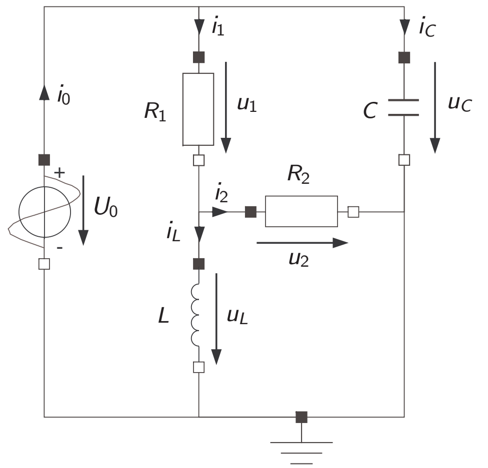
Numerical Simulation of Dynamic Systems XVI

Prof. Dr. François E. Cellier
Department of Computer Science
ETH Zurich

April 23, 2013

Structural Singularities

Unfortunately, the approaches proposed in the previous two presentations still don't always work:

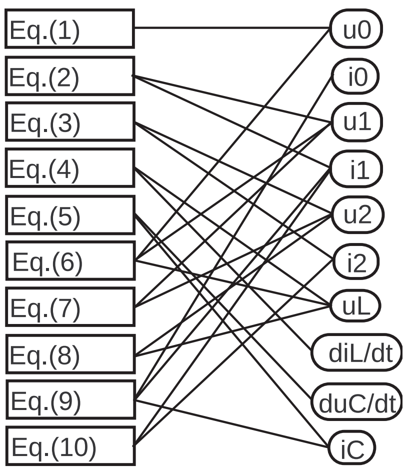


$$
\begin{array}{rlcl}
1: & u_0 & = & f(t) \\
2: & u_1 & = & R_1 \cdot i_1 \\
3: & u_2 & = & R_2 \cdot i_2 \\
4: & u_L & = & L \cdot \frac{di_L}{dt} \\
5: & i_C & = & C \cdot \frac{du_C}{dt} \\
 & & & \\
6: & u_0 & = & u_1 + u_L \\
7: & u_C & = & u_1 + u_2 \\
8: & u_L & = & u_2 \\
 & & & \\
9: & i_0 & = & i_1 + i_C \\
10: & i_1 & = & i_2 + i_L
\end{array}
$$

$\Rightarrow$ **We got again 10 implicitly formulated DAEs in 10 unknowns.**

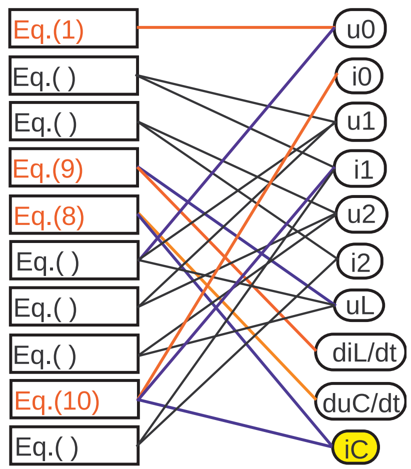
Structural Singularities II

Let us try the same approach. The structure digraph of the DAE system can be drawn as follows:



Structural Singularities III

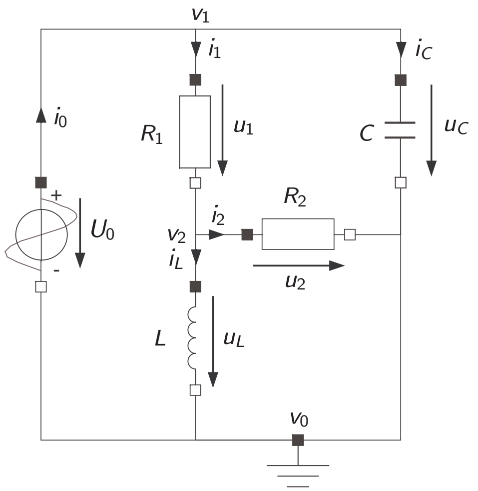
After a few steps of causalization:



