Object-Oriented Modeling of Classic Physical Systems using Linear Implicit Equilibrium Dynamics

Dirk Zimmer *

Posted Date: 19 March 2024

doi: 10.20944/preprints202403.1139.v1

Keywords: Object-oriented modeling; Code Generation; Modeling principles

Article

Object-Oriented Modeling of Classic Physical Systems using Linear Implicit Equilibrium Dynamics

Dirk Zimmer

German Aerospace Center (DLR), Germany; dirk.zimmer@dlr.de; Tel.: +49 8153 28 13 55

Abstract: Pairs of potential and flow variables have long been established for interfacing object-oriented component models of physical systems. Recently, however, the use of triplets has been discovered for the purpose of robust modeling. New and powerful Modelica libraries have been developed such as the DLR ThermoFluid Stream library or the introduction of the Dialectic Mechanics library. Their use of triplets is entangled with a special modeling approach that uses Linear Implicit Equilibrium Dynamics. In this paper, we study the basic motivation of this approach, its benefits and drawbacks before we finally demonstrate how it beneficially impacts the generation of corresponding simulation code.

Keywords: object-oriented modeling; code Generation; modeling principles

1. Introduction

Pairs of potential (effort) and flow variables have long been established for the equation-based, object-oriented modeling of physical systems. For instance, Table 1 lists the interface currently in use for the Modelica Standard Library [1] and Table 2 lists the interface typically used for Bond-graphs [2,3].

Table 1. Connection pairs as used in the Modelica standard library

Domain	potential variable	flow variable
translational mechanics	position: r [m]	force: f [N]
rotational mechanics	angle: φ [rad]	torque: τ [Nm]
thermofluid streams	thermodynamic state: Θ [1]	mass-flow rate: $\dot{m}$ [kg/s]
electrical	voltage potential: v [V]	current: i [A]

[1] The thermodynamic state is a tuple. Its number of elements and their units may vary depending on the medium that is modelled. For an ideal gas, one may use pressure and temperature: ([Pa], [K]).

Table 2. Connection pairs as defined by the bond-graph methodology

Domain	effort	flow
translational mechanics	force: f [N]	velocity: v [m/s]
rotational mechanics	torque: τ [Nm]	angular velocity: ω [rad/s]
hydraulics/pneumatics	pressure: p [Pa]	Volumetric flow: Φ [m^3/s]
thermal	Temperature: T [K]	Entropy flow $\dot{S}$ [J/Ks]
electrical	voltage: u [V]	current: i [A]

For bond-graphs, the product of these pairs always represents a flow of energy. Although Modelica is less dogmatic, the flow of energy is still contained in the information of the connection pair.

The sheer size of the Modelica Standard Library, its industry wide adoption and also the use similar interfaces in a wide array of commercial tools provide empiric evidence that these pairs form very useful interfaces. And indeed, these pairs provide an answer for what is necessary for the object-oriented modeling of classic physical systems:

 doi:10.20944/preprints202403.1139.v1

The equations whose solution represent a physical system can be distributed among its components

Thanks to these pairs, we can indeed distribute the equations of physical systems among different components. Unfortunately, the very same pairs are also the root cause for many very persistent problems encountered in the object-oriented modeling of physical systems.

- Irregular systems can be composed and are hard to diagnose.
- Very difficult handling of variable structure systems [4] (change of equations at run-time) due to the high index of the Differential Algebraic Equation (DAE) system.
- Very difficult code generation for large-scale system simulation [5] since often the complete equation system is needed for structural analysis
- Highly complex generation of simulation code requiring compilers that have high development costs. (The decade spanning effort of OpenModelica [6] provides evidence)

All these problems will be diminished (as demonstrated in later chapters), when we fulfill what is sufficient for object-oriented modeling:

Any valid combination of components (under rules of limited complexity) shall have a solution representing a physical system.

This statement centers on the ability to make an a-priori statement on the solvability of the modeled system and sadly (as we will see later), we can cannot fulfill this statement with the presented pairs of effort and flow. Instead, this paper suggests to use triplets as in Table 3.

Table 3. Connection triplets suggested in this paper that will lead to a robustly solvable form.

Domain	signal	potential variable	flow variable
trans. mechanics	position: r [m]	velocity: v [m/s]	force: f [N]
rot. mechanics	angle: φ [rad]	ang. velocity: ω [rad/s]	torque: τ [Nm]
thermofluid str.	thermodynamic state: $\widehat{\Theta}$ [1]	inertial pressure r [Pa]	mass-flow rate: $\dot{m}$ [kg/s]
electrical	?	?	?

[1] The same comment as to Table 1 applies. Here the thermodynamic state is also formulated based on the steady-mass flow pressure $\hat{p}$. More explanation on this from section 3 onwards.

Before we can aim to find a sufficient form and understand these triplets, we first need to understand what actually led us to the necessary pairs of potential (effort) and flow variables. Hence, we need to revisit the very basic fundamentals of classic physics.

2. The Principle of Stationary Action

The central question is what minimum interface is needed so that we can distribute the equation of systems in classical physics. The natural starting point is hence the most fundamental law forming the basis of all classical physics: the principle of stationary action also known as Hamilton's Principle. It is based on the action S for an element moving along a path $q(t)$:

$$S = \int_{t_a}^{t_b} L(q(t), \dot{q}(t))\, dt = \int_{t_a}^{t_b} T\big(\dot{q}(t)\big) - V\big(q(t)\big)\, dt$$

The action is the integral of the Lagrangian L over the timespan $[t_a, t_b]$. The Lagrangian itself can be expressed as the difference between kinetic energy T and potential energy V. The principle of stationary action states that:

$$\frac{\partial S}{\partial q(t)} = 0$$

This principle is the basis of classic physics because it is also fulfilled by the underlying quantum physics although in a statistical manner as described by Feynman [7]. Yet for a sufficient large number

 doi:10.20944/preprints202403.1139.v1

of quantum events, the expected value will be fulfilled with great precision and an assumption of continuity can be made for macroscopic systems.

The principle of stationary action is here written for a conservative system. It would require extensions for non-conservative systems but there is no need here to complicate matters. In any case, the following literature may be advised [8–11] (chapter 20). A few points are however often underrepresented and hence deserve discussion here.

The principle of stationary action is based on the Lagrangian $T - V$. We see that a different sign is applied for the kinetic energy T than for all other forms of (potential) energy V. The kinetic energy has thus a special role. Not only is its role special but also its form. The gradient of the kinetic energy forms a continuous bijective function for the total domain (whether relativistic or not), meaning that it has a well-defined inverse over the complete domain. This property is important for the discussion of another question: is there always a solution for the principle of stationary action?

Given that we can assume no strict restrictions on the complexity of the potential V (other than having a derivative), it may seem unsolvable in the general case. Fortunately, the plot of Figure 1 indicates how to approach this problem.

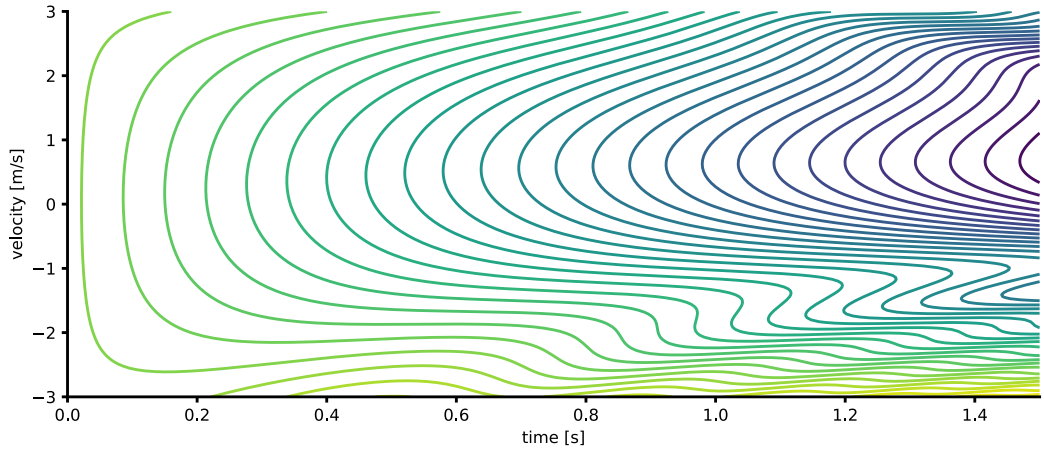


Figure 1. This topological map illustrates the complexity of the action over time (x-axis) for a straight path of constant velocity (y-axis). The potential V has here been arbitrary chosen in this example and the actual values are irrelevant. Relevant is that the complexity of the action is increasing over time.

