 7. Delta robot arm coordinates with end effector location x_{c3} indicated.

3.1 Arm Dynamics

In deriving the dynamics of each arm, we can lump together the two distal links into a single effective link. Referring to Figures 6-7 in which the fixed "world" frame has axes labeled $[x_1, x_2, x_3]$, let $\phi = [\phi_1, \phi_2, \phi_3]^T$ be the generalized angular position for the arm, defined as follows. The servomotor angle is ϕ_1, which is the rotation of the proximal link about the x_1-axis, measured with respect to the x_2-axis. The universal joint position is represented with ϕ_2 representing the rotation *about* the x_1-axis measured with respect to the x_2-axis, and ϕ_3 representing the rotation *about* the x_2-axis measured with respect to the $x_2 - x_3$ plane. Note that, in these coordinates, the universal joint has a singularity at $\phi_2 = 0$. However, this is outside the range of motion of the robot once the three arms are kinematically constrained by the end effector.

Assuming that the distal links are thin rods, i.e., neglecting the inertia of the distal link about its longitudinal axis, the kinetic energy of each arm, including 1/3 the mass of

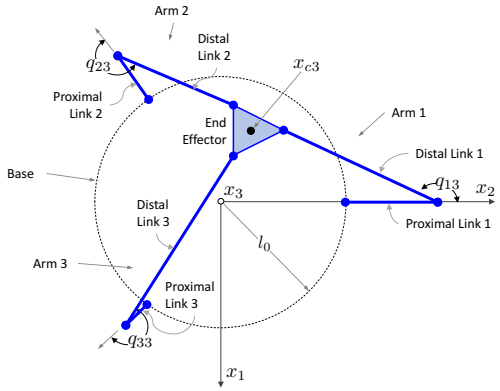

Figure 8. Delta robot coordinates, bottom view, looking up.

the end effector, is

$$T(\phi,\dot{\phi}) = \frac{1}{2}m_1\dot{x}_{c1}^T\dot{x}_{c1} + \frac{1}{2}m_2\dot{x}_{c2}^T\dot{x}_{c2} + \frac{1}{6}m_3\dot{x}_{c3}^T\dot{x}_{c3} + \frac{1}{2}J_1\dot{\phi}_1^2 + \frac{1}{2}J_2\left(\sin(\phi_2)^2\dot{\phi}_3^2 + \dot{\phi}_2^2\right), \quad (6)$$

where the position of the center of mass of the proximal link is

$$x_{c1} = \begin{bmatrix} 0.0 \\ l_{c1}\cos(\phi_1) \\ l_{c1}\sin(\phi_1) \end{bmatrix}, \quad (7)$$

the position of the center of mass of the distal link is

$$x_{c2} = \begin{bmatrix} l_{c2}\sin(\phi_2)\sin(\phi_3) \\ l_1\cos(\phi_1) + l_{c2}\cos(\phi_2) \\ l_1\sin(\phi_1) + l_{c2}\sin(\phi_2)\cos(\phi_3) \end{bmatrix}, \quad (8)$$

the position of the center of mass of the end effector is

$$x_{c3} = \psi(\phi) := \begin{bmatrix} l_2\sin(\phi_2)\sin(\phi_3) \\ l_0 - l_3 + l_1\cos(\phi_1) + l_2\cos(\phi_2) \\ l_1\sin(\phi_1) + l_2\sin(\phi_2)\cos(\phi_3), \end{bmatrix}, \quad (9)$$

the velocities $\dot{x}_{c1}$, $\dot{x}_{c2}$ and $\dot{x}_{c3}$ are computed by the chain rule to be functions of ϕ, $\dot{\phi}$ and the parameters are listed in Table 1. Note that the forward kinematics of the arm are defined as $\psi(\phi)$ in (9). The gravitational potential energy of each arm is

$$V(\phi) = -g((l_{c1}m_1 + l_1(m_2 + m_3/3))\sin(\phi_1) + (l_{c2}m_2 + l_2m_3/3)\sin(\phi_2)\cos(\phi_3)), \quad (10)$$

where gravity points along the positive x_3 axis and 1/3 of the mass of the end effector is included in each arm. The Lagrangian

$$L(\phi,\dot{\phi}) = T(\phi,\dot{\phi}) - V(\phi) \quad (11)$$

is used to define the arm equations of motion with Lagrange's equation,

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\phi}} - \frac{\partial L}{\partial \phi} = bu, \quad (12)$$

DOI 10.3384/ecp19157333

Table 1. Delta robot parameter definitions.