- After four causalization steps, we got into troubles.
- The two connections attached to variable i_C have meanwhile both been colored in blue.
- Hence we are left without any equation to compute i_C.
- **The DAE system contains a structural singularity.**

Structural Singularities IV

Let us try another approach. We introduce the *node potentials* as additional variables:



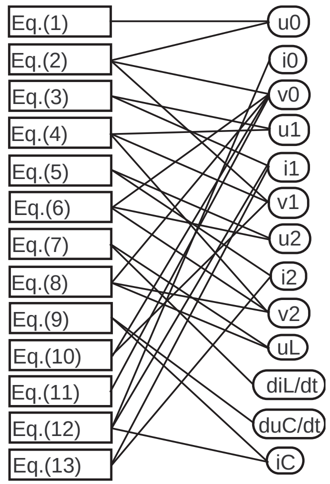
1: $u_0 = f(t)$
2: $u_0 = v_1 - v_0$
3: $u_1 = R_1 \cdot i_1$
4: $u_1 = v_1 - v_2$
5: $u_2 = R_2 \cdot i_2$
6: $u_2 = v_2 - v_0$
7: $u_L = L \cdot \frac{di_L}{dt}$
8: $u_L = v_2 - v_0$
9: $i_C = C \cdot \frac{du_C}{dt}$
10: $u_C = v_1 - v_0$
11: $v_0 = 0$

12: $i_0 = i_1 + i_C$
13: $i_1 = i_2 + i_L$

⇒ **We now got 13 implicitly formulated DAEs in 13 unknowns.**

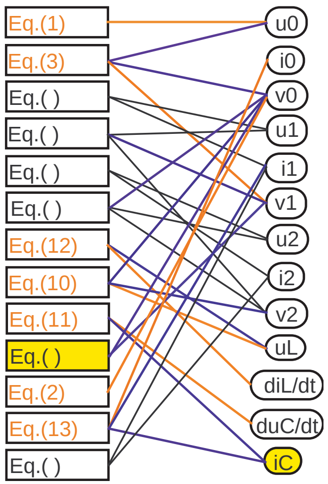
Structural Singularities V

The structure digraph of the DAE system can be drawn as follows:



Structural Singularities VI

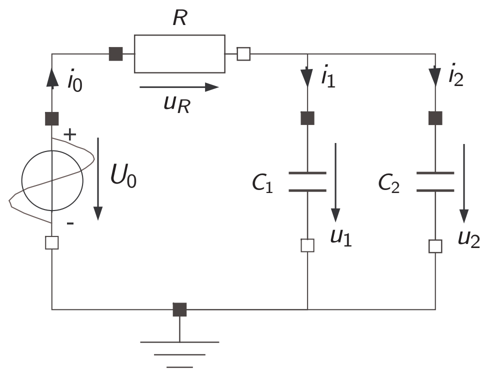
After a few steps of causalization:



- This time around, we were able to causalize seven equations before getting into troubles.
- Once again, the two connections attached to variable i_C have meanwhile both been colored in blue.
- Hence we are left without any equation to compute i_C.
- However, we seem to have made the problem worse, in that we now also have an equation, the former Eq.(10), that has its two attached connections colored in blue.
- Hence Eq.(10) has now become redundant, and we won't be able to use it at all.

Structural Singularity Elimination

Before we deal with the above circuit, let us choose a much simpler circuit that exhibits the same problems.



1: $u_0 = f(t)$
2: $u_R = R \cdot i_0$
3: $i_1 = C_1 \cdot \frac{du_1}{dt}$
4: $i_2 = C_2 \cdot \frac{du_2}{dt}$

