Properties of Differential-Algebraic Equation Systems (DAE)

Francesco Casella

Dipartimento di Elettronica e Informazione

Politecnico di Milano

Introduction

Object-oriented system models can be reduced to (large) systems of differential-algebraic equations, resulting from the aggregation of the component equations and of the connection equations.

Some fundamental definitions about DAE's will now be reviewed, because their understanding is essential to fully grasp object-oriented simulation tools (such as Dymola) actually convert Modelica models into simulation code:

- ODE's and DAE's
- Index of DAE systems
- Index reduction in DAE systems

The fundamental mathematical definitions will be presented together with typical engineering modelling examples, in order to make their understanding easier.

Basic definitions

Explicit ordinary differential equations (ODE)

$$\dot{x} = f(x, u, p, t)$$

State-space models (ODE + output equations)

$$\dot{x} = f(x, u, p, t)$$
$$y = g(x, u, p, t)$$

x : Dynamic variables
y : Algebraic variables
u : Exogenous input variables
p : Constant parameters
t : Time

Semi-explicit Differential-Algebraic Equations (DAE)

$$\dot{x} = f(x, y, u, p, t)$$
$$0 = g(x, y, u, p, t)$$

Implicit Differential-Algebraic Equations

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

In the following slides, we will drop the p, u, and t terms from the equations for the sake of simplicity, as their values are known, and just focus on the unknowns

Integration of ODE and state-space systems

There is a large number of integration algorithms (and software packages) to numerically integrate ODE systems

- Euler's algorithm (very simple, poor accuracy)
- One-step methods: (Runge-Kutta, RADAU, ...)
- Multi-step algorithms (Adams-Bashfort, Adams-Moulton, ...)

$$\dot{x} = f(x)$$

They can iteratively compute the value of the state x at the next simulation step given:

- the current state (and the result of the past simulation steps for multi-step methods)
- a software routine to compute the $f()$ function (and its Jacobian matrix, for implicit algorithms)

In the case of state-space systems, once the next value of the state has been computed, it is straightforward to compute the next value of the system outputs by plugging in the value of the state in the ouput equations

$$y = g(x)$$

For those who are familiar with the SIMULINK simulation environment, this is basically what happens when a block diagram is simulated: if there are no algebraic loops, it is possible to introduce an suitable ordering in the blocks, and solve them one at a time, using already available quantities at the inputs.

Integration of DAE Systems

Consider a semi-explicit DAE system:

$$\dot{x} = f(x, y)$$
$$0 = g(x, y)$$

If the Jacobian of the algebraic equations is non-singular, then by the implicit function theorem it is possible to (locally) invert g and compute y as a function of x.

$$\left|\frac{\partial g}{\partial y}\right| \neq 0 \qquad \Rightarrow \qquad y = h(x)$$

Given the current state (and, possibly, the result of past simulation steps), numerical algorithms such as DASSL can efficiently compute the values of x and y at the next simulation step.

In the case of implicit DAE system $F(x, \dot{x}, y) = 0$ defining $G(x, z) = 0, \quad z = \begin{bmatrix} \dot{x} \\ y \end{bmatrix}$

the solvability condition becomes: $\left|\frac{\partial G}{\partial z}\right| \neq 0$

If the Jacobians are non-singular, these DAE's are (at least locally) equivalent to state-space systems. You can think about them as “implicit state-space systems”

High-Index DAE: a first example

If the Jacobian is singular, then the DAE cannot be solved as such. Their structure is inherently different from the implicit state-space systems we've just seen

The most common cause of this problem is that some of the algebraic equations define **algebraic constraints** between the **dynamic variables x**, rather than a relationship between the dynamic variables x and the algebraic variables y

Example:

$$\begin{aligned} \dot{x}_1 &= 1 - y_1 \\ \dot{x}_2 &= -x_2 + y_1 \\ 0 &= x_1 - x_2 \end{aligned}$$