This graph is a topological map of the action over time for paths of different constant velocity. We recognize that in this (completely arbitrary) example, the action has become a very complex function after a time-step of one second. However, for a small step of time, the action remains simple and well-natured. The explanation is straight forward: no matter how complex V is shaped, it takes a motion through time and space to gather this complexity. Hence at $t = t_a$, the well-natured kinetic energy dominates the action.

This means we can reliably solve the principle of stationary action if the step in time $t_b - t_a$ is small enough. What is small enough? Under the assumption of continuity, there is only one answer: dt. This leads us ultimately to the famous Euler-Lagrange equation and is also the explanation why time derivatives are so useful in physics.

$$\frac{\partial L}{\partial q} - \frac{d}{dt}\frac{\partial L}{\partial \dot{q}} = 0$$

Now we see also why it is so important that the partial derivative of the kinetic energy has a special form and a well-defined inverse. This ensures that the second term can always compensate the first in an unambiguous way. Please note that such a general statement on solvability is extremely helpful.

For illustration, let us now apply the Euler-Lagrange to the following example system for the pressure wave of a fluid in a pipe. The derivative of our path: $\dot{q}(t)$ represents the volume flow $\dot{Q}$. Correspondingly the path $q(t)$ then represents the shift in volume Q.

 doi:10.20944/preprints202403.1139.v1

- The kinetic energy is then defined as: $\mathrm{T} = \frac{\mathrm{I}\rho}{2}\dot{\mathrm{Q}}^2$ with ρ being the density and the inertance defined by the geometry length s and cross section A: $\mathrm{I} = \int \frac{\mathrm{ds}}{\mathrm{A}}$
- The potential energy can be formulated by $V = \frac{K}{2Q_{ref}} Q^2$ where K is the bulk modulus and Q_{ref} a reference volume.

  The Lagrangian is now defined by:

$$L = \frac{I\rho}{2}\dot{Q}^2 - \frac{K}{2Q_{ref}}Q^2$$

and we can apply the Euler-Lagrange equation which results in:

$$\frac{\partial\left(\frac{I\rho}{2}\dot{Q}^2 - \frac{K}{2Q_{ref}}Q^2\right)}{\partial Q} - \frac{d}{dt}\frac{\partial\left(\frac{I\rho}{2}\dot{Q}^2 - \frac{\mathrm{K}}{2Q_{ref}}Q^2\right)}{\partial\dot{Q}} = 0$$

Fortunately, many terms vanish when taking the partial derivative and the resulting equation is a differential equation of second order that expresses the wave in the fluid based on its volumetric shift:

$$Q\frac{K}{Q_{ref}} + I\rho\ddot{Q} = 0$$

The elegance of the Euler-Lagrange equation is that it provides a solution of n dimensions with n equations for an underlying variational problem that is actually a $2n$ dimensional problem (please note that one shall treat ∂q and $\partial\dot{q}$ as independent variables [11]).

This elegant "compression" is however working against the intent of object-oriented modeling: for examples of higher dimensions, it will be very difficult to redistribute the equations to individual components. Hence the result of Euler-Lagrange is mostly unsuited for object-oriented modeling.

Fortunately, there is an alternative form that can be achieved by the Legendre transformation that transforms the Lagrangian $T - V$ into the Hamiltonian $T + V$. The stationary action principle can now be reformulated for our example. For the Legendre transformation we require the generalized momentums that are defined as

$$p_i = \frac{\partial L}{\partial\dot{q}^i}$$

For illustration, let us reiterate our previous example. Based on the former Lagrangian, we can determine the generalized momentum p:

$$p = \frac{\partial L}{\partial\dot{q}} = \frac{\partial(\frac{I\rho}{2}\dot{Q}^2 - \frac{K}{2Q_{ref}}Q^2)}{\partial\dot{Q}} = I\rho\dot{Q}$$

Using p to express the kinetic energy T is then applied for composing the Hamiltonian $H = T + V$:

$$H = \frac{1}{2}\frac{p^2}{I\rho} + \frac{K}{2Q_{ref}}Q^2$$

Given the Hamiltonian, we can now apply the famous pair of Hamilton equations:

$$\frac{dq}{dt} = \frac{\partial H}{\partial p}$$

$$\frac{dp}{dt} = -\frac{\partial H}{\partial q}$$

In our example, these translate to:

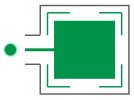
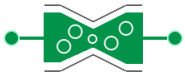
$$\frac{dq}{dt} = \dot{Q} = \frac{\partial\left(\frac{1}{2}\frac{p^2}{I\rho} + \frac{K}{2Q_{ref}}Q^2\right)}{\partial p} = \frac{p}{I\rho}$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 March 2024 doi:10.20944/preprints202403.1139.v1

$$\frac{dp}{dt} = -\frac{\partial\left(\frac{1}{2}\frac{p^2}{I\rho} + \frac{K}{2Q_{ref}}Q^2\right)}{\partial q} = -\frac{Q}{\kappa Q_{ref}}$$

We now get $2n$ first order differential equations, based on generalized position and generalized momentum. These equations express the same system as the previous solution derived by Euler-Lagrange. (simply differentiate the first equation and eradicate dp/dt by substitution). Different from the Euler-Lagrange solution, these equations can be nicely distributed among components, especially if we reformulate them based on a corresponding pair. In our setting it is natural to choose the volume flow $\dot{Q} = dq/dt$ and the pressure $P = dp/dt$ as pair. The corresponding components are presented in Table 4, row 1 and 2.

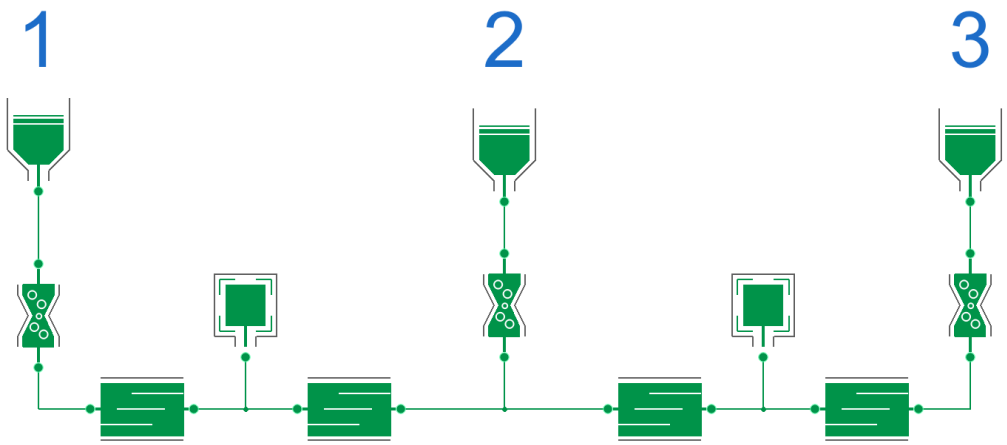
Table 4. Examples of components formulated based on the pairs resulting from the Hamiltonian equations. The variables forming the pair are marked in bold.