5: $u_0 = u_R + u_1$
6: $u_2 = u_1$

7: $i_0 = i_1 + i_2$

⇒ **We now got 7 implicitly formulated DAEs in 7 unknowns.**

Structural Singularity Elimination II

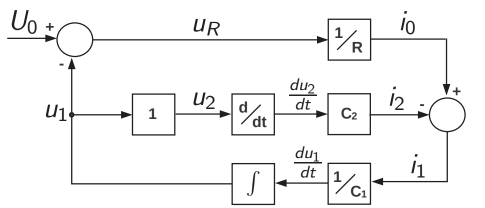
- If we choose u_1 and u_2 as state variables, then both u_1 and u_2 are considered known variables, and Eq.(6) has no unknown left. Thus, that equation must be considered a *constraint equation*.
- We can turn the causality around on one of the capacitive equations, solving e.g. for the variable i_2, instead of $\frac{du_2}{dt}$. Consequently, the solver has to solve for $\frac{du_2}{dt}$ instead of u_2, thus the *integrator* has been turned into a *differentiator*.
- In the model equations, u_2 must now be considered an unknown, whereas $\frac{du_2}{dt}$ is considered a known variable.

Structural Singularity Elimination III

The equations can now easily be brought into causal form:

$$u_0 = f(t) \qquad i_0 = \frac{1}{R} \cdot u_R$$
$$i_2 = C_2 \cdot \frac{du_2}{dt} \qquad i_1 = i_0 - i_2$$
$$u_2 = u_1 \qquad \frac{du_1}{dt} = \frac{1}{C_1} \cdot i_1$$
$$u_R = u_0 - u_1$$

with the block diagram:



Structural Singularity Elimination IV

- Numerical differentiation is a bad idea if explicit formulae are being used. These algorithms are highly unstable.
- Using implicit formulae, numerical integration and differentiation are essentially the same, but implicit formulae call for an iteration at every step.
- Pantelides proposed a different approach. He noted that, if:

$$u_2(t) = u_1(t), \forall t$$

it follows that:

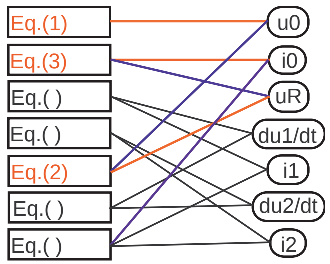
$$\frac{du_2(t)}{dt} = \frac{du_1(t)}{dt}, \forall t$$

Structural Singularity Elimination V

Thus, we can symbolically differentiate the constraint equation, and replace the constraint equation by its derivative:

$$u_0 = f(t)$$
$$u_R = R \cdot i_0$$
$$i_1 = C_1 \cdot \frac{du_1}{dt}$$
$$i_2 = C_2 \cdot \frac{du_2}{dt}$$
$$u_0 = u_R + u_1$$
$$\frac{du_2}{dt} = \frac{du_1}{dt}$$
$$i_0 = i_1 + i_2$$

with the partially causalized structure digraph:



- **The constraint equation has indeed disappeared. After partial causalization of the equations, we are now faced with an algebraic loop in four equations and four unknowns, a situation that we already know how to deal with.**

Structural Singularity Elimination VI

This approach works, but has a disadvantage.

- We again have two integrators in the model that we can seemingly integrate separately and independently of each other.
- Yet, this is an illusion. The constraint on the capacitive voltages has not disappeared. It has only been hidden.
- It is true that we can now numerically integrate $\frac{du_1}{dt}$ into u_1, and $\frac{du_2}{dt}$ into u_2. However, we must still satisfy the original constraint equation when choosing the initial conditions for the two integrators.
- The second integrator does not represent a true state variable. In fact, it is wasteful. We don't need two integrators, since the system has only one *degree of freedom*, i.e., one energy storage.

Structural Singularity Elimination VII

Let us modify the approach. Rather than replacing the constraint equation by its derivative, we shall augment the set of equations by the differentiated constraint equation:

$$u_0 = f(t)$$
$$u_R = R \cdot i_0$$
$$i_1 = C_1 \cdot \frac{du_1}{dt}$$
$$i_2 = C_2 \cdot \frac{du_2}{dt}$$
$$u_0 = u_R + u_1$$
$$u_2 = u_1$$
$$\frac{du_2}{dt} = \frac{du_1}{dt}$$
$$i_0 = i_1 + i_2$$

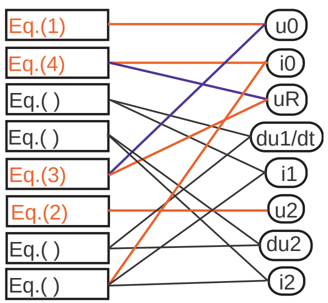
- We now have one equation too many. We need to throw another equation away.
- We throw one of the integrators away, e.g. the one that computes u_2 out of $\frac{du_2}{dt}$.
- Now, both u_2 and $\frac{du_2}{dt}$ are considered unknowns, and we have eight model equations in eight unknowns.