Three equations in the three unknowns $\dot{x}_1,\ \dot{x}_2,\ y_1$

$$\left[\frac{\partial g}{\partial y}\right] = [0]$$

The Jacobian is singular: it is not possible to solve the algebraic equations in the form $y = h(x)$ in order to determine y

The only way to solve the system is to differentiate the algebraic equations and add them to the set of equations. In this way the system might be solved

High-Index DAE: a first example

Original system:

$$\begin{aligned}\dot{x}_1 &= 1 - y_1\\ \dot{x}_2 &= -x_2 + y_1\\ 0 &= x_1 - x_2\end{aligned}$$

Augmented system (original system + differentiated algebraic equations)

$$\begin{aligned}\dot{x}_1 &= 1 - y_1\\ \dot{x}_2 &= -x_2 + y_1\\ 0 &= x_1 - x_2\\ 0 &= \dot{x}_1 - \dot{x}_2\end{aligned} \longrightarrow 0 = 1 - y_1 + x_2 - y_1 \longrightarrow y_1 = (1 + x_2)/2 \nearrow \begin{aligned}\dot{x}_1 &= (1 - x_1)/2\\ \dot{x}_2 &= (1 - x_1)/2\\ x_2 &= x_1\\ \dot{x}_1 &= \dot{x}_2\end{aligned}$$

Important remarks:

- in some cases, more than one differentiation could be required to solve the system
- the augmented system of equation is redundant: it is not really necessary to solve the second differential equation, as it is already known that $x_1 = x_2$
- the initial conditions for the dynamic variables cannot be chosen arbitrarily: $x_1(0) = x_2(0)$

Definition of Index in Semi-Explicit DAE's

Consider a semi-explicit DAE system:

$$\dot{x} = f(x, y, t)$$
$$0 = g(x, y, t)$$

If there are no algebraic equations (ODE system) then the index is zero. Otherwise, differentiate the algebraic equations to obtain:

$$\dot{x} = f(x, y, t)$$
$$g_x(x, y, t)\dot{x} + g_y(x, y, t)\dot{y} + g_t(x, y, t) = 0$$

If the Jacobian g_x is square and nonsingular, then it is possible to solve for $\dot{y}$ and the index is one.

$$\dot{x} = f(x, y)$$
$$\dot{y} = -g_y(x, y, t)^{-1}\left[g_x(x, y, t)\dot{x} + g_t(x, y, t)\right] = 0$$

Otherwise, re-cast the problem as a semi-explicit DAE and iterate the procedure on the algebraic equations until it is possible to solve for $\dot{y}$

The minimum number of times it is necessary to differentiate the DAE system (in part or as a whole) in order to compute $\dot{y}$ is the index of the system

Note that the last differentiation is actually not needed in order to numerically solve the system, as it is just necessary to compute $\dot{x}, \ y$

Higher-Index models in O-O modelling

DAE systems with index > 1 are known as *higher*-index problems.

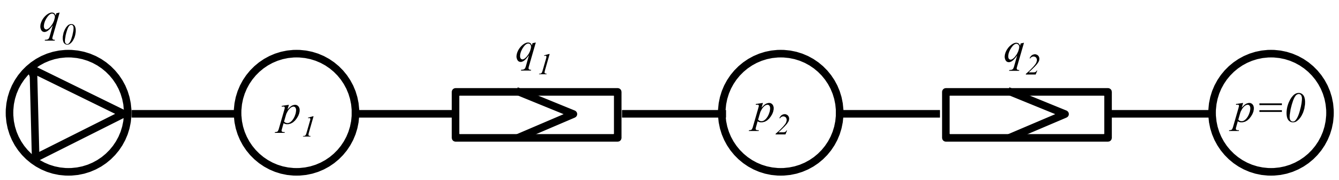
In some context, higher-index models are considered as ill-conditioned, or as results of bad modelling practice, but in fact this is definitely not the case, as we will see in several simple representative cases:

- Hydraulic network
- Lumped-parameter thermal system
- 1-D mechanical system
- CSTR system
- Idealized control system

The problem with higher-index DAE's is that they require either symbolic manipulation or problem-specific solvers to be solved correctly. General-purpose DAE solvers (such as DASSL) cannot solve them reliably: the integration error does not converge to zero as the step-lenght is reduced to zero.