Symbol	Description (Units)
l_0	Base radius (m)
l_1	Length of proximal link (m)
l_2	Length of distal link (m)
l_3	Width of end effector (m)
l_{c1}	Distance to proximal link center of mass (m)
l_{c2}	Distance to distal link center of mass (m)
m_1	Mass of proximal link (kg)
m_2	Mass of distal mass (kg)
m_3	Mass of end effector (kg)
J_1	Rotational inertia, proximal link ($\mathrm{kg}\cdot\mathrm{m}^2$)
J_2	Rotational inertia, distal link ($\mathrm{kg}\cdot\mathrm{m}^2$)

giving

$$m(\phi)\ddot{\phi} + c(\phi,\dot{\phi}) + g(\phi) = bu, \tag{13}$$

where m is the 3×3 inertia matrix, c is the 3×1 vector of Coriolis and centripetal torques, g is the 3×1 vector of torques due to gravity, $b=[1,0,0]^T$ and u is the servomotor torque. Expressions for m, c and g are given in Appendix 1.

3.2 Robot Lagrangian Dynamics

Each of the three arms is identical except for a 120° rotation about the z-axis. To represent the dynamics of the full robot, we sum the unconstrained Lagrangians for each arm (11), and augment the result with the holonomic constraints that equate the x_{c3} positions of the end effectors of each arm (9) in the world coordinates. Lagrange's equation gives the constrained dynamical equations.

Referring to Figure 8, define $q_i \in \mathbb{R}^3$ for $1 \le i \le 3$, to be the generalized angular position of each of the three arms, replacing the ϕ-notation used in Section 3.1. Define $q=[q_1,q_2,q_3]^T \in \mathbb{R}^9$ and the unconstrained Lagrangian as

$$L_u(q,\dot{q}) = L(q_1,\dot{q}_1) + L(q_2,\dot{q}_2) + L(q_3,\dot{q}_3),$$

and form the augmented robot Lagrangian as

$$L_a(q,\dot{q}) = L_u(q,\dot{q}) + \lambda^T h(q), \tag{14}$$

where the constraint $h(q):\mathbb{R}^9 \to \mathbb{R}^6$ is

$$h(q) = \begin{bmatrix} \psi(q_1) - R_z(2\pi/3)\cdot\psi(q_2) \\ \psi(q_1) - R_z(-2\pi/3)\cdot\psi(q_3) \end{bmatrix}, \tag{15}$$

the rotation matrix

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{16}$$

ψ is defined in (9), and $\lambda \in \mathbb{R}^6$ is a vector of Lagrange multipliers. Then the Lagrangian equations of motion for the robot are

$$\begin{aligned} \frac{d}{dt}\frac{\partial L_a}{\partial \dot{q}} - \frac{\partial L_a}{\partial q} &= \lambda^T H(q) + Bu && (17)\\ h(q) &= 0, && (18) \end{aligned}$$

where

$$H(q) = \frac{\partial h(q)}{\partial q}.$$

Defining $v=\dot{q}$, (17)-(18) can be written as a set of 24 first-order DAEs of Index 3 (Brenan et al., 1996; Kunkel and Mehrmann, 2006), in the variables $q \in \mathbb{R}^9$, $v \in \mathbb{R}^9$ and $\lambda \in \mathbb{R}^6$,

$$\begin{aligned} \dot{q} &= v && (19)\\ M(q)\dot{v} + C(q,v) + G(q) &= \lambda^T H(q) + Bu && (20)\\ h(q) &= 0, && (21) \end{aligned}$$

where

$$\begin{aligned} M(q) &= \mathrm{diag}\,(m(q_1), m(q_2), m(q_3)) \in \mathbb{R}^{9\times 9},\\ C(q,v) &= \mathrm{diag}(c(q_1,v_1), c(q_2,v_2), c(q_3,v_3)) \in \mathbb{R}^9,\\ G(q) &= \mathrm{diag}(g(q_1), g(q_2), g(q_3)) \in \mathbb{R}^9,\\ B &= \mathrm{diag}\,(b,b,b) \in \mathbb{R}^{9\times 3}. \end{aligned}$$

Equations (19) - (21) are a complete dynamic model of the delta robot, but index reduction is necessary for simulation and application of modern control theory.

3.3 Robot Hamiltonian Dynamics

For some applications such as port-Hamiltonian analysis (van der Schaft, 2013) it is useful to have a Hamiltonian model of the robot. This is derived in similar fashion by defining the momentum vector $p \in \mathbb{R}^9$ and the Hamiltonian $H = T + V$ for each arm, augmenting the constraint (15) by the Lagrange multiplier λ and solving the Hamiltonian equations, resulting in

$$\begin{aligned} M(q)\dot{q} &= p && (22)\\ \dot{p} &= \frac{1}{2} v \frac{\partial M(q)}{\partial q} v - G(q) + Bu + H^T(q)\lambda && (23)\\ h(q) &= 0, && (24) \end{aligned}$$

where the partial derivatives of M need to be computed symbolically. This formulation has about the same computational complexity as (19)-(21), results in similar numerical solutions using the same type of solver, but could be used with a symplectic solver for speedup.