Component	Symbol	Equation
Fluid Inertance		$\frac{d\dot{\boldsymbol{Q}}}{dt} I\rho = \Delta \boldsymbol{P}$
Compressible volume		$\frac{d\boldsymbol{P}}{dt}\frac{Q_{ref}}{K} = -\dot{\boldsymbol{Q}}$
Tank [1]		$\frac{d\boldsymbol{P}}{dt}\frac{A}{\rho g} = -\dot{\boldsymbol{Q}}$
Flow resistance [2]		$\zeta\frac{\dot{\boldsymbol{Q}}\lvert\dot{\boldsymbol{Q}}\rvert}{\dot{Q}_{ref}^2} = \Delta \boldsymbol{P}$

[1] A is the cross-section area of the tank and g the gravitational constant. [2] $\dot{Q}_{ref}^2$ is a reference volume flow and ζ is the quadratic friction coefficient.

We can then choose to model a different storage of potential energy such as gravitational pressure. Even though all our derivations were done on conservative systems, the same pair can be used to model non-conservative components such as the friction of a valve. These two additional components are presented in Table 4, row 3 and 4.

Given these 4 components of Table 4 and the pair of effort and flow, we can now assemble a more complex system and for instance model the example of 3 communicating vessels as in Figure 2. Classic object-oriented modeling becomes functional.



 doi:10.20944/preprints202403.1139.v1

Figure 2. This diagram represents a model of three communication vessels. Gravitational pressure is modeled at the tanks. Non-linear flow resistance is used to model the valve at the outlets. The connecting pipes are modeled using inertances and compressible volumes.

In the corresponding simulation results (Figure 3) we can see the oscillation that result from the natural parameters of water. As typical for many systems the exchange between potential and kinetic energy is of high frequency.

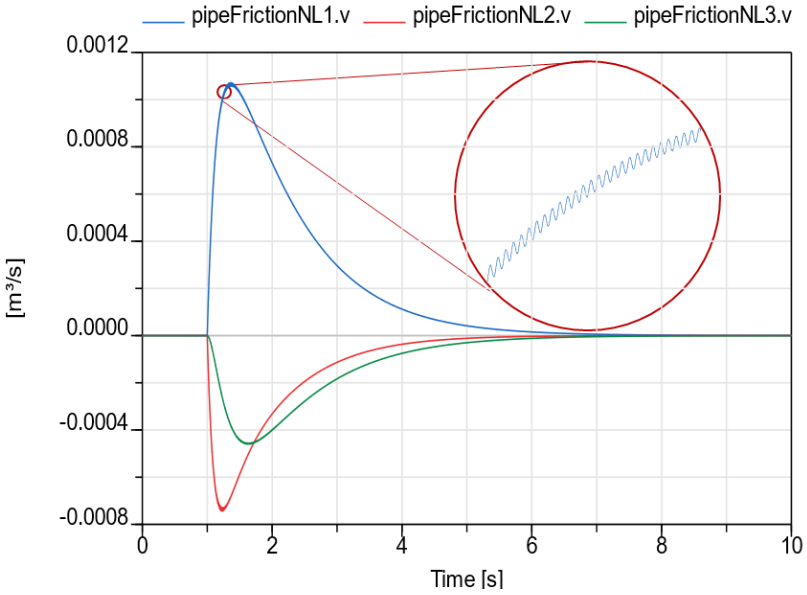


Figure 3. Simulation result of the model of Figure 2, showing the 3 volume flows going through the tank's orifices. A sudden change of water level is imposed to the system a time $t = 1$ to trigger the low-frequency response.

3. From Necessary to Sufficient

We have seen that the well-known pairs of potential and flow variables can be derived from the Hamiltonian form of the principle of stationary action (although not always in the exact same way as for the hydraulic example above).

Different form the Lagrangian $T - V$, the kinetic energy loses its special role in the Hamiltonian $T + V$. Treating all energy forms equal may deceivingly suggest that energy flows are all what the modelers needs (Bond graph modelling is a prime example of this fallacy). Yet, whereas each valid Hamiltonian represents a sum of energy terms, not all sums of energy terms represent a valid Hamiltonian. This leads to a serious problem for object-oriented modeling. Whereas the pairs are necessary to redistribute the equations resulting out of a valid Hamiltonian, only very few recombination possible by these pairs form a valid Hamiltonian. This implies that the modeler can formulate non-physical systems too and modelers even do this on purpose (albeit often without being fully aware of the consequences). For instance, let us consider the following recombination of components as depicted in Figure 4.

 doi:10.20944/preprints202403.1139.v1

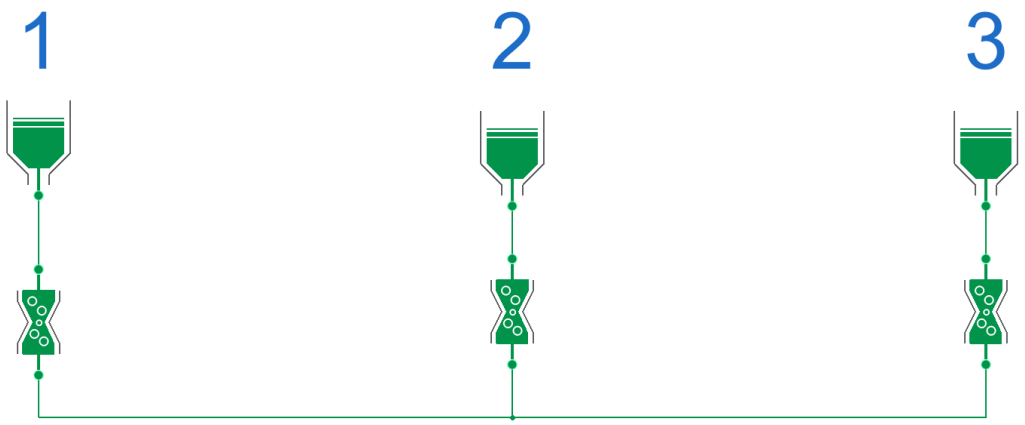


Figure 4. Modeling 3 communicating vessels without modeling the pipe just by the gravitational pressure and the flow-resistance of the orifice. Note, that there is an algebraic non-linear equation system to determine the pressure below the valves so that all flows are in balance.

Here the modeler has now completely removed the inertia and compressibility of water from the model. This means that there is no representation of the kinetic energy anymore. We have removed exactly the component that is required to formulate the principle of stationary action and guarantee its solvability. Instead we now use the pairs to formulate a constraint problem in the domain of energy. Whether this can be solved is in general undefined. And indeed, we have to solve a non-linear equations system in this example to perform the simulation. In this example, the solution could reliably be found during simulation time but in the general case no statement on solvability can be given.

It is however important to recognize that modelers do such things (mostly) not out of malevolence or ignorance but to solve relevant problems. The slight additional cost of solving the non-linear equation system was rewarded in this example by the much higher gain of removing the high frequency from the system enabling a very efficient simulation of the system (albeit significantly altering the trajectory).

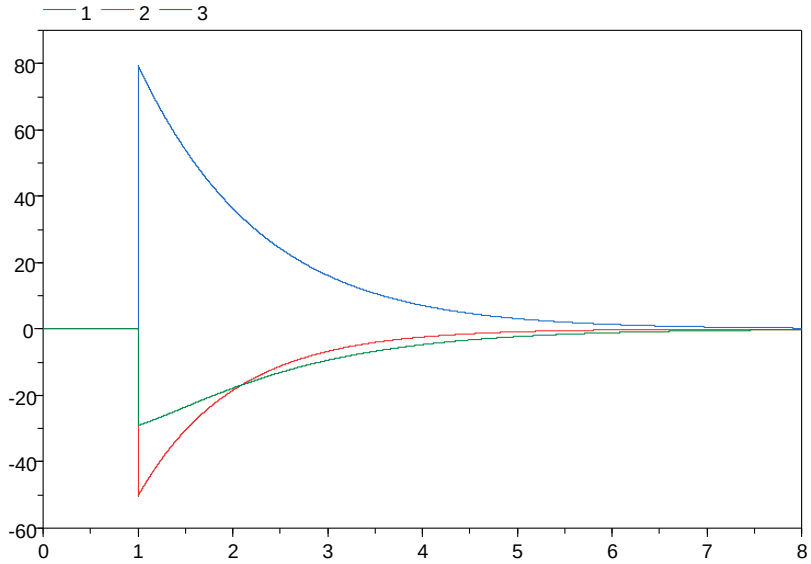


Figure 5. Simulation result of the model of Figure 4, showing the 3 volume flows going through the tank's orifices. In this simplified model the high frequency has been eliminated and the flow rate is now dominated by the resistance.