Example 1: High-Index Hydraulic Network Model - I

Consider the model of a simple hydraulic network, comprising an ideal volumetric pump, two volumes, and two linear pressure losses (laminar flow). The model is built by aggregation of the single component models.

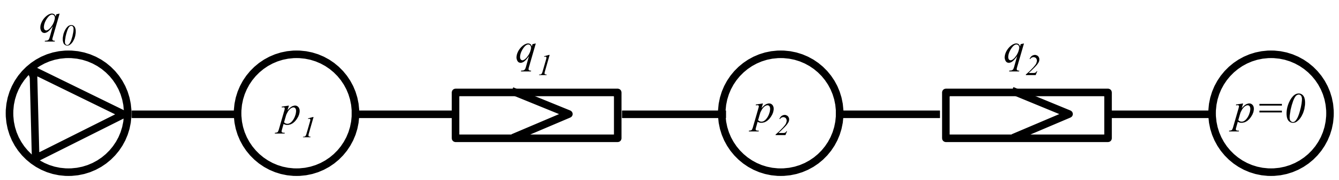


After trivial substitutions and variable elimination, the equations of the system can be reduced to the following system. If all the capacitances and resistances are greater than zero, the system is an index 1 DAE which can easily be solved.

$$
\begin{aligned}
C_1 \dot{p}_1 &= (q_0 - q_1) \\
C_2 \dot{p}_2 &= (q_1 - q_2) \\
q_0 &= f(t) \\
p_1 - p_2 &= R_1 q_1 \\
p_2 &= R_2 q_2
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
\dot{p}_1 &= \frac{1}{C_1}\left[f(t) - \frac{p_1 - p_2}{R_1} \right] \\
\dot{p}_2 &= \frac{1}{C_2}\left[\frac{p_1 - p_2}{R_1} - \frac{p_2}{R_2} \right] \\
q_0 &= f(t) \\
q_1 &= \frac{p_1 - p_2}{R_1} \\
q_2 &= \frac{p_2}{R_2}
\end{aligned}
$$

Example 1: High-Index Hydraulic Network Model - II

Suppose that $R_1 << R_2$. The system will be *stiff*, i.e. there will be two different dynamics with widely different time constants: the two pressures will tend to become almost equal in a very short time (say, a few milliseconds), and then will move together with a much slower dynamics (say, a few seconds).



Now, if we're not interested in the fast dynamics (because q_0 does not excite it) and in the very small pressure difference between the two volumes, we'd like to substitute the model of the first pressure loss with a trivial one, i.e. $P_1 = P_2$.

The resulting system has a singular Jacobian, and it is thus of higher index.

$$
\begin{aligned}
&C_1 \dot{p}_1 = (q_0 - q_1) \\
&C_2 \dot{p}_2 = (q_1 - q_2) \\
&q_0 - f(t) = 0 \\
&p_1 - p_2 = 0 \\
&p_2 - p_3 - R_2 q_2 = 0
\end{aligned}
\qquad
\left[\frac{\partial g}{\partial y}\right] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & R_2 \end{bmatrix}
$$

Is this the end?