3.4 Index Reduction

Following the same approach from Section 2, the constraint (21) is replaced with a linear combination of its first two derivatives with respect to time. Define

$$\begin{aligned} z_0 &= h(q) && (25)\\ z_1 &= \dot{z}_0 = \frac{\partial H(q)}{\partial q}\dot{q} && (26)\\ z_2 &= \dot{z}_1 = \dot{H}(q)\dot{q} + H(q)M^{-1}(q)\left(\lambda^T H(q)\right.\\ &\qquad \left. + Bu - C(q,\dot{q}) - G(q)\right), && (27) \end{aligned}$$

and replace (21) or (24) with

$$z_2 + \alpha_1 z_1 + \alpha_0 z_0 = 0, \tag{28}$$

where $s^2 + \alpha_1 s + \alpha_0$ is a Hurwitz polynomial (all roots in the open left-half plane). The model (19)-(20) and (28) or (22)-(23) and (28), is an index 1 DAE with 18 differential equations, 6 algebraic equations and 24 states q, v and λ, or q, p and λ, respectively.

It is interesting to express the dynamics in Zero Dynamics Normal Form, as we did for the pendulum. Following (Isidori, 1989), we define $\xi = [z_0\ z_1]^T \in \mathbb{R}^{12}$ to be the "linear" part. Then there exist coordinates $\eta \in \mathbb{R}^6$ which are functions of q, v and u (after algebraically eliminating λ) so that (19)-(20) and (28) can be written locally in Zero Dynamics Normal Form (Isidori, 1989),

$$\dot{\eta} = f(\eta, \xi, u) \tag{29a}$$

$$\dot{\xi} = A\xi, \tag{29b}$$

where the 12 eigenvalues of A are located at the roots of (28), and the 6-dimensional zero dynamics

$$\dot{\eta} = f(\eta, 0, u) \tag{30}$$

are the dynamics of the robot. In other words, there is a 6-dimensional manifold defined by $\xi = 0$ on which the robot dynamics exist and evolve according to (30). The ξ-dynamics are exponentially stable, are not controllable from u, and once they converge to 0, do not affect q or v. This means that if we linearize (19)-(20) and (28), we expect to see 12 poles and zeros at the roots to (28), and these dynamics are neither controllable nor observable. They are easily removed using a Hankel-norm model truncation. The resulting reduced-order model is six dimensional and equivalent to a linearization obtained otherwise.

In practice, expressions for z_1 and z_2 in (26)-(27) are computed automatically using the `der(·)` operator. Also, because the model is an index 1 DAE (instead of an index 0 ODE), it is not necessary to compute the inverse of the inertia matrix for either the Lagrangian or Hamiltonian formulations. Further, it is not necessary to compute η or f in (29a)-(29b). Deriving these expressions is done to understand the geometric structure and properties.

The primary disadvantages of Baumgarte's method are that 24 equations in 24 variables are produced, instead of the minimal six (although λ can be removed by algebraic manipulation, leaving 18 implicit first-order differential equations in 18 differential variables), and that numerical solutions to (19)-(20) and (28) will drift off the constraint manifold $h = 0$ when the system is in motion. However, for this application we find the drift to be small, is computable for monitoring purposes, and controllable in the sense that it is reduced by reducing the solver tolerance. Moreover, simulation times for (19)-(20) and (28) are an order-of-magnitude faster than the model that results from index reduction by the dummy derivative method, despite the fact that we require three times more equations and dynamic states, due to the simplicity of the equations.

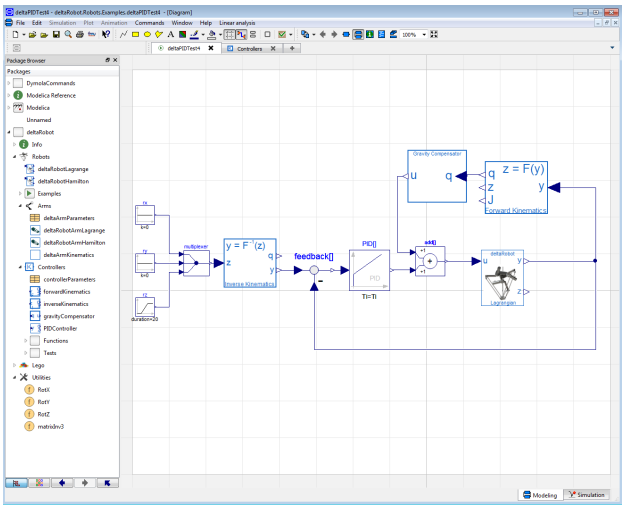

Figure 9. Screenshot of the Modelica deltaRobot library (left) and an a gravity-compensating PID feedback controller (right), showing the use of forward and inverse kinematics, gravity compensation and PID. The library contains models of the kinematics at the lowest level, arms, and robots at its highest level. We also have a package of controller components and a growing library of tasks, such as assembling Lego.