Hence, what would we need to do to ensure solvability and also enable powerful modeling approach? Essentially, we have to fulfill two criteria:

 doi:10.20944/preprints202403.1139.v1

(a) Ensure that the kinetic energy is always sufficiently represented so that we can guarantee solvability according to the steady-action principle.
(b) Enable the modeler to reduce the interaction between kinetic and potential energies to key points in the system so that irrelevant high frequency behavior can be effectively suppressed.

For some domains, these two criteria can be met in a methodological sound manner by going from a pair to a triplet.

Part of this triplet is a pair that expresses the kinetic energy. This pair is accompanied by a signal that describes the non-linear configuration under which the kinetic energy can be exchanged. This rather abstract picture is best described by concrete examples. One for thermo-fluid systems and one for mechanical systems.

In the following we discuss the modeling of two domains using triplets.

3.1 Triplet Interface for Thermofluid Systems

Here is the triplet interface for thermo-fluid streams:

- r: inertial pressure (potential)
- $\dot{m}$: mass-flow rate (flow)
- $\widehat{\Theta}$: Vector representing the thermodynamic state of the medium (signal)

- $\hat{p}$: steady-mass flow pressure
- $\hat{h}$: steady-mass flow enthalpy
- X: mass fractions

Key to understand this interface is the dynamic of the inertial pressure r. It represents the pressure needed to accelerate a mass-flow which forms linear law based on the inertance [12,13] of the fluid L:

$$r = L\frac{d\dot{m}}{dt}$$

The inertance L is thereby independent of the thermodynamic state of the fluid and just defined by the geometry of the flow with the length ds and cross section area A:

$$L = \int \frac{ds}{A}$$

If we now enforce that the inertance law is part of every single component, we have fulfilled criterion (a).

To fulfill criterion (b), we decompose the pressure into two components:

$$p = \hat{p} + r$$

The complement to r is hence the newly defined steady-mass flow pressure $\hat{p}$ For the steady-state with a constant mass flow rate, r will go to zero and $p = \hat{p}$, which explains its name.

A very elegant way to reduce the interaction points between potential energies and kinetic energy is to use $\hat{p}$ to form an approximation for the thermodynamic state $\widehat{\Theta}$ of the fluid. Furthermore, where it is not reasonable to mix pressure, it is feasible to provide mixing laws for $\hat{p}$. This enables an explicit downstream computation of $\widehat{\Theta}$.

The kinetic energy will then only interact at the boundaries of streams which are either volume elements or sources and sinks. At junctions, the kinetic energy of different streams may interact with each other. Ultimately this means that a different spatial resolution is applied for the effects of r than for $\hat{p}$. This assumption is in line with many modeling tasks and deviations are limited to rapid transients where model uncertainty is typically very high. At steady mass flows there is no difference which is important for most applications.

The resulting computational scheme is that all the non-linear computations regarding the thermodynamic state including properties like viscosity, heat conductivity or density are performed in explicit form downstream. This means that the signal $\widehat{\Theta}$ is always computed from the source to the sink. To ensure this, all circular flows need to contain at least one volume. A linear system then computes the time derivative of all mass-flows. This system is formed by the equation for the pairs

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 March 2024 doi:10.20944/preprints202403.1139.v1

of r and $\dot{m}$. It is guaranteed to be linear because of the characteristics of L and it will be regular since all inertances are strictly greater than zero.

The reader is advised to literature [14,15] to follow the above statements in detail. Revisiting our example of section 2, there is fortunately a less complex replacement model. Since our models assume constant density and viscosity, there is no need for the decomposition of p and for the signal $\widehat{\Theta}$. The solution would (in this particular case) be equivalent to represent the inertance for each mass-flow but we are free to neglect the compressibility. The corresponding diagram is shown in Figure 6 and the simulation result in Figure 7. Also because each mass flow is represented by the sum of all of its inertances, the frequency of the response is low.

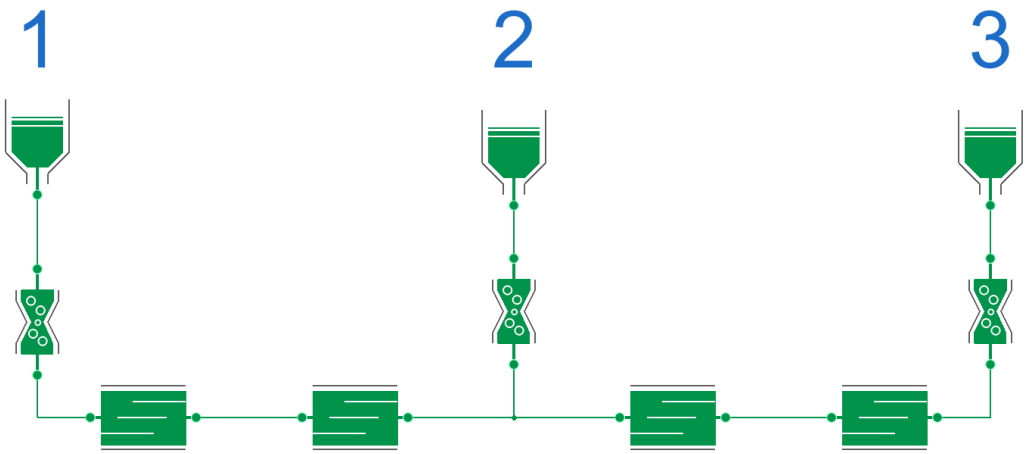


Figure 6. For this particular model of the communicating vessels, the LIED approach has an equivalent counterpart using conventional connectors. Modeling the inertia but leaving out the compressibility does the trick here.

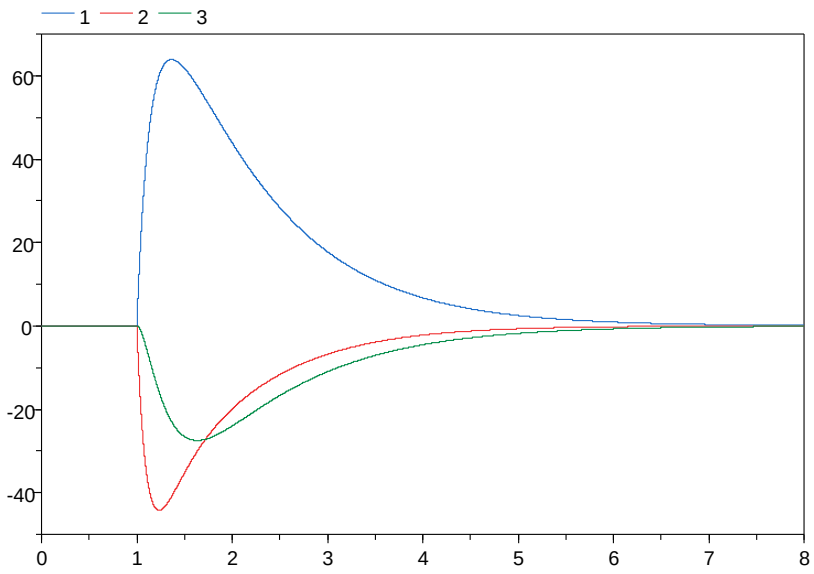


Figure 7. Modelling the communicating vessels by the sheer exchange of potential energy.

3.2 Triplet Interface for Mechanical Systems

It is possible to formulate a similar triplet to model multi-body mechanics:

- v: velocity (potential)
- f: force (flow)
- s: position (signal)

 doi:10.20944/preprints202403.1139.v1

Where v and s represent positions and velocities in a generalized way for both translatory and rotational motion.

The product of velocity and force thereby represents a flow of energy which will contain also the kinetic energy. To ensure criterion (a), we simply must enforce that there are no massless components or (by the means of connection rules) that there is at least a mass representing all degrees of freedom of a kinetic chain.