Example 1: High-Index Hydraulic Network Model - III

Of course not! To make the system solvable, it is possible to differentiate the algebraic constraint equation and add it to the set of equations, that can thus be solved:

$$
\begin{aligned}
&C_1\,\dot{p}_1 = (q_0 - q_1)\\
&C_2\,\dot{p}_2 = (q_1 - q_2)\\
&q_0 - f(t) = 0\\
&p_1 - p_2 = 0\\
&p_2 - R_2 q_2 = 0\\
&\dot{p}_1 - \dot{p}_2 = 0
\end{aligned}
$$

$$
\begin{aligned}
&\dot{p}_1 = \frac{f(t) - p_2/R_2}{C_1 + C_2}\\
&\dot{p}_2 = \frac{f(t) - p_2/R_2}{C_1 + C_2}\\
&q_0 = f(t)\\
&q_1 = \frac{C_2 q_0 + C_1 q_2}{C_1 + C_2}\\
&q_2 = \frac{p_2}{R_2}\\
&p_1 = p_2
\end{aligned}
$$

There are still two problems left:

- The resulting system has 6 equations for the 5 unknowns $\dot{p}_1, \dot{p}_2, q_0, q_1, q_2$
- The initial conditions for its "states" p_1, p_2 cannot be assigned arbitrary values, because of the constraint equations.

Some additional manipulation is thus required.

Dummy Derivative Method

Whenever an algebraic equation is differentiated, one of the resulting derivatives is substituted everywhere with a new algebraic variable (the dummy derivative). In this way, a reduced-order system is obtained with the same number of equations and variables.

In the case of the hydraulic network, the pressure p_2 is downgraded to the rank of algebraic variable, and a first-order index-1 DAE system is obtained, which can be readily transformed into state-space form.

$$
\begin{aligned}
&C_1 \dot{p}_1 = (q_0 - q_1)\\
&C_2 p_{2\mathrm{der}} = (q_1 - q_2)\\
&q_0 - f(t) = 0\\
&p_1 - p_2 = 0\\
&p_2 - R_2 q_2 = 0\\
&\dot{p}_1 - p_{2\mathrm{der}} = 0
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
&\dot{p}_1 = \frac{f(t) - p_2/R_2}{C_1 + C_2}\\
&p_2 = p_1\\
&p_{2\mathrm{der}} = \frac{f(t) - p_2/R_2}{C_1 + C_2}\\
&q_0 = f(t)\\
&q_1 = \frac{C_2 q_0 + C_1 q_2}{C_1 + C_2}\\
&q_2 = \frac{p_2}{R_2}
\end{aligned}
$$

In this way, we've actually obtained a **first order system**, only showing the slow dynamics we're actually interested into. The system can be simulated more easily, and used e.g. for control system design, where low-order models are usually sought.

Index Reduction Algorithms

The hydraulic network example shows what can be done with high-index models by manually manipulating the equations.

Modern simulation environments contain algorithm for symbolic manipulation that are able to carry out this task automatically for system of arbitrary complexity.

The first algorithm to be used is Pantelides' algorithm, which is able to determine which equations must be differentiated in order to make the higher-index problem solvable, by reducing it to index one.

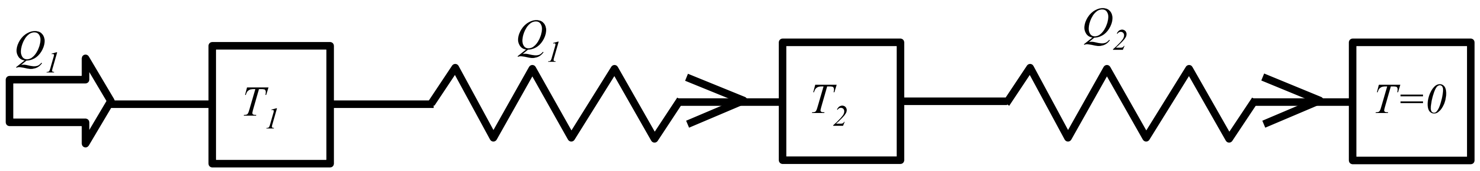
The second algorithm is the Dummy Derivative algorithm, which makes a clever selection of the dummy derivatives in order to obtain a non-singular reduced-order system.