4 Modelica Library

We have created a Modelica library including models of the delta robot, various control algorithms that are derived from the model, and assembly tasks such as stacking blocks and assembling Lego bricks. A screen shot of the library is shown in Figure 9. For the delta robot models, the library is organized as a hierarchy, with partial models of the kinetics and parameters at the lowest level, extended into full models of the arms at the intermediate level, and models of the full robot at the highest level. We provide partial code listings of these components in the Appendix. At a higher level, multiple robots can be declared, and constraints among them defined in a manner analogous to what we have done for the arms. This allows for analysis of cooperative control using the same mathematics and approach. We remark that this is difficult using the Modelica standard library, because constraint forces acting on different parts of the end effector, for example, are difficult to introduce. The Lagrangian approach provides a natural way for additional constraint forces to be introduced, making this formulation more natural and effective when developing force and assembly control algorithms.

In the subsections that follow, we describe some of the control system blocks that we have constructed from the DAE model, each of which is realized as a functions using algorithm blocks.

4.1 Forward Kinematics

The forward kinematics function takes as input the three joint measurements at the servos and computes the other six joint angles (which are unactuated and unmeasured),

DOI 10.3384/ecp19157333

and the location of the end effector in world coordinates. The robot Jacobian is also computed. The forward kinematics are one-to-one but not onto, and defined implicitly by (15), which needs to be solved numerically. Specifically, partition q into measured and unmeasured states by defining $y = [q_{11}, q_{21}, q_{31}]^T$ to represent the measured joint angles, and $x = [q_{12}, q_{13}, q_{22}, q_{23}, q_{32}, q_{33}]^T$ to represent the unmeasured joint angles, and rearrange the variables of h so that (15) can be written

$$h(x,y) = 0. \tag{31}$$

This is solved for x using Newton's method

$$\frac{\partial h}{\partial x}(x_k,y) \cdot (x_{k+1} - x_k) = -h(x_k,y), \tag{32}$$

which typically converges to 7 decimal places of accuracy in 2-3 iterations since it can be initialized close to its solution in a real-time application. Each iteration requires the solution to a 6-dimensional set of linear equations. With the solution (x,y), the end effector location is computed using ψ in (9), and the robot Jacobian is also computed.

4.2 Inverse Kinematics

The inverse kinematics takes as input a location of the end effector $w \in \mathbb{R}^3$ and computes values for the joint angles $q \in \mathbb{R}^9$. This is not one-to-one: there is not a unique solution for all values of w. The inverse kinematics defined implicitly by the nine equations

$$\psi(q_i) - w = 0 \tag{33}$$

for $i = 1,2,3$. This is solved using Newton's method with some logic for choosing the desirable solutions. Each Newton iteration involves computing the solution to three 3-dimensional linear systems of equations, making the complexity less than the forward kinematics.

4.3 Gravity Compensation

One popular control scheme is to cancel the effect of gravity on the manipulator with an inner loop, and then close an outer feedback loop with a PD or PID compensator. The gravity compensating feedback is computed as the solution to the 9-dimensional set of linear equations

$$\begin{bmatrix} u \\ \lambda \end{bmatrix} \cdot [B\, H^T(q)] = G(q), \tag{34}$$

where in any real-time application q is computed via the forward kinematics from the joint measurements y. A closed-loop model including a delta robot, gravity compensation and using forward and inverse kinematics is shown in Figure 9.

4.4 Feedback Linearization

A feedback linearizing control law can be defined as follows. Let

$$w_1 = \psi(q_1) \tag{35}$$

denote the location of the end effector. Symbolically differentiate this twice

$$w_2 = \dot{w}_1 = d\psi(q_1)v_1 \tag{36}$$

$$\dot{w}_2 = \dot{d\psi}(q_1)v_1 + d\psi(q_1)\dot{v}_1. \tag{37}$$

Solving (20) for $\dot{v}$ and substituting the result into (37) gives

$$\dot{w}_2 = \alpha(q) + \beta(q) \cdot u$$

from which the control law

$$u = \frac{1}{\beta(q)}\left(-\alpha(q) - k_1 w_1 - k_2 w_2 + w_r\right)$$

renders the system linear from w_r to w_1. Expressions for α and β can be computed automatically. They require inversion of the 9×9 inertia matrix M, which is not difficult because it is block diagonal.