Structural Singularity Elimination VIII

We shall replace $\frac{du_2}{dt}$ by du_2 to symbolize that this is now an algebraic variable:

$$u_0 = f(t)$$
$$u_R = R \cdot i_0$$
$$i_1 = C_1 \cdot \frac{du_1}{dt}$$
$$i_2 = C_2 \cdot du_2$$
$$u_0 = u_R + u_1$$
$$u_2 = u_1$$
$$du_2 = \frac{du_1}{dt}$$
$$i_0 = i_1 + i_2$$

with the partially causalized structure digraph:



- **We are again faced with an algebraic loop in four equations and four unknowns.**

Structural Singularity Elimination IX

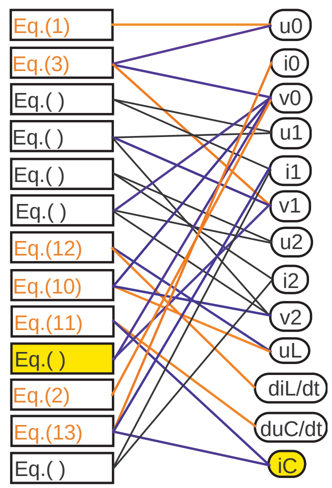
- In the mathematical literature, *structurally singular* systems are called *higher-index problems*, or more precisely, structurally singular physical systems lead to mathematical descriptions that present themselves in the form of higher-index DAEs.
- The *perturbation index* is a measure of the constraints among equations.
- An *index-0 DAE* contains neither algebraic loops nor structural singularities.
- An *index-1 DAE* contains algebraic loops, but no structural singularities.
- A DAE with a *perturbation index* > 1, a so-called *higher-index DAE*, contains structural singularities.

Structural Singularity Elimination X

- The algorithm by Pantelides is a *symbolic index reduction* algorithm.
- Each application of the algorithm reduces the perturbation index by one. Hence it may be necessary to apply the Pantelides algorithm more than once.
- For example, a mechanical system with constraints among positions or angles, such as a motor with a load, whereby the motor and the load are described separately by differential equations, leads to an index-3 DAE system.
- By applying the Pantelides algorithm once, the original constraint between positions gets reduced to a constraint between velocities or angular velocities, which are still state variables.
- By applying the Pantelides algorithm a second time, the constraint involving velocities gets reduced to a constraint between accelerations or angular accelerations, which are no longer outputs of integrators, and therefore, are no longer state variables.
- **It is not surprising that, after applying the Pantelides algorithm, we ended up with an algebraic loop. This is usually the case.**

Structural Singularity Elimination XI

Let us now return to our original circuit:



1:	u_0	=	$f(t)$
3:	u_0	=	$v_1 - v_0$
?:	u_1	=	$R_1 \cdot i_1$
?:	u_1	=	$v_1 - v_2$
?:	u_2	=	$R_2 \cdot i_2$
?:	u_2	=	$v_2 - v_0$
12:	u_L	=	$L \cdot \frac{di_L}{dt}$
10:	u_L	=	$v_2 - v_0$
11:	i_C	=	$C \cdot \frac{du_C}{dt}$
⇒:	u_C	=	$v_1 - v_0$
2:	v_0	=	0
13:	i_0	=	$i_1 + i_C$
?:	i_1	=	$i_2 + i_L$

⇒ **We need to differentiate the constraint equation.**

Structural Singularity Elimination XII

1:	u_0	=	$f(t)$
3:	u_0	=	$v_1 - v_0$
?:	u_1	=	$R_1 \cdot i_1$
?:	u_1	=	$v_1 - v_2$
?:	u_2	=	$R_2 \cdot i_2$
?:	u_2	=	$v_2 - v_0$
12:	u_L	=	$L \cdot \frac{di_L}{dt}$
10:	u_L	=	$v_2 - v_0$
11:	i_C	=	$C \cdot \frac{du_C}{dt}$
⇒:	u_C	=	$v_1 - v_0$
2:	v_0	=	0
13:	i_0	=	$i_1 + i_C$
?:	i_1	=	$i_2 + i_L$

