Mathematical Modeling in Modelica: The Art of Compressing Reality

Dirk Zimmer
DLR
Institute of System Dynamics and Control

North American Modelica Conference
27.10.2022 Dallas
Equation-Based, Object-Oriented Modeling

Control (Torque, Pitch)

Inertia

model Inertia
    Interfaces.Flange_a flange_a;
    parameter SI.Inertia J;
    SI.AngularVelocity w;
    SI.AngularAcceleration a;
    equation
        w = der(flange_a.phi);
        a = der(w);
        J*a = flange_a.tau;
    end Inertia;
Customer Deployment, Digital Twins
Collaborative Design Processes, MBSE
Local Design Optimization and Control
Basic Model Evaluation / Simulation
Multi-Domain Total System Composition
Object-Oriented System Composition
Idealized Component Classes
Fundamental Laws

Application-oriented part in imperative form

Knowledge-oriented part in declarative form

Code Generation
Simulation Error

A fatal exception occurred at 027:C8127 by the non-linear equation system solver. Here is a cryptic error code that is of absolutely no use: 420.
Simulation has been stopped to prevent damage from your virtual universe.

*press any key to acknowledge defeat
*press Ctrl+Alt+Del if you think that this is any better
*by the way, we deleted your hard-drive

Press any key to continue
Motivation Example: An Air Cycle
Motivation Example: An Air Cycle

- This leads to a system with more than 200 non-linear equations (with more than 40 iteration variables)
Simulation Error

A fatal exception occurred at 027:C8127 by the non-linear equation system solver. Here is a cryptic error code that is of absolutely no use: 420. Simulation has been stopped to prevent damage from your virtual universe.

*press any key to acknowledge defeat
*press Ctrl+Alt+Del if you think that this is any better
*by the way, we deleted your hard-drive

Press any key to continue _
Simulation does not always work
Simulation does not always work
A fatal exception occurred at 027:C8127 by the non-linear equation system solver.

Here is a cryptic error code that is of absolutely no use: 420.
Simulation has been stopped to prevent damage from your virtual universe.

*press any key to acknowledge defeat
*press Ctrl+Alt+Del if you think that this is any better
*by the way, we deleted your hard-drive

Press any key to continue _
**Application-oriented part in imperative form**

- Customer Deployment, Digital Twins
- Collaborative Design Processes, MBSE
- Local Design Optimization and Control
- Basic Model Evaluation / Simulation

**Knowledge-oriented part in declarative form**

- Multi-Domain Total System Composition
- Object-Oriented System Composition
- Idealized Component Classes
- Fundamental Laws

**Code Generation**
Customer Deployment, Digital Twins
Collaborative Design Processes, MBSE
Local Design Optimization and Control
Basic Model Evaluation / Simulation
Multi-Domain Total System Composition
Object-Oriented System Composition
Idealized Component Classes
Fundamental Laws

Application-oriented part in imperative form
Knowledge-oriented part in declarative form

Expert Knowledge
Code Generation
Idealization is the Centerpiece of Modeling

- Reduction of Domains
- Reduction of Spatial Complexity
- Isolation of Interaction
- Assuming Instantaneous Equilibrium
- Taking One as a Whole

Final result → Hagen Poiseuille

$$\Delta p = \frac{8\mu L Q}{\pi R^4}$$
Idealization is the Centerpiece of Modeling

- Reduction of Domains
- Reduction of Spatial Complexity
- Isolation of Interaction
- Assuming Instantaneous Equilibrium
- Taking One as a Whole
Idealization is the Centerpiece of Modeling

- Reduction of Domains
- Reduction of Spatial Complexity
- Isolation of Interaction
- Assuming Instantaneous Equilibrium
- Taking One as a Whole
Idealization is the Centerpiece of Modeling

- Reduction of Domains
- Reduction of Spatial Complexity
- Isolation of Interaction
- Assuming Instantaneous Equilibrium
- Taking One as a Whole
Idealization is the Centerpiece of Modeling