5 Linear Control Design and Analysis

The model (19)-(20), (28) and control functions described in the previous section, realized in the deltaRobot Modelica library, enable dynamic analysis and model-based design of new control algorithms for various tasks related to pick-and-place and robotic assembly. Here we show some results of an example dynamic analysis. We compute the linearization of the delta robot using values for parameters that are measured from a delta robot in our laboratory, at the equilibria $q_{i1} = 0\,\text{rad}$, meaning that the proximal links are all horizontal. A pole-zero plot is shown in Figure 10. First, notice that there are 12 pole-zero cancellations at $s = -5$ corresponding to the dynamics of (29a), as expected. These do not affect the input-output behavior and can be eliminated from the linear system by a Hankel norm truncation. Perhaps surprisingly, this configuration is open-loop unstable. Note that this configurations is well within the reachable workspace. (The unstable root crosses into the right-half plane at an angle of approximately $q_{i1} = 22^\circ$, for our robot.) This kind of instability is a common characteristic of robotic manipulators, and has important consequences. For example, stabilizing feedback gains have lower limits (Skogestad and Postlethwaite, 2005). In some applications such as fine force control, it is common practice to reduce feedback gains to maintain stability during contact. But the lower bound means that this practice has has limits, which are not obvious without a model-based analysis.

6 Elevator Cable Sway

Modeling elevator cable sway is another example where we have applied Baumgarte's method. The system is diagrammed in Figure 11. The traveling cable, which supplies power and signals to the car, is attached to the bottom of the car at one end, and the inside of the elevator shaft at the other. The cable experiences horizontal motion ("sway") when the car moves or when the building

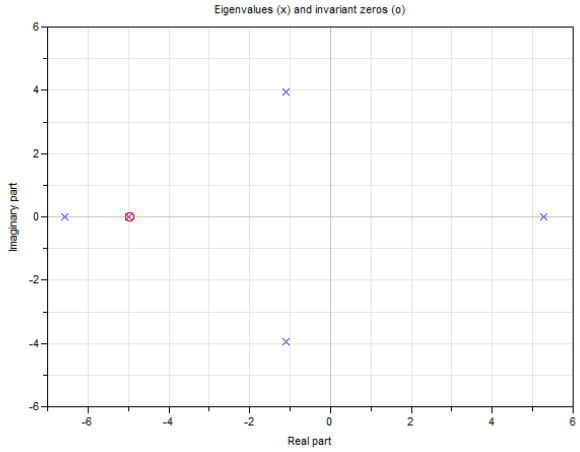

Figure 10. Pole-zero plot of the delta robot in equilibrium with $q_1 = 0$rad for the three proximal links. There are 12 pole-zero pairs at $s = -5$ corresponding to the dynamics of (28). The plot shows four poles at approximately $s = -2 \pm j$, one at $s = -6.5$, and an unstable pole at $s = 5.2$.

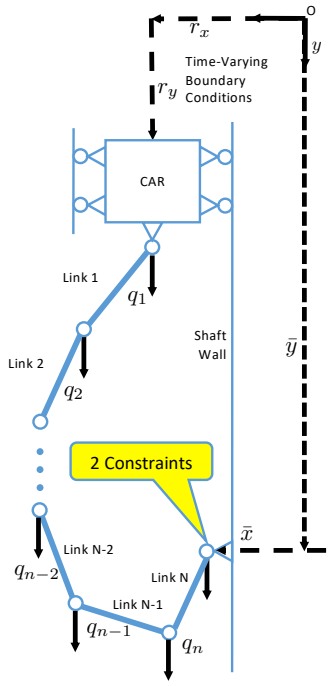

Figure 11. Elevator Cable Sway.

sways due to wind or earthquake. Because it can be damaged by striking the wall, we design a feedback controller attenuate the cable sway by moving the car.

The system can be modeled as a constrained chain of rigid links with springs and dampers between each link (Tomaszewski and Pieranski, 2005),

$$\dot{q} = v \tag{38a}$$

$$M(q)\dot{v} + C(q)v^2 + Dv + G(q) + Kq + a(q)\ddot{r}_x + b(q)\ddot{r}_y = \lambda H^T(q) \tag{38b}$$

$$h(q) = 0 \tag{38c}$$

where

$$h(q) = \begin{bmatrix} \bar{x} - \sum_{k=1}^{N} l\sin(q_k) & \bar{y} - \sum_{k=1}^{N} \cos(q_k) \end{bmatrix}^T, \tag{39}$$

$q \in \mathbb{R}^N$ is the vector of link angles, $v \in \mathbb{R}^N$ is the vector of angular velocities, M, C, D, K and G are the inertia, centripetal, damping and gravity matrices, respectively, $\ddot{r}_x$ and $\ddot{r}_y$ are the x and y acceleration of the frame marked "O," respectively, $\lambda \in \mathbb{R}^2$ is the Lagrange multiplier vector, $h = 0$ represents the constraint of the chain attached to the wall at locations $\bar{x}$ and $\bar{y}$, and N is the number of links, typically $N = 100$.