⇒

1:	u_0	=	$f(t)$
3:	u_0	=	$v_1 - v_0$
?:	u_1	=	$R_1 \cdot i_1$
?:	u_1	=	$v_1 - v_2$
?:	u_2	=	$R_2 \cdot i_2$
?:	u_2	=	$v_2 - v_0$
13:	u_L	=	$L \cdot \frac{di_L}{dt}$
11:	u_L	=	$v_2 - v_0$
12:	i_C	=	$C \cdot du_C$
4:	u_C	=	$v_1 - v_0$
10:	du_C	=	$dv_1 - dv_0$
2:	v_0	=	0
14:	i_0	=	$i_1 + i_C$
?:	i_1	=	$i_2 + i_L$

- **In the process of differentiation, we introduced two new variables, dv_0 and dv_1, for which we don't have equations yet. We need to differentiate the equations defining v_0 and v_1 and add them to the set of equations.**

Structural Singularity Elimination XIII

1:	u_0	=	$f(t)$
3:	u_0	=	$v_1 - v_0$
?:	u_1	=	$R_1 \cdot i_1$
?:	u_1	=	$v_1 - v_2$
?:	u_2	=	$R_2 \cdot i_2$
?:	u_2	=	$v_2 - v_0$
13:	u_L	=	$L \cdot \frac{di_L}{dt}$
11:	u_L	=	$v_2 - v_0$
12:	i_C	=	$C \cdot du_C$
4:	u_C	=	$v_1 - v_0$
10:	du_C	=	$dv_1 - dv_0$
2:	v_0	=	0
14:	i_0	=	$i_1 + i_C$
?:	i_1	=	$i_2 + i_L$

⇒

1:	u_0	=	$f(t)$
3:	u_0	=	$v_1 - v_0$
11:	du_0	=	$dv_1 - dv_0$
?:	u_1	=	$R_1 \cdot i_1$
?:	u_1	=	$v_1 - v_2$
?:	u_2	=	$R_2 \cdot i_2$
?:	u_2	=	$v_2 - v_0$
15:	u_L	=	$L \cdot \frac{di_L}{dt}$
13:	u_L	=	$v_2 - v_0$
14:	i_C	=	$C \cdot du_C$
4:	u_C	=	$v_1 - v_0$
12:	du_C	=	$dv_1 - dv_0$
2:	v_0	=	0
5:	dv_0	=	0
16:	i_0	=	$i_1 + i_C$
?:	i_1	=	$i_2 + i_L$

- **In the process of differentiation, we introduced yet a new variables, du_0. We need to differentiate the equation defining u_0.**