To fulfill criterion (b), we have to address the elements of a multi-body system that define the degrees of freedom for the system: the joint elements. These are typically revolute joints or prismatic joints but exist in various combinations that define a different set of positional states. Naturally, the velocity is defined as direct derivative of the position and the acceleration as second derivative:

$$\frac{dv}{dt} = \frac{d^2s}{dt^2}$$

In section 3.1, we have reduced the interaction of kinetic and potential energy by lowering the spatial resolution of the kinetic part. For multi-body mechanics we have to reduce the temporal resolution of the kinetic part. We can do so by applying a first-order filter:

$$\frac{dv}{dt}T_D = \frac{ds}{dt} - v$$

Using the time constant T_D, we can now effectively limit the occurring frequencies of the mechanical system (at the cost of becoming more energy dissipative) without modifying the steady-state solution.

Just for the sake of quick illustration: Figure 8 from [16] shows the dynamics of a lightweight object moved in clamp modeled by a very stiff spring. The figure simply illustrates how the oscillatory dynamics is approximated with a replacement dynamics leading to the same (quasi) steady-state solution.

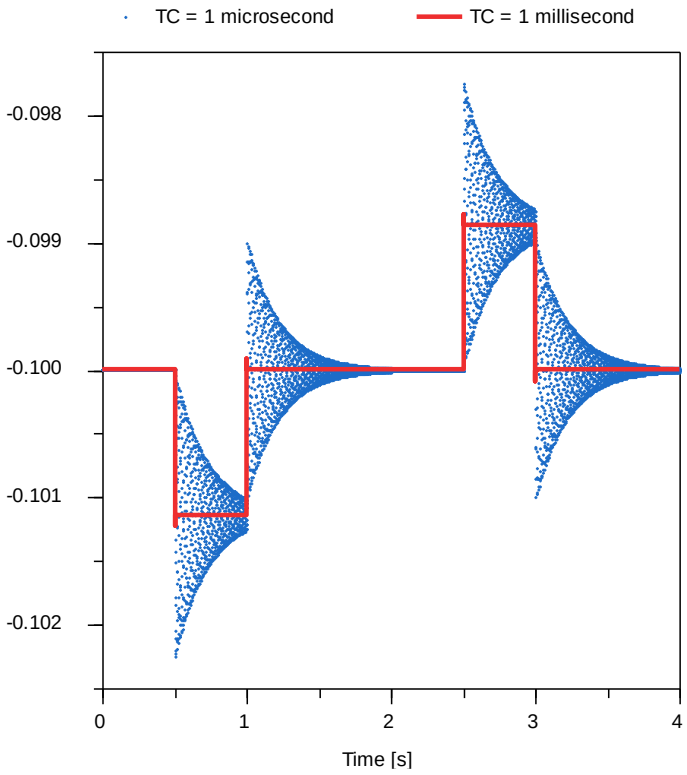


Figure 8. Penetration depth into the left claw represented by an elasto-gap, for the choice of two different time constants, taken from [16]. Both agree on the time-averaged solution. TC of the legend represents T_D. The dots of the blue result from a decaying high-frequent oscillation. The red line is the low frequency approximation.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 March 2024 doi:10.20944/preprints202403.1139.v1

Using the triplet, we can formulate a difference between the motion in the elastic regime ds/dt and the velocity in the kinetic regime v. Because of this separation of two regimes, the author has denoted it as dialectic mechanics. The interface enables again that all non-linear equations regarding the geometrical configuration can be formulated in explicit form from the root of the kinematic chain to its end by the signal s. To ensure this tree structure, all kinematic loops must contain a flexible element. The pair of v and f then sets up a linear system of equations for the kinetic energy.

First implementations and analysis of this modeling approach are presented in [16,17]. Models using this interface are especially suitable for the simulation of contacts and limited joints also under hard-real time constraints. We expect many future applications in robotics and related tasks.

4. Triplets and Object-Oriented Modeling

The two triplets presented in section 3 can be directly transformed into code of the equation-based, object-oriented language Modelica. Table 5 contains the corresponding code. For both domains, corresponding Modelica libraries have been developed.

Table 5. The triplets as represented in the Modelica language for equation-based modeling.

Thermofluids	Mechanics

connector Outlet
  output[1] Medium.ThermState state
  "thermodynamical state";
  Pressure r "inertial pressure";
  flow MassFlowRate m_flow
  "mass flow rate";
end Outlet;
	connector FlangeOn1DTranslatory[2]
	  output[1] Position s
	  "positional signal";
	  Velocity v "(kinetic) velocity";
	  flow Force f "force";
	
	end FlangeOn1DTranslatory;

[1] For each connector with an output there is also a counterpart with an input. Two sexes of connectors are thus needed. Only one is shown. The flow variable is marked in Modelica by the keyword **flow**. An unmarked variable is regarded as potential variable. [2] For the mechanical connector also a variant with two pairs of effort and flow is presented in [16]. This will be however discontinued in favor of the presented triple.

The DLR ThermoFluid Stream Library [15] is a fully functional open-source that is already in use both in industry and academia. Figure 9 shows an exemplary model diagram of the thermal architecture of an electric car. We can clearly see that a complex architecture is nicely composed out of its individual components. The end-user is not directly confronted with the triplets. The modeler simply needs to take care about the connection rules that all loops contain at least one volumetric element.

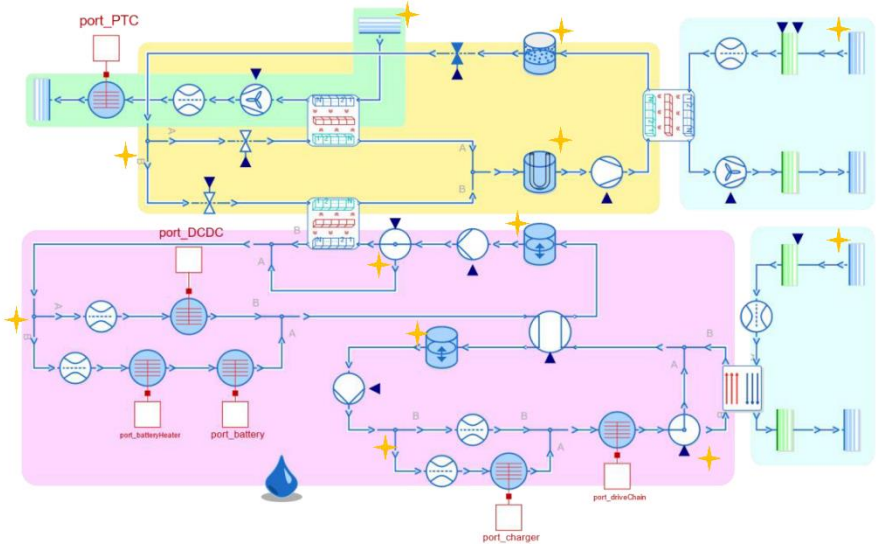


Figure 9. Exemplary model diagram taken from [15] of the thermal architecture of an electric car. Different cooling loops, including a vapour cycle can be combined to cool the power electronics,

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 March 2024 doi:10.20944/preprints202403.1139.v1

battery and passenger compartment of the car. Certain components are marked with a star which is not part of the original diagram. The meaning of this marking is explained in section 6.

The library for Dialectic Mechanics is still under internal development and testing. Figure 10 shows the model diagram of a kinematic loop for a hypothetical landing gear kinematic. This demonstrates that also here, complex systems can be composed in an object-oriented fashion and the end-user simply has to uphold the connection rules. The kinematic loops a closed by flexible elements and each chain contains at least one mass.