In some cases (e.g. multibody mechanical systems) it is impossible to find a unique selection of dummy derivatives which is always non-singular – for these cases, the DD algorithm includes strategies to select different sets of state variables, which will be dynamically switched during the simulation (state variable pivoting). This is usually not required for thermo-hydraulic systems such as those found in energy conversion systems.

We'll get back to these algorithms later in this course.

Example 2: Lumped Parameter Thermal System

Consider the system represented in this diagram, composed of a heat source, two heat capacitances and two heat resistances:

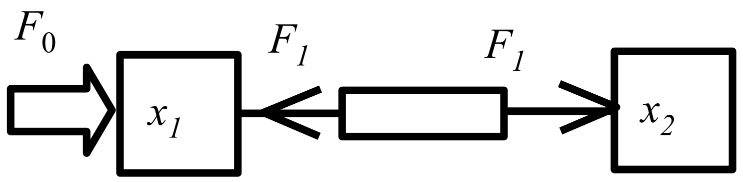


$$
\begin{aligned}
&C_1 \dot{T}_1 = (Q_0 - Q_1)\\
&C_2 \dot{T}_2 = (Q_1 - Q_2)\\
&Q_0 = f(t)\\
&T_1 - T_2 = R_1 Q_1\\
&T_2 = R_2 Q_2
\end{aligned}
$$

By analogy, it is obvious that the structure of the mathematical model is exactly the same as the previous case. Thus, if the first resistance is much small than the second, one could use $T_1 = T_2$ as a model for the left-hand-side thermal conduction component, and get a reduced-order system model.

Example 3: Mechanical System - I

Consider the 1-D mechanical system represented in this diagram, composed of a force source and two rigidly connected masses:



After trivial equation elimination, the equations describing the system are:

$$
\begin{aligned}
\dot{x}_1 &= x_3 \\
\dot{x}_2 &= x_4 \\
\dot{x}_3 &= (F_0 - F_1)/M_1 \\
\dot{x}_4 &= F_1/M_2 \\
F_0 &= f(t) \\
x_2 &= x_1 + L
\end{aligned}
$$

The Jacobian of the system is singular (F_1 cannot be computed from x). We differentiate the constraint equation and add it to the system

Example 3: Mechanical System - II

$$\begin{array}{l} \dot{x}_1 = x_3 \\ \dot{x}_2 = x_4 \\ \dot{x}_3 = (F_0 - F_1)/M_1 \\ \dot{x}_4 = F_1/M_2 \\ F_0 = f(t) \\ \dot{x}_2 = \dot{x}_1 \end{array} \quad \Rightarrow \quad \begin{array}{l} \dot{x}_1 = x_3 \\ \dot{x}_2 = x_4 \\ \dot{x}_3 = (F_0 - F_1)/M_1 \\ \dot{x}_4 = F_1/M_2 \\ F_0 = f(t) \\ x_3 = x_4 \end{array}$$

This is still not enough for F_1 to show up in the algebraic equations, so we differentiate them one more time:

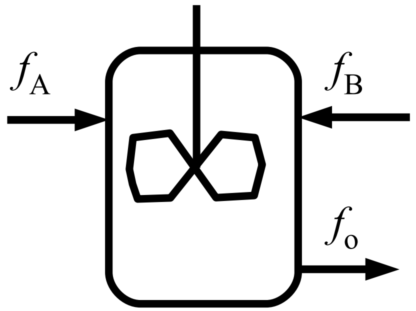
$$\begin{array}{l} \dot{x}_1 = x_3 \\ \dot{x}_2 = x_4 \\ \dot{x}_3 = (F_0 - F_1)/M_1 \\ \dot{x}_4 = F_1/M_2 \\ F_0 = f(t) \\ \dot{x}_4 = \dot{x}_3 \end{array} \quad \Rightarrow \quad \begin{array}{l} \dot{x}_1 = x_3 \\ \dot{x}_2 = x_4 \\ \dot{x}_3 = F_0/(M_1 + M_2) \\ \dot{x}_4 = F_0/(M_1 + M_2) \\ F_0 = f(t) \\ F_1 = F_0 M_2/(M_1 + M_2) \end{array} \quad \Rightarrow \quad \begin{array}{l} \dot{x}_1 = x_3 \\ \dot{x}_3 = F_0/(M_1 + M_2) \\ x_{2\mathrm{der}} = x_4 \\ x_{4\mathrm{der}} = F_0/(M_1 + M_2) \\ F_0 = f(t) \\ F_1 = F_0 M_2/(M_1 + M_2) \end{array}$$