Equation (38) is a DAE of index 3, and we reduce the index exactly as we did earlier, replacing the constraint h with a linear combination of its first two derivatives. The resulting index-1 model is then used for simulation and feedback control design. The details are omitted for space reasons, and we present the results of one particular feedback controller which takes as input a single measurement of horizontal cable displacement, filters the measurement through a lead compensator which is designed using a frequency response computed from the model, and applies the output to the car motion controller. In Figure 12 we see the horizontal displacement of the car due to building sway that is caused by an earthquake, when the controller is off. This causes the cable to sway. In Figure 13, the feedback controller is engaged 50s after the earthquake begins, and moves the car up and down for a period of 300s, attenuating the cable sway by 75%.

We remark that a model of an *open* chain is elementary to construct from the Modelica Standard Library (MSL) and has been used for benchmarking (Casella, 2015). However, we have not been successful in modeling the *constrained* chain using the MSL, because the index reduction fails for large values of N. Even if it did compile, consistent initialization would be a challenge. On the other hand, using Baumgarte's method, we are able to compile models with $N > 200$ and can initialize the DAE using the procedure outlined in Section 2.

7 Conclusion

In this paper we show how Baumgarte's method of index reduction can be used in a Modelica model of a constrained mechanical systems. The method reduces the model index prior to compilation, so that the model does not undergo automatic index reduction by the compiler. Baumgarte's method has some advantages over the "dummy derivative" method that is integrated into Modelica compilers for some models. It may be easier to compute consistent initial conditions, the derived models can be used directly to derive model-based control algorithms, and simulations may run faster. On the other hand, the method does not enforce constraints exactly, and drift occurs during simulations. We find, however, that this drift is not consequential for our mechatronic applications, and in

DOI 10.3384/ecp19157333

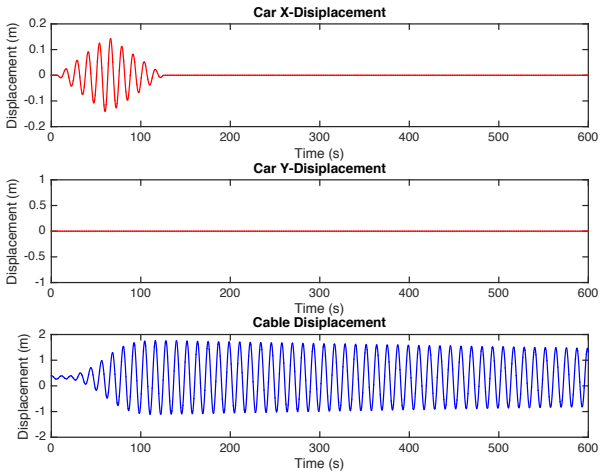

Figure 12. Car *x* (top) and *y* (middle) motion, and elevator cable sway (bottom) due to earthquake.

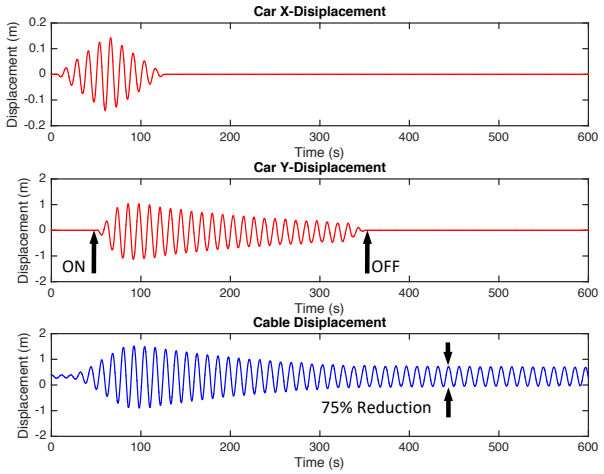

Figure 13. Car *x* (top) and *y* (middle) motion, under feedback control, and elevator cable sway (bottom) during earthquake.

fact the method allows for compilation and simulation of some models that otherwise cannot compile and initialize. We believe the method may find successful application in other domains, particularly for problems in which consistent initial conditions are difficult to compute.