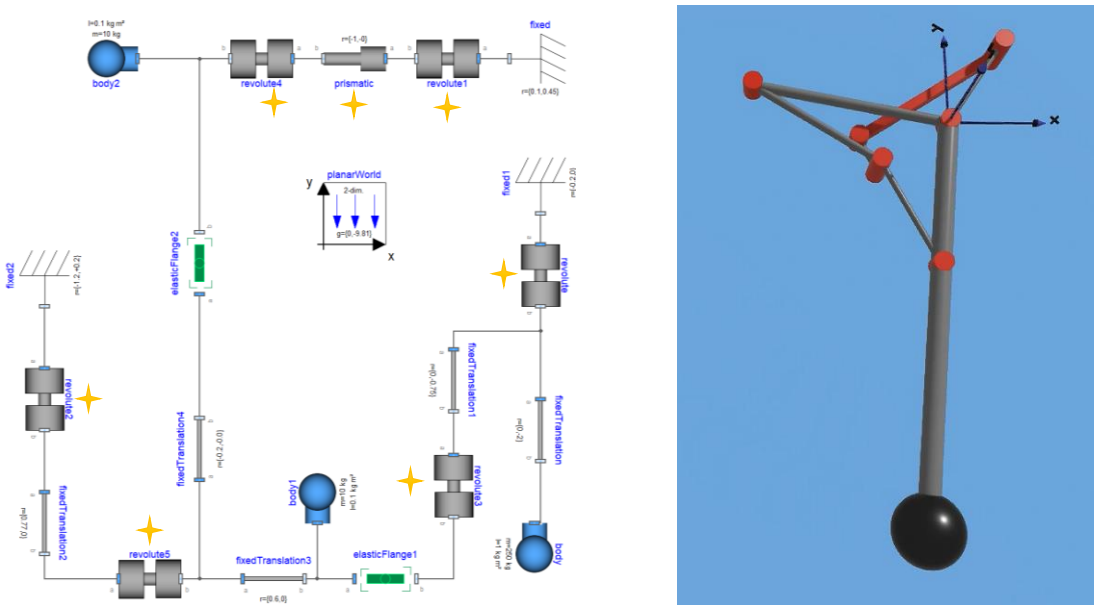


Figure 10. Modelica Diagram (left) of a landing gear kinematic (right) using the Dialectic Mechanics library. The interface of the LIED approach leads to a nice decomposition into ideal components. Regarding the star marking, the same comment as in Figure 9 applies.

Using Modelica for these examples, a complete DAE system is built based on the composed models. The complete system of equations is then flattened (meaning that all equations are collected in a global set) and then state variables are selected and symbolic index reduction (as for example in [18]) is performed to transform the DAE into a set of Ordinary Differental Equations (ODEs). This typically results in a larger piece of simulation code where all variables are globally collected.

5. Triplets and Linear Implicit Equilibrium Dynamics

We have derived the triplet interface out of the motivation to have a robust modeling interface that ensures solvability and yet enables effective modeling by the suppression of high frequency behavior (that occurs in real physical systems). It is remarkable that for both cases the resulting computational structure of the complete system is of a particular form that we define here as Linear Implicit Equilibrium Dynamics (LIED).

A DAE system with potential state derivatives $\dot{\boldsymbol{x}}$, time t and algebraic variables $\mathbf{w}$

$$\mathbf{0} = \boldsymbol{F}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \mathbf{w}, t)$$

is defined as LIED system when it can be transformed into the following form:

$$\begin{bmatrix} \mathbf{w}_E \\ \dot{\boldsymbol{x}}_E \end{bmatrix} = \mathrm{g}(\boldsymbol{x}_I, \boldsymbol{x}_E, t)$$

$$\mathbf{A}(\boldsymbol{x}_I, \boldsymbol{x}_E, \mathbf{w}_E) \begin{bmatrix} \mathbf{w}_I \\ \dot{\boldsymbol{x}}_I \end{bmatrix} = \mathrm{f}(\boldsymbol{x}_I, \boldsymbol{x}_E, \mathbf{w}_E, t)$$

 doi:10.20944/preprints202403.1139.v1

We see that both the algebraic variables as well as the state derivatives can be split into a fully explicit part ($\dot{\boldsymbol{x}}_E$; $\mathbf{w}_E$) and a part ($\dot{\boldsymbol{x}}_I$; $\mathbf{w}_I$) with a linear system in implicit form expressed by the matrix **A**. Furthermore, the following conditions shall hold true:

- $\dot{\boldsymbol{x}}_E \cap \dot{\boldsymbol{x}}_I \subseteq \dot{\boldsymbol{x}}$
- $\mathbf{w}_E \cap \mathbf{w}_I \supseteq \mathbf{w}$
- $\dot{\boldsymbol{x}}_E \cap \dot{\boldsymbol{x}}_I \cap \mathbf{w}_I \supseteq \dot{\boldsymbol{x}}$
- $\dot{\boldsymbol{x}}_E, \dot{\boldsymbol{x}}_I, \mathbf{w}_E, \mathbf{w}_I$ are all disjoint

These conditions essentially mean that it is allowed to perform certain symbolic mechanism of index reduction such as the dummy derivative method [18] originating from Pantelides [19]. Using this method, states variables of $\boldsymbol{x}$ can be transformed to algebraic variables in $\mathbf{w}_I$ and further derivatives may be added to $\mathbf{w}_I$ or $\mathbf{w}_E$. In practice this is important because it means that the linear implicit dynamics can be expressed by far fewer states than suggested in the original DAE formulation.

To get a practical feeling for this LIED form, let us discuss the decomposition of the linear and non-linear part, for the presented triplets.

- For thermofluid streams, the signal for the thermodynamic state $\widehat{\Theta}$ will be part of $\mathbf{w}_E$. States associated with this state such as the temperature of a volume will be part of $\boldsymbol{x}_E$. A subset of the flow variables $\dot{m}$ will form $\boldsymbol{x}_I$. The subset is chosen in such a way that all streams are described by $\boldsymbol{x}_I$ in a non-redundant way. The redundant mass flows as well as the inertial pressures r become part of $\mathbf{w}_I$.
- For dialectic mechanics, the signal for the position will be part of $\mathbf{w}_E$. The potential and flow variables of the interface v and f will be part of $\mathbf{w}_I$. The state vector $\boldsymbol{x}_E$ is typically empty. The members of $\boldsymbol{x}_I$ are chosen from the internal variables in such a way that the motion of the kinematic chain is described in a non-redundant way. For each degree of freedom its kinetic velocity and its position form state variables in $\boldsymbol{x}_I$.

6. How to Create Simulation Code for LIED Systems?

The original intention of the LIED approach was simply to ensure that no non-linear system is created in implicit form that spans across the components and hence a robust solution of the model evaluation could be taken for granted, given robust component models. When we started with it, we expected it to be the only notable change from other DAE models in Modelica and that all other features of a of code generation (as briefly outlined at the end of section 4) would basically remain untouched.

However, over time, it was gradually recognized that LIED systems are much simpler to transfer to simulation code than general DAEs. Let us go through these observed simplifications one-by one:

- Because we avoid the creation of non-linear equation systems, we do not need a non-linear equation system solver anymore.
- For the same reason, constraint equations among potential states cannot be non-linear and hence no dynamic state selection is needed [20]
- Even stronger: we can select the states on component level. For example, the stars in Figure 9 and 10 mark those components that define states in $\boldsymbol{x}_I$.
- Because we can select the states on component level, this means that the dummy-derivative method can be applied also on component level before system composition.
- Since the goal of the linear equation system is to have a synchronized replacement dynamics towards the equilibrium, we know suitable tearing variables for this system. These will be the linear state derivatives: $\dot{\boldsymbol{x}}_L$ or at least a subset of it.
- The residual for a tearing variable can be attributed to the same component as the tearing variable.

The terminology of the last two points may require further explanation. A tearing variable is a variable that can be used to probe the reaction of an algebraic system which is assessed by a corresponding residual. If the system is linear: $\mathbf{M}\mathbf{w} = \mathbf{b}$, the probing can be used to construct **M** and **w** and find the solution for **w**.

 doi:10.20944/preprints202403.1139.v1

All of the points above represent observations resulting from modeling many components and system examples using the LIED approach. However, these observations have profound implications. For each component we know:

- the set of pairs of state-variables and their derivatives it adds to the system.
- the set of pairs of tearing and residual variables it adds to the system.

If this is the case, we can basically causalize everything already on the component level. In concrete terms, this means for each component:

- we stipulate the states
- we stipulate the tearing variables of the linear system and the corresponding residuals
- we perform the dummy derivative method on those equations where necessary.
- we define the causality of the interface variables
- we causalize all equations into assignments in a particular order
- we group the list of assignments depending on their dependence of the inputs.