- Reduction of Domains
- Reduction of Spatial Complexity
- Isolation of Interaction
- Assuming Instantaneous Equilibrium
- Taking One as a Whole
Idealization is the Centerpiece of Modeling

Reduction of Domains
Reduction of Spatial Complexity
Isolation of Interaction
Assuming Instantaneous Equilibrium
Taking One as a Whole
Idealization provides means to compress reality

Reduction of Domains  Reduction of Spatial Complexity  Isolation of Interaction  Assuming Instantaneous Equilibrium  Taking One as a Whole

Real system  Information Loss (implicit or explicit)  Idealization  Mathematical Model  Simulation

Or taking video compression as analogy

Raw Video Data  Loss function  Compression  Video file  Video encoding
Idealization provides means to compress reality

Reduction of Domains  Reduction of Spatial Complexity  Isolation of Interaction  Assuming Instantaneous Equilibrium  Taking One as a Whole

Real system  Information Loss (implicit or explicit)  Idealization  Mathematical Model  Simulation

These fundamental concepts and postulates, which cannot be further reduced logically, form the essential part of a theory, which reason cannot touch. It is the grand object of all theory to make these irreducible elements as simple and as few in numbers as possible.

Idealization provides means to compress reality

Reduction of Domains → Reduction of Spatial Complexity → Isolation of Interaction → Assuming Instantaneous Equilibrium → Taking One as a Whole

Real system → Information Loss (implicit or explicit) → Idealization → Mathematical Model → Simulation

Elon Musk @elonmusk · 20. Sep. 2018
Antwort an @vicentes @DanAloni und @slashdot
Physics can be thought of as the compression algorithms of reality

70 141 1.064
Idealization provides means to compress reality

Reduction of Domains  Reduction of Spatial Complexity  Isolation of Interaction  Assuming Instantaneous Equilibrium  Taking One as a Whole

Real system  Information Loss (implicit or explicit)  Idealization  Mathematical Model  Simulation

 [...] But simplicity certainly reflects what we mean by understanding: **understanding is compression**. So perhaps this is more about the human mind than it is about the universe.

- Gregory J Chaitin,

On the intelligibility of the universe and the notions of simplicity, complexity and irreducibility, 2002

Founder of algorithmic information theory
The problem of over-idealization

Fails to achieve instantaneous equivalent of blade element torque and air-flow momentum change

Fails to achieve instantaneous balance of power between compressor, fan and turbine

Fails to attribute kinetic energy at point of maximum stretch
Over-idealization is predominant in text books
The problem of over-idealization: idealization in components

\[ u_0 = 6V \]

\[ 6V \rightarrow i \rightarrow 12V \]

\[ u_1 = R_1 i \Rightarrow i = G_1 u_1; \quad u_0 = u_1 + u_2; \quad u_2 = R_2 i \Rightarrow i = G_2 u_2; \]

\[ u_1 = \frac{G_2 u_0}{G_1 + G_2} = \frac{0u_0}{0 + 0} = \text{Singular!} \]

If we idealize on the component level:

\[ \lim_{\Theta \to 0} G_1(\Theta) = \lim_{\Theta \to 0} G_1'\Theta \Rightarrow G_1 = 0 \]

\[ \lim_{\Theta \to 0} G_2(\Theta) = \lim_{\Theta \to 0} G_2'\Theta \Rightarrow G_2 = 0 \]
The problem of over-idealization: idealization in components

\[ u_0 = 6V \]

\[ u_0 = u_1 + u_2; \]

\[ u_2 = R_2 i \Rightarrow i = G_2 u_2; \]

\[ u_0 = \lim_{\Theta \to 0} \frac{G'_2 \Theta u_0}{G'_1 \Theta + G'_2 \Theta} = \frac{G'_2}{G'_1 + G'_2} u_0 \quad \text{Regular!} \]

\[ u_1 = \frac{G_2 u_0}{G_1 + G_2} \]

\[ \Rightarrow \quad u_1 = R_1 i \Rightarrow i = G_1 u_1; \]

... when idealization is applied on system level.
Over-idealization is the root of our problems.