The original system is therefore index 3. After applying the dummy derivatives algorithm, the system is reduced to a second-order index 1 problem.

Example 4: CSTR

Consider a simple continuosly-stirred tank reactor model, where the reaction A + B ↔ C takes place.

We can write the mass balances (in moles) for the three species:



$$\dot{n}_a = f_A - \frac{n_A}{n_A + n_B + n_C} f_o - r$$

$$\dot{n}_b = f_B - \frac{n_B}{n_A + n_B + n_C} f_o - r$$

$$\dot{n}_c = r - \frac{n_C}{n_A + n_B + n_C} f_o$$

If the reaction rate is specified by the reaction kinetics, then we get an index 1 DAE

$$r = f(nA, nB, nC)$$

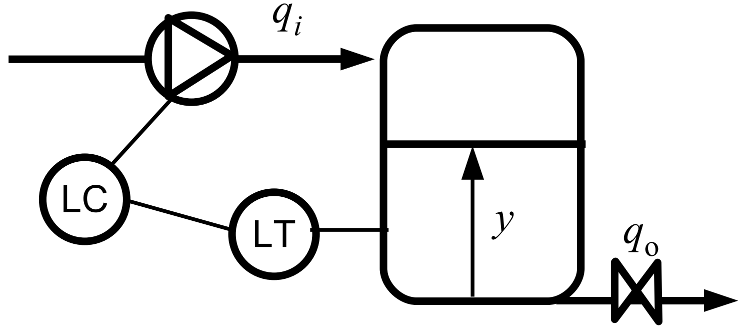
If instead we state that the reaction is at chemical equilibrium, we get an index 2 DAE

$$g(nA, nB, nC) = 0$$

In the latter case, it is possible to compute r by differentiating the equilibrium equation; by using the dummy derivative method, the sytem can be reduced to a second-order, index 1 DAE.

Example 5: Ideal Controller

Consider this simple system, composed by a tank and a level control system:



$$\dot{y}=(q_i-q_o)/A$$
$$q_o=K\sqrt{y}$$
Tank

$$\dot{x}=e$$
$$e=y°-y$$
$$q_i=K_p e+K_I x$$
PI controller

$$y°=y$$
Ideal controller

If we combine the equation of the tank and of the PI controller, we obtain a second-order, index-1 DAE. The controller parameters must be properly tuned.

If we know in advance that the controller performance will be very good, we might use the ideal controller equation instead. In this case we obtain an index 2 DAE. By applying the dummy derivative method, this is reduced to a purely algebraic system. In this case, the derivative of the set point $y°$ is required – therefore the set point must be differentiable.

$$\begin{aligned}\dot{y}&=(q_i-q_o)/A\\ q_o&=K\sqrt{y}\\ y&=y°\end{aligned} \;\Rightarrow\; \begin{aligned}\dot{y}&=(q_i-q_o)/A\\ q_o&=K\sqrt{y}\\ y&=y°\\ \dot{y}&=\dot{y}°\end{aligned} \;\Rightarrow\; \begin{aligned}y_{der}&=(q_i-q_o)/A\\ q_o&=K\sqrt{y}\\ y&=y°\\ y_{der}&=\dot{y}°\end{aligned} \;\Rightarrow\; \begin{aligned}q_i&=K\sqrt{y°}+A\dot{y}°\\ q_o&=K\sqrt{y°}\\ y&=y°\\ y_{der}&=\dot{y}°\end{aligned}$$