A Delta Robot Modelica Model

The Delta robot arm kinematics are defined in the following partial Modelica model.

```
partial model deltaArmKinematics
 deltaArmParameters p;   // Parameters
 Real q[3],psi[3],dpsi[3,3];
equation
 psi[1]=p.L2*sin(q[2])*sin(q[3]);
 psi[2]=p.L3-p.L0+p.L1*cos(q[1])...
  +p.L2*cos(q[2]);
 psi[3]=p.L1*sin(q[1])+...
  p.L2*sin(q[2])*cos(q[3]);
 // Gradient of end effector location ...
 dpsi[1,1]=0.0;
 dpsi[1,2]=p.L2*cos(q[2])*sin(q[3]);
 dpsi[1,3]=p.L2*sin(q[2])*cos(q[3]);
 dpsi[2,1]=-p.L1*sin(q[1]);
 dpsi[2,2]=-p.L2*sin(q[2]);
 dpsi[2,3]=0.0;
 dpsi[3,1]=p.L1*cos(q[1]);
 dpsi[3,2]=p.L2*cos(q[2])*cos(q[3]);
 dpsi[3,3]=-p.L2*sin(q[2])*sin(q[3]);
end deltaArmKinematics;
```

Arm dynamics are defined extending the kinematics model. These expressions are computed in *Mathematica* and exported via scripts, automatically generating the Modelica code.

```
model deltaRobotArmLagrange
extends deltaArmKinematics;
Real v[3], tau[3];
protected
Real M[3,3],C[6,3],G[3];
equation
// Inertia Matrix...
m[1,1]=p.J1+p.LC1^2*M1+p.L1^2*(p.M2+p.M3);
m[1,2]=p.L1*(p.LC2*p.M2+p.L2*p.M3)...
 *(cos(q[1])*cos(q[2])*cos(q[3])...
 +sin(q[1])*sin(q[2]));
m[1,3]=-p.L1*(p.LC2*p.M2+p.L2*p.M3)...
 *cos(q[1])*sin(q[2])*sin(q[3]);
m[2,1]=m[1,2];
m[2,2]=p.J2+p.M2*p.LC2^2+p.M3*L2^2;
m[2,3]=0.0;
m[3,1]=m[1,3];
m[3,2]=0.0;
m[3,3]=(p.J2+p.M2*p.LC2^2+p.M3*p.L2^2)*sin(
    q[2])^2;
// Centripetal and Coriolis vectors...
c[1,1]=0.0;
c[1,2]=p.L1*(p.LC2*p.M2+p.L2*p.M3)...
 *(cos(q[1])*sin(q[2])-cos(q[2])*cos(q[3])*
     sin(q[1]));
c[1,3]=p.L1*(p.LC2*p.M2+p.L2*p.M3)...
 *sin(q[1])*sin(q[2])*sin(q[3]);
c[2,1]=p.L1*(p.LC2*p.M2+p.L2*p.M3)...
 *(cos(q[2])*sin(q[1])-cos(q[1])*cos(q[3])*
     sin(q[2]));
c[2,2]=0.0; c[2,3]=0.0;
c[3,1]=-(p.L1*(p.LC2*p.M2+p.L2*p.M3)...
 *cos(q[3])*cos(q[1])*sin(q[2]));
c[3,2]=-(p.J2+p.LC2^2*p.M2+p.L2^2*p.M3)....
 *cos(q[2])*sin(q[2]);
c[3,3]=0.0; c[4,1]=0.0;
c[4,2]=0.0; c[4,3]=0.0;
c[5,1]=-2.0*p.L1*(p.LC2*p.M2+p.L2*p.M3)...
 *cos(q[1])*cos(q[2])*sin(q[3]);
c[5,2]=0.0;
c[5,3]=(p.J2+p.LC2^2*p.M2+p.L2^2*p.M3)*sin
    (2*q[2]);
c[6,1]=0.0; c[6,2]=0.0; c[6,3]=0.0;
// Gravity vector...
G[1]=-p.g*(p.LC1*p.M1+p.L1*(p.M2+p.M3))...
 *cos(q[1]);
G[2]=-p.g*(p.LC2*p.M2+p.L2*p.M3)...
 *cos(q[2])*cos(q[3]);
G[3]= p.g*(p.LC2*p.M2+p.L2*p.M3)...
 *sin(q[2])*sin(q[3]);
// Arm Dynamics...
```

```
der(q) = v;
m*der(v)+c[1,:]*v[1]^2+c[2,:]*v[2]^2...
 +c[3,:]*v[3]^2+c[4,:]*v[1]*v[2]...
 +c[5,:]*v[2]*v[3]+c[6,:]*v[1]*v[3]...
 +G+p.DAMPING.*v = tau;
end deltaRobotArmLagrange;
```