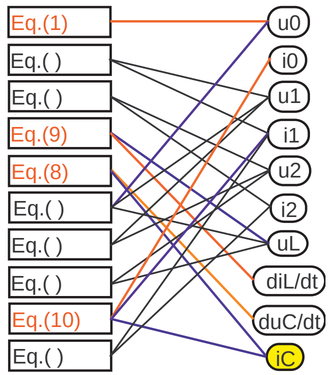
Structural Singularity Elimination XIV

$$\begin{array}{llll}
1: & u_0 & = & f(t) \\
3: & u_0 & = & v_1 - v_0 \\
11: & du_0 & = & dv_1 - dv_0 \\
?: & u_1 & = & R_1 \cdot i_1 \\
?: & u_1 & = & v_1 - v_2 \\
?: & u_2 & = & R_2 \cdot i_2 \\
?: & u_2 & = & v_2 - v_0 \\
15: & u_L & = & L \cdot \frac{di_L}{dt} \\
13: & u_L & = & v_2 - v_0 \\
14: & i_C & = & C \cdot du_C \\
4: & u_C & = & v_1 - v_0 \\
12: & du_C & = & dv_1 - dv_0 \\
2: & v_0 & = & 0 \\
5: & dv_0 & = & 0 \\
16: & i_0 & = & i_1 + i_C \\
?: & i_1 & = & i_2 + i_L
\end{array}
\quad \Rightarrow \quad
\begin{array}{llll}
1: & u_0 & = & f(t) \\
6: & du_0 & = & \frac{df(t)}{dt} \\
3: & u_0 & = & v_1 - v_0 \\
12: & du_0 & = & dv_1 - dv_0 \\
?: & u_1 & = & R_1 \cdot i_1 \\
?: & u_1 & = & v_1 - v_2 \\
?: & u_2 & = & R_2 \cdot i_2 \\
?: & u_2 & = & v_2 - v_0 \\
16: & u_L & = & L \cdot \frac{di_L}{dt} \\
14: & u_L & = & v_2 - v_0 \\
15: & i_C & = & C \cdot du_C \\
4: & u_C & = & v_1 - v_0 \\
13: & du_C & = & dv_1 - dv_0 \\
2: & v_0 & = & 0 \\
5: & dv_0 & = & 0 \\
17: & i_0 & = & i_1 + i_C \\
?: & i_1 & = & i_2 + i_L
\end{array}$$

- **We are done. We now have an algebraic loop in five equations and five unknowns.**

Structural Singularity Elimination XV

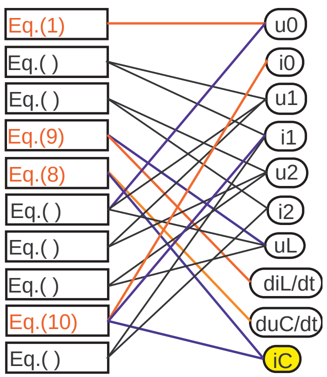
Let us now return to the original description of the model without node potentials:



- We got stuck without finding a constraint equation.
- We ended up with an algebraic loop in six equations, but only five unknowns, as the sixth unknown, i_C, doesn't appear in the algebraic loop.
- The constraint equation is hidden inside the algebraic loop.

⇒ In this situation, we need to differentiate the entire algebraic loop and add the differentiated equations to the set.

Structural Singularity Elimination XVI



$$\begin{array}{llll}
1: & u_0 & = & f(t) \\
?: & u_1 & = & R_1 \cdot i_1 \\
?: & u_2 & = & R_2 \cdot i_2 \\
9: & u_L & = & L \cdot \frac{di_L}{dt} \\
8: & i_C & = & C \cdot \frac{du_C}{dt} \\
?: & u_0 & = & u_1 + u_L \\
?: & u_C & = & u_1 + u_2 \\
?: & u_L & = & u_2 \\
10: & i_0 & = & i_1 + i_C \\
?: & i_1 & = & i_2 + i_L
\end{array}$$

⇒ We need to differentiate the entire algebraic loop and remove one of the integrators that appears inside the loop equations.

Structural Singularity Elimination XVII

$$\begin{array}{llll}
1: & u_0 & = & f(t) \\
?: & u_1 & = & R_1 \cdot i_1 \\
?: & u_2 & = & R_2 \cdot i_2 \\
9: & u_L & = & L \cdot \frac{di_L}{dt} \\
8: & i_C & = & C \cdot \frac{du_C}{dt} \\
?: & u_0 & = & u_1 + u_L \\
?: & u_C & = & u_1 + u_2 \\
?: & u_L & = & u_2 \\
10: & i_0 & = & i_1 + i_C \\
?: & i_1 & = & i_2 + i_L
\end{array}
\quad \Rightarrow \quad
\begin{array}{llll}
1: & u_0 & = & f(t) \\
?: & u_1 & = & R_1 \cdot i_1 \\
?: & du_1 & = & R_1 \cdot di_1 \\
?: & u_2 & = & R_2 \cdot i_2 \\
?: & du_2 & = & R_2 \cdot di_2 \\
?: & u_L & = & L \cdot \frac{di_L}{dt} \\
14: & i_C & = & C \cdot du_C \\
?: & u_0 & = & u_1 + u_L \\
?: & du_0 & = & du_1 + du_L \\
15: & u_C & = & u_1 + u_2 \\
13: & du_C & = & du_1 + du_2 \\
?: & u_L & = & u_2 \\
?: & du_L & = & du_2 \\
16: & i_0 & = & i_1 + i_C \\
?: & i_1 & = & i_2 + i_L \\
?: & di_1 & = & di_2 + \frac{di_L}{dt}
\end{array}$$