No tool can recover information that has been lost due to idealization.
Over-idealization is the root of our problems.

So how to fix the issue?
What do we want to compress?

For system simulation, we aim to compress the Eigendynamics.
What do we want to compress?

For system simulation, we aim to compress the **Eigendynamics**

Reduction of Domains
What do we want to compress?

For system simulation, we aim to compress the **Eigendynamics**

- Reduction of Domains
- Reduction of Spatial Complexity
What do we want to compress?

For system simulation, we aim to compress the **Eigendynamics**

- Reduction of Domains
- Reduction of Spatial Complexity
- Isolation of Interaction
What do we want to compress?

For system simulation, we aim to compress the **Eigendynamics**

- Reduction of Domains
- Reduction of Spatial Complexity
- Isolation of Interaction
- Assuming Instantaneous Equilibrium
- Taking One as a Whole
What do we want to compress?
For system simulation, we aim to compress the **Eigendynamics**

- Reduction of Domains
- Reduction of Spatial Complexity
- Isolation of Interaction
- Assuming Instantaneous Equilibrium
- Taking One as a Whole
What do we want to compress?

- Hence we end up with differential equations for the dynamic part and algebraic parts to enable compression

- Unfortunately, there is no practical way to ensure the solvability of the non-linear landscape

- Hence, I propose to aim for a different target...
What do we want to compress?

- The implicit system that balances the dynamics shall be only linear, so that it has exactly one solution.

- We use auxiliary dynamics (blue) to express the non-linearities that cannot be expressed by the linear part.

- We can find such forms for complex physical system.

- This form is an attractive balance between effectiveness and scalability.
An effective compression is mostly about making a deliberate error.

We shall look for an error that helps to compress a lot but impacts the simulation result only by little

How to do that for thermofluid streams?
Starting from Euler’s law for 1D fluid stream

- Let us start with Euler’s equation
- And bring it into integral form for an arbitrary pipe section

\[
\rho \frac{\partial v_s}{\partial t} + \rho v_s \frac{\partial v_s}{\partial s} = -\frac{\partial p}{\partial s} - \frac{\partial p_{ext}}{\partial s}
\]

\[
\int \rho \frac{\partial v_s}{\partial t} ds + \rho \bar{v} \Delta v = -\Delta p - \Delta p_{ext}
\]
Starting from Euler’s law for 1D fluid stream

- We can substitute $v_s$ by the mass flow rate:

$$v_s = \frac{\dot{m}}{\rho A_s}$$

- Assuming the mass flow rate is constant over the stream line, we can take it out of the integral.

- The **inertance** $L$ is defined as

$$L = \int \frac{1}{A_s} ds$$

- And the **inertial pressure** $r$ is defined by

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

\[
\int \rho \frac{\partial v_s}{\partial t} ds + \rho \ddot{v} \Delta v = -\Delta p - \Delta p_{ext}
\]

\[
\frac{d\dot{m}}{dt} \int \frac{1}{A_s} ds + \Delta q = -\Delta p - \Delta p_{ext}
\]

\[
\Delta r + \Delta q = -\Delta p - \Delta p_{ext}
\]
Starting from Euler’s law for 1D fluid stream

- Let us define the **steady mass flow pressure** $\hat{p}$

\[ p = \hat{p} + r \]

- It is defined as the complement to the inertial pressure. Plugging in the definition…

- … and rearranging…

- … let us remind that $q$ and $p_{ext}$ depend on the thermodynamic state of the medium

\[ \Delta r + \Delta q = -\Delta p - \Delta p_{ext} \]

\[ \Delta q = -\Delta \hat{p} - \Delta p_{ext} \]

\[ \Delta \hat{p} = \Delta p_{ext} + \Delta q \]

\[ \Delta \hat{p} = \Delta p_{ext}(p, \dot{m}, ...) + \Delta q(p, \dot{m}, ...) \]
Starting from Euler’s law for 1D fluid stream

\[
\Delta \hat{p} = \Delta p_{ext}(\hat{p}, \dot{m}, ...) + \Delta q(\hat{p}, \dot{m}, ...)
\]