Below is the Lagrangian robot model. The Hamiltonian version is similar. Note that the derivatives of h are computed automatically.

```
model deltaRobotLagrange
Arms.deltaRobotArmLagrange arm1,arm2,arm3;
Real lambda[6];
Real h0[6],h1[6],h2[6];
Input Real u[3];
parameter Real POLE=5.0;
constant Real Rot2[3,3] = Utilities.RotZ
    (2.0*PI/3.0);
constant Real Rot3[3,3] = Utilities.RotZ
    (-2.0*PI/3.0);
constant Real B[3] = {1, 0, 0};
equation
// tau = H^T(q) * lambda...
arm1.tau=transpose(arm1.dpsi)*lambda[1:3]
 +transpose(arm1.dpsi)*lambda[4:6]+B*u[1];
arm2.tau=-transpose(Rot2*arm2.dpsi)*...
 lambda[1:3]+B*u[2];
arm3.tau=-transpose(Rot3*arm3.dpsi)*...
 lambda[4:6]+B*u[3];
// Baumgarte's method of index reduction...
h0=cat(1,arm1.psi-Rot2*arm2.psi,...
 arm1.ps -Rot3*arm3.psi);
h1=der(h0);
h2=der(h1);
zeros(6)=h2+2.0*POLE*h1+POLE^2*h0;
end deltaRobotLagrange;
```

We remark that the index-3 model can be constructed by replacing the last line with

```
h0=zeros(6);
```

which will compile in Dymola using the "dummy derivative" method for index reduction. The result is two sets of DAEs with some switching logic.

References

Bernhard Bachmann. Mathematical aspects of object-oriented modeling and simulation. In *Proceedings of the 5th International Modelica Conference*, 2006.

Olivier A. Bauchau and André Laulusa. Review of contemporary approaches for constraint enforcement in multibody systems. *Journal of Computational and Nonlinear Dynamics*, 2007.

J. W. Baumgarte. Stabilization of constraints and integrals of motion in dynamic systems. *Computer Methods in Applied Mechanics and Engineering*, 1:1–16, 1972.

J. W. Baumgarte. A new method of stabilization for holonomic constraints. *ASME Journal of Applied Mechanics*, 50:869–870, 1983.

Scott A. Bortoff. Object-oriented modeling and control of delta robots. In *IEEE Conference on Control Technology and Applications*, pages 251–258, 2018.

K. E. Brenan, S. L. Cambell, and L. R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. SIAM, 1996.

J. Brinker, B. Corves, and M. Wahle. A comparative study of inverse dynamics based on clavel's delta robot. In *Proceedings of the 14th IFToMM World Congress*, Oct. 2015.

Francesco Casella. Simulation of large-scale models in modelica: State of the art and future perspectives. In *Proceedings of the 11th International Modelica Conference*, pages 459–468, 2015.

Francois E. Cellier. *Continuous System Simulation*. Springer, 2006.

Francois E. Cellier and Jurden Greifeneder. *Continuous System Modeling*. Springer, 1991.

R. Clavel. Device for the movement and positioning of an element in space. U.S. Patent 4, 976, 582, Dec. 11 1990.

Hilding Elmqvist, Toivo Henningsson, and Martin Otter. Innovations for future modelica. In *Proceedings of the 12th International Modelica Conference*, pages 693–702, 2017.

Peter Fritzon. *Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. Wiley, 2015.

Philippe Guglielmetti. *Model-Based Control of Fast Parallel Robots: A Global Approach in Operational Space*. PhD thesis, Ecole Polytechnique Federale de Lausanne, 1994.

Alberto Isidori. *Nonlinear Control Systems*. Springer-Verlag, 1989.

Peter Kunkel and Volker Mehrmann. *Differential-Algebraic Equations: Analysis and Numerical Solution*. European Mathematical Society, 2006.

Sven Erik Mattsson and Gustaf Söderlind. Index reduction in differential algebraic equations using dummy derivatives. *SIAM Journal on Scientific Computing*, 14(3), 1993.

Jean-Pierre Merlet and Clement Gosselin. *Springer Handbook of Robotics*, chapter Parallel Mechanisms and Robots. Springer, 2008.

Sigurd Skogestad and Ian Postlethwaite. *Multivariable Feedback Control: Analysis and Design*. Wiley, 2005.

M. M. Spong and M. Vidyasagar. *Robot Dynamics and Control*. Wiley, 2004.

Staicu St. and Carp-Ciocardia D. C. Dynamic analysis of clavel's delta parallel robot. In *Proceedings of the 2003 International Conference on Robotics and Automation*, pages 4116–4121, 2003.

Waldemar Tomaszewski and Piotr Pieranski. Dynamics of ropes and chains: 1. the fall of the folded chain. *New Journal of Physics*, 7(45), 2005.

A.J. van der Schaft. *Surveys in Differential-Algebraic Equations I*, chapter Port-Hamiltonian Differential-Algebraic Systems, pages 173–226. Springer, 2013.

DOI 10.3384/ecp19157333