- **In the process of differentiation, we introduced yet a new variables, du_0. We need to differentiate the equation defining u_0.**

Structural Singularity Elimination XVIII

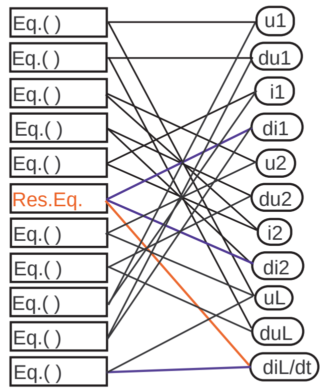
$$\begin{array}{llcl} 1: & u_0 & = & f(t) \\ ?: & u_1 & = & R_1 \cdot i_1 \\ ?: & du_1 & = & R_1 \cdot di_1 \\ ?: & u_2 & = & R_2 \cdot i_2 \\ ?: & du_2 & = & R_2 \cdot di_2 \\ ?: & u_L & = & L \cdot \frac{di_L}{dt} \\ 14: & i_C & = & C \cdot du_C \\ ?: & u_0 & = & u_1 + u_L \\ ?: & du_0 & = & du_1 + du_L \\ 15: & u_C & = & u_1 + u_2 \\ 13: & du_C & = & du_1 + du_2 \\ ?: & u_L & = & u_2 \\ ?: & du_L & = & du_2 \\ 16: & i_0 & = & i_1 + i_C \\ ?: & i_1 & = & i_2 + i_L \\ ?: & di_1 & = & di_2 + \frac{di_L}{dt} \end{array} \quad \Rightarrow \quad \begin{array}{llcl} 1: & u_0 & = & f(t) \\ 2: & du_0 & = & \frac{df(t)}{dt} \\ ?: & u_1 & = & R_1 \cdot i_1 \\ ?: & du_1 & = & R_1 \cdot di_1 \\ ?: & u_2 & = & R_2 \cdot i_2 \\ ?: & du_2 & = & R_2 \cdot di_2 \\ ?: & u_L & = & L \cdot \frac{di_L}{dt} \\ 15: & i_C & = & C \cdot du_C \\ ?: & u_0 & = & u_1 + u_L \\ ?: & du_0 & = & du_1 + du_L \\ 16: & u_C & = & u_1 + u_2 \\ 14: & du_C & = & du_1 + du_2 \\ ?: & u_L & = & u_2 \\ ?: & du_L & = & du_2 \\ 17: & i_0 & = & i_1 + i_C \\ ?: & i_1 & = & i_2 + i_L \\ ?: & di_1 & = & di_2 + \frac{di_L}{dt} \end{array}$$

- **We ended up with 17 equations in 17 unknowns, containing an algebraic loop of 11 equations and 11 unknowns.**

Tearing Algebraic Loops

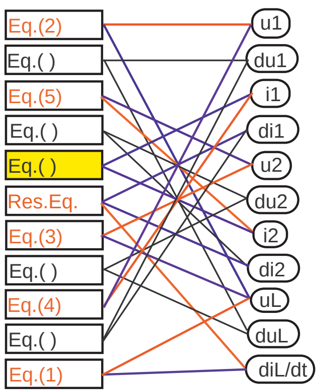
Let us look at the algebraic loop equations after selection of a tearing variable and a residual equation:

$$\begin{array}{llcl} ?: & u_0 & = & u_1 + u_L \\ ?: & du_0 & = & du_1 + du_L \\ ?: & u_2 & = & R_2 \cdot i_2 \\ ?: & du_2 & = & R_2 \cdot di_2 \\ ?: & i_1 & = & i_2 + i_L \\ \text{res.eq.:} & di_1 & = & di_2 + \frac{di_L}{dt} \\ ?: & u_L & = & u_2 \\ ?: & du_L & = & du_2 \\ ?: & u_1 & = & R_1 \cdot i_1 \\ ?: & du_1 & = & R_1 \cdot di_1 \\ ?: & u_L & = & L \cdot \frac{di_L}{dt} \end{array}$$