Practical experience so far indicates that performing index reduction to transform to the LIED form can be performed in a very methodical and deterministic manner. It is thus far easier to generate simulation code for the LIED modeling approach than it would be for general higher-index DAEs. Neither there is a need for global flattening anymore nor are elaborate heuristics needed for the selection of state or tearing variables. Indeed, the generation of simulation code is so easy that a direct implementation in C++ becomes feasible. To demonstrate this, the following code excerpts illustrate the implementation for a thermofluid stream library (using idealized water) in C++.

First, we have to define the interface. This is naturally more tedious than in a language like Modelica because there is no direct support in the C++ language. Yet, it is feasible and after all, interfaces only need to be defined once:

Listing 1. ThermoFluid Interface in C++

```
class ThermodynamicStateOut: public Signal{
  public:
    double p; //pressure [Pa] (steady mass flow)
    double h; //enthalpy [J] (steady mass flow)
     […]
};

class ThermodynamicStateIn: public ThermodynamicStateOut
{
  public:
    void connect(ThermodynamicStateOut* o);
    […]
};

class MassFlowOut : public Signal{
  public:
    double flow; //mass flow rate [kg/s]
    double flow_der; //derivative of mass flow rate [kg/s2]
    […]
};

class MassFlowIn : public MassFlowOut{
  public:
     void connect(MassFlowOut* o);
    […]
};

class InertialPressureOut : public Signal{
```

 doi:10.20944/preprints202403.1139.v1


```
  public:
    double r;
    […]
};

class InertialPressureIn : public InertialPressureOut
{
  public:
    void connect(InertialPressureOut* o);
    […]
};

class ThermalPlugOut : public Signal{
   public:
     ThermodynamicStateOut state{};
     MassFlowOut m{};
     InertialPressureIn inertial{};
     […]
};
class ThermalPlugIn : public Signal{
  public:
    ThermodynamicStateIn state{};
    MassFlowIn m{};
    InertialPressureOut inertial{};
    void connect(ThermalPlugOut* o);
    […]
};

class Connection {
  public:
    Connection(ThermalPlugOut* o,
               ThermalPlugIn* i) {
                i->connect(o);
     };
};
typedef std::vector<Connection>  Connections;
```

To best understand the interface, let us look at the classes `ThermalPlugOut` for a nominal outlet flow and at `ThermalPlugIn` for a nominal inlet flow first. These contain the same 3 components as the corresponding Modelica connector of the DLR ThermoFluid Stream library.

There are two notable differences however. In Modelica, inertial pressure and mass flow were not causalized signals as in the C++ implementation. Also the mass-flow signal in the C++ library consists of the mass-flow rate and its derivative. In Modelica, this is not necessary since symbolic differentiation can be applied by the Modelica compiler. Using this interface, we can now implement a component such as the pressure drop:

Listing 2. Implementation of a pressure drop component

```
class PressureDrop : public Component{
  public:
    ThermalPlugIn inlet;
    ThermalPlugOut outlet;
     PressureDrop(double v_ref,double dp_ref)
    void evalState();
```

 doi:10.20944/preprints202403.1139.v1


```
    void evalFlow();
    void evalInertial();
    double v_ref;
    double dp_ref;

    virtual void metainfo(Meta& meta)
  override;
    […]
};
```

First, we declare our interface for outlet and inlet, then we have to implement three methods. The first is `evalState` and computes the thermodynamic state downstream:

Listing 3. Calculation of the pressure drop by the corresponding method

```
void PressureDrop::evalState() {
   const double v =
     inlet.m.flow / density(inlet.state);
   const double v_norm = v/v_ref;
   const double dp = 0.5*dp_ref*
     (v_norm + v_norm*v_norm);
   outlet.state.h = inlet.state.h;
   outlet.state.p = inlet.state.p – dp;
}
```

The second method is `evalFlow` to ensure what flows in is what flows out. However, this constraint is restated for the derivative. This is because the dummy derivative method is applied on the component level.

Listing 4. Trivial implementation of `evalFlow`

```
void PressureDrop::evalFlow() {
  outlet.m = inlet.m;
}
```

The third one is `evalInertia` that implements the law for the inertance as in the ThermoFluid Stream Library.

Listing 5. Calculation of the inertial pressure

```
void PressureDrop::evalState() {
   const double v =
     inlet.m.flow / density(inlet.state);
   const double v_norm = v/v_ref;
   const double dp = 0.5*dp_ref*
     (v_norm + v_norm*v_norm);
   outlet.state.h = inlet.state.h;
   outlet.state.p = inlet.state.p – dp;
}
```

In similar manner the other components of our introductory example can be implemented. Each of these components declares its interfaces, defines and implements methods representing the computational blocks. For all these components, meta-information must be collected by a dedicated virtual method to register state and tearing variables as well as to track the signal dependence of the

 doi:10.20944/preprints202403.1139.v1

computing blocks. Finally, we can compose the introductory example of section 2 and 3 directly in C++:

Listing 7. Total system composition

```
class ComVessels: public Component {
public:
  OutTank t1{};
  InTank t2{};
  InTank t3{};
   Splitter s{};
  PressureDrop p1{};
  PressureDrop p2{};
  PressureDrop p3{};

  Connections con {
    Connection{&t1.outlet, &p1.inlet},
    Connection{&p1.outlet, &s.inlet},
    Connection{&s.outlet1, &p2.inlet},
    Connection{&p2.outlet, &t2.inlet},
    Connection{&s.outlet2, &p3.inlet},
    Connection{&p3.inlet, &t3.inlet},
  };
  […]

};
```

Regarding that C++ is a statically compiled imperative general-purpose language, the final result is astonishingly close to what a modeler is used to from a domain specific language like Modelica.

When an instance of the class is coupled to a simulator, the meta information of all its computational methods is collected as well as the structural information regarding the signals, the states and the tearing variables for the linear equation system. Then the methods are put into right order for complete or partial model evaluation. The full model evaluation is thus simply a list of method calls that are called in their respective order. The linear implicit system defined by $\mathbf{A}(\boldsymbol{x}_I, \boldsymbol{x}_E, \mathbf{w}_E)$ is reconstructed by using the tearing variables and their residuals for probing. This requires a number of partial model evaluations.

This also means that all component code is statically compiled but the total system composition is performed at run-time. Advanced tasks such as variable structure systems would thus be comparably easy to achieve.

This is also the purpose of this code demonstration. It is not suggesting that we should use C++ as modeling language like Modelica but to highlight that for a certain class of models the object-oriented modeling code could (and maybe should) be translated to object-oriented imperative code that can be statically compiled even before total system composition. This would avoid the flattening of all equations before code generation and help to overcome many limitations of current Modelica compilers with respect to scalability.

7. Conclusions

This paper started with a very fundamental consideration about the solvability of general formulations for classic physical systems, then presented a new interface for the object-oriented modeling and ended up showing a simplified generation of simulation code potentially suitable to handle challenges such as large-scale system simulation or variable structure systems. It is quite

 doi:10.20944/preprints202403.1139.v1

remarkable that the two derived criteria for handling the kinetic energy have such practical consequences for object-oriented modeling.

Indeed, that a more restrictive class of modeling enables a simpler compilation scheme is neither new nor surprising. The same can be said about the many conventional signal-based modeling schemes or simple modeling schemes as Forrester's System Dynamics [21]. Typically, the disadvantage is that the easier generation of simulation code has to be paid by an inferior modeling approach and indeed modeling complex mechanics or thermofluid streams is painful when using purely signal-based approaches (nevertheless this pain has been taken in industrial practice all too often).

The remarkable thing about the LIED approach is that a simulation engineer has a simple scheme for code generation but can also conveniently model both mechanics and thermo-fluid streams in a very robust manner. The corresponding Modelica Libraries prove this [15,16]. Both application domains are known to be rather difficult but LIED can even be applied to the challenging parts of these fields such as handling stiff contact mechanics or elaborate by-passes in complex thermal architectures. It is yet unclear for what other domains LIED is an attractive choice.