Tearing Algebraic Loops II

A few causalization steps later:

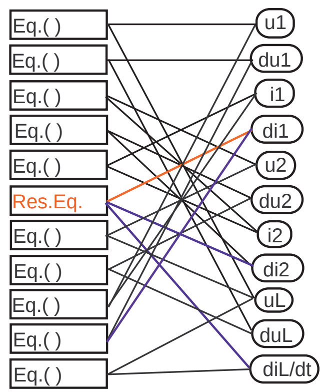


- We seem to have gotten stuck with another constraint equation.
- Yet, this is a very different problem from the one discussed before. This constraint was caused by a poor selection of a tearing variable and a residual equation.
- Had we chosen a different tearing variable or a different residual equation, this problem would not have occurred.
- Sometimes, our simple heuristics for the selection of tearing variables and residual equations maneuver themselves into a corner, and in those situations, we must be prepared to backtrack.

Tearing Algebraic Loops III

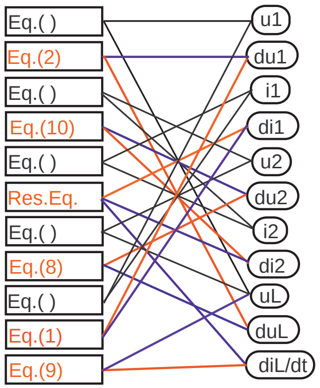
Let us select a different tearing variable from the same residual equation:

$$\begin{array}{llcl} ?: & u_0 & = & u_1 + u_L \\ ?: & du_0 & = & du_1 + du_L \\ ?: & u_2 & = & R_2 \cdot i_2 \\ ?: & du_2 & = & R_2 \cdot di_2 \\ ?: & i_1 & = & i_2 + i_L \\ \text{res.eq.:} & di_1 & = & di_2 + \frac{di_L}{dt} \\ ?: & u_L & = & u_2 \\ ?: & du_L & = & du_2 \\ ?: & u_1 & = & R_1 \cdot i_1 \\ ?: & du_1 & = & R_1 \cdot di_1 \\ ?: & u_L & = & L \cdot \frac{di_L}{dt} \end{array}$$



Tearing Algebraic Loops IV

A few causalization steps later:



- We were able to causalize six of the eleven equations.
- We thus need to select a second residual equation and a second tearing variable, in order to complete the causalization of the algebraic equation system.

Tearing Algebraic Loops V

- **Dymola** implements the Pantelides algorithm essentially in the form explained in this presentation.
- Yet, Dymola uses a more complex set of heuristics for selecting the tearing variables, one that has furthermore not been published and is therefore not available for discussion.
- Dymola often prefers to keep additional tearing variables in order to prevent divisions by zero from occurring during the simulation.
- **Sol** employs a different approach. Rather than assuming all state variables to be known and throwing out individual state variables when constraint equations are encountered, Sol assumes initially all state variables to be unknown and adds them one at a time until the number of unknowns matches the number of equations.

Conclusions

- In this presentation, we looked at the problem of *structural singularities* contained in the set of DAEs extracted from an object-oriented description of the system to be simulated.
- We discussed a variant of the *Pantelides algorithm* for the systematic index reduction in structurally singular (higher-index) models.
- The algorithm is very efficient and has been successfully implemented in **Dymola** and also in a number of other object-oriented modeling and simulation environments.

References

1. Cellier, F.E., and H. Elmqvist (1993), "Automated Formula Manipulation Supports Object-Oriented Continuous-System Modeling," *IEEE Control Systems*, **13**(2), pp.28-38.
2. Zimmer, Dirk (2010), *Equation-based Modeling of Variable-structure Systems*, Ph.D. Dissertation, Dept. of Computer Science, ETH Zurich, Switzerland.