Figure 11 attempts to qualitatively depict the trade-off between computational complexity and algorithmic complexity. LIED forms a very exposed point on a hypothetical Pareto front. This means that for a large number of applications it is a very attractive choice.

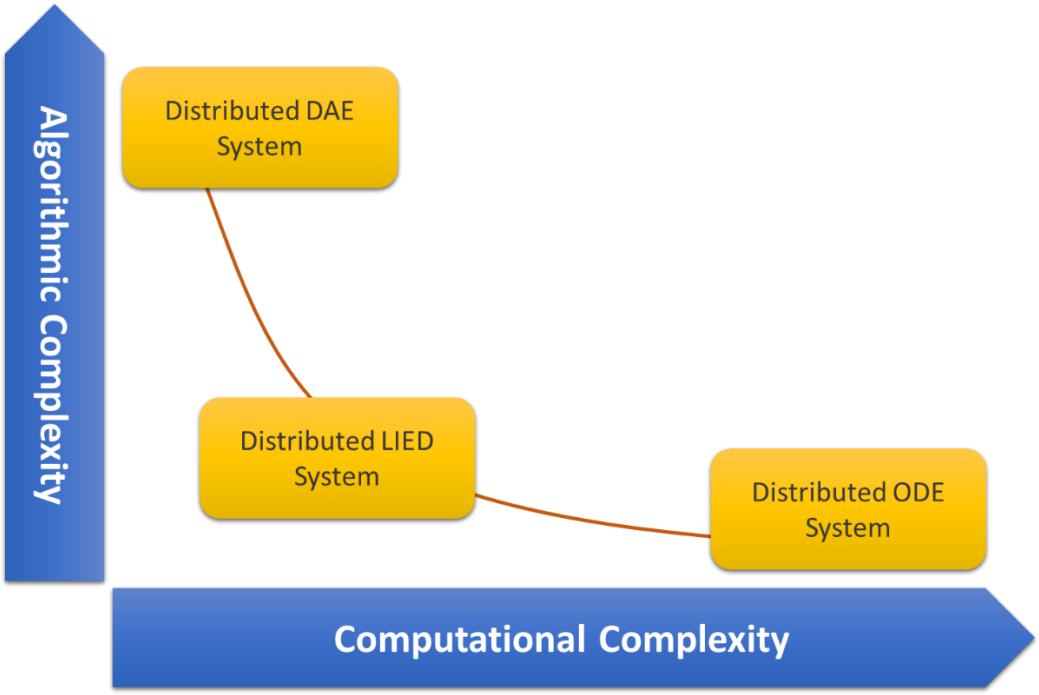


Figure 11. Hypothetical Pareto front weighing computational complexity against algorithmic complexity for code generation. Distributed (object-oriented) DAE Systems enable a very effective compression of reality and very efficient models. The same compression is often infeasible to achieve with distributed ODE system but code is much easier to generate. LIED systems form an attractive compromise. The ultimate choice of the modeling approach depends however on the concrete application.

It might be not unimportant to note that the actual discovery of the LIED approach happened not the way it was presented in this paper. Rather the opposite reflects reality: practical applications came first; generalization and theory came second. The approach was simply born out of the necessity (or desperation) to find robust solutions for aircraft environmental control systems and to handle gripping of stiff objects with robots.

For this paper, this means that most of its statements are to be regarded as empirical. For some specific LIED systems, formal statements on structure and solvability could be made [14] but in

general, the statements on suitable system composition and code generation are merely observations from simulation practice and test implementations. Further solidification of the theoretical understanding is needed. This is a call to ourselves but also an invitation to the research community.

As the question marks in Table 3 indicate, there remains an open question whether the LIED approach can also be successfully applied for electrical systems so that it is of practical value. Clearly, kinetic energy plays a far less important role for electrical systems, also the formulation of potential energy requires a generalized form. Hence, the reasoning of this paper cannot be directly converted. Maybe extensions are needed, maybe it is infeasible? Again, this open question is an invitation to the research community.

The primary motivation of this paper remains to raise awareness on this class of models and the possibilities it enables for the generation of simulation code. Although it states mostly observations, these observations seem valuable to share.

Supplementary Materials: The following supporting information can be downloaded at the website of this paper posted on Preprints.org.

Funding: This research received no external funding.

Acknowledgments: The author wants to thank Prof. Martin Otter for additional background on Hamilton's principle and a review of the LIED definition. Furthermore the author expresses his gratitude to Modelica Community and the many though-provoking and encouraging discussions in the aftermath in the context of this research.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. P. Fritzson, Principles of Object-Oriented Modeling and Simulation with Modelica 3.3, 2nd ed., IEEE Press, Piscataway, New Jersey, 2014, pp. 1256.
2. Cellier, F.E. (1991) Continuous System Modeling. Springer Verlag New York. 755p.
3. Karnopp, D.C., D.L. Margolis, R.C. Rosenberg (2006), System Dynamics: Modeling and Simulation of Mechatronic Systems. 4th Edition, John Wiley&Sons, New York, 576p.
4. Zimmer, D. (2010), Equation-Based Modeling of Variable Structure Systems PhD Dissertation, ETH Zürich, 219 pages.
5. Braun, Willi et al. "Solving large-scale Modelica models: new approaches and experimental results using OpenModelica." International Modelica Conference (2017).
6. Fritzson, Peter A. et al. "The OpenModelica Integrated Modeling, Simulation, and Optimization Environment." Proceedings of The American Modelica Conference 2018, October 9-10, Somberg Conference Center, Cambridge MA, USA (2019)
7. Feynman, Richard P. (1942). Laurie M. Brown (ed.). The Principle of Least Action in Quantum Mechanics. PhD Dissertation, Princeton University. World Scientific (with title "Feynman's Thesis: a New Approach to Quantum Theory") (published 2005). ISBN 978-981-256-380-4.
8. Georg Hamel, Theoretische Mechanik 1949 (Reprint 1978)
9. Torby, Bruce (1984). "Energy Methods". Advanced Dynamics for Engineers. HRW Series in Mechanical Engineering. United States of America: CBS College Publishing. ISBN 0-03-063366-4.
10. Landau LD and Lifshitz EM (1976) Mechanics, 3rd. ed., Pergamon Press. ISBN 0-08-021022-8.
11. Penrose, R., The Road to Reality. Vintage Books, New York, 2004.
12. B.S. Massey, Mechanics of Fluids, Chapman & Hall. ISBN 0-412-34280-4, 1989.
13. C.E. Brennen, Internet Book on Fluid Dynamics, Dankat Publishing, Pasadena, California, USA, 2015.
14. Zimmer, D. (2020), Robust Object-Oriented Formulation of Directed Thermofluid Stream Networks . Mathematical and Computer Modelling of Dynamic Systems, Vol 26, Issue 3.
15. Zimmer, D., N. Weber, M. Meißner (2022) The DLR ThermoFluid Stream Library. MDPI Electronics - Special Issue.
16. Zimmer, D., C. Oldemeyer (2023). "Introducing Dialectic Mechanics". Proceedings of the 15th International Modelica Conference, Aachen.
17. Oldemeyer, C., D. Zimmer (2023). "Dialectic Mechanics: Extension for Hard Real-time Simulation". Proceedings of the 15th International Modelica Conference, Aachen.
18. Mattsson, S.E., Gustaf Söderlind (1993). "Index Reduction in Differential-Algebraic Equations Using Dummy Derivatives" In: SIAM Journal on Scientific Computing 1993 14:3, 677-692
19. Pantelides, C. (1988), The consistent initialization of differential-algebraic systems, SIAM J. Sci. Statist. Comput., 9 (1988), 213–231

 doi:10.20944/preprints202403.1139.v1

20. Sven Erik Mattsson, H. Olsson, H. Elmqvist (2000) "Dynamic Selection of States in Dymola". Proceedings of Modelica Workshop 2000.
21. Junglas, P. (2016), Causality of System Dynamics Diagrams, SNE Simulation Notes Europe 26/3, 147-154