"On ne voit bien qu'avec le cœur. L'essentiel est invisible pour les yeux"
Starting from Euler’s law for 1D fluid stream

- Using \( \hat{p} \) instead of \( p \) is acceptable because:
  - when \( d\hat{m}/dt = 0 \), the error is zero
  - for gases, \( r \) is typically small
  - for liquids, the RHS is typically insensitive to \( r \)
  - many formulas (as for friction) anyway assume steady mass flow…
  - One may also say that \( \hat{p} \) and \( r \) have a different spatial resolution.
- The scheme is applicable to 96.3% of our typical Modelica use cases…

\[
\Delta \hat{p} = \Delta p_{ext}(\hat{p}, \hat{m}, ...) + \Delta q(\hat{p}, \hat{m}, ...)
\]
Starting from Euler’s law for 1D fluid stream

- Using $\hat{p}$ instead of $p$ is acceptable because:
  - when $\frac{dm}{dt} = 0$, the error is zero
  - for gases, $r$ is typically small
  - for liquids, the RHS is typically insensitive to $r$
  - many formulas (as for friction) anyway assume steady mass flow…
  - One may also say that $\hat{p}$ and $r$ have a different spatial resolution.
- The scheme is applicable to 96.3% of our typical Modelica use cases…

$$\Delta \hat{p} = \Delta p_{ext}(\hat{p}, \dot{m}, ...) + \Delta q(\hat{p}, \dot{m}, ...)$$

Real system $\rightarrow$ Information Loss (implicit or explicit) $\rightarrow$ Idealization
A First System: Resulting Equations

The black equations can be computed straightforward downstream.

What about the red equations? How to get: \( \frac{d\hat{m}_1}{dt} \), \( r_1 \) \( \frac{d\hat{m}_2}{dt} \), \( r_2 \)
A first circuit: resulting equations

We can directly extract:
• \( \frac{dm_1}{dt} \cdot L_1 = r_1 - r_A = r_1 \)
• \( \frac{dm_2}{dt} \cdot L_2 = r_2 - r_A = r_2 \)

The B junction results in:
• \( \hat{p}_1 + r_1 = \hat{p}_2 + r_2 \)

Index-reduction over junction A yields:
• \( \frac{d\dot{m}_1}{dt} = -\frac{d\dot{m}_2}{dt} \)

from \( \dot{m}_1 + \dot{m}_2 = \dot{m}_0, \dot{m}_0 \) given

Simple linear equation system
• Matrix has only constant entries
• Can be inverted upfront
• Dymola does that!

Downstream computation of thermodynamic state, assuming
Computation of mass-flow change using linear equation system over \( r \) (mostly upfront invertible)
Resulting BLT Form

- The large block at the end is linear and invertible upfront

Robust component models
  ➔
Robust system model

- Size of the linear block proportional to number of branches

small non-linear blocks for complex components

trivial components
Resulting BLT Form

small non-linear blocks for complex components

trivial components

Linear plane
And this method scales…

- The ENERGIZE Model describes a single-aisle more-electric aircraft
- It contains a combines electrical power and thermal management
- It allows the simulation of full flight missions under a wide variety of climatic conditions
  - > 18,000 equations
  - > 300 states
  - faster than real-time
Positioning of the approach

**Finite Volume Approach:**
- No non-linear equation systems
- Many states
- Stiffness
- High-Frequency Oscillations

**DLR ThermoFluid Stream:**
- No non-linear equation systems
- Few states
- Manipulable Stiffness
- Manipulable Frequency

**Algebraic Stream Approach:**
- Complicated non-linear equation systems
- No states at all or very few
- Non-stiff / no oscillations
How about mechanics?

**Mass-Spring Approach**
- No non-linear equation systems
- Many states
- Stiffness
- High-Frequency Oscillations

**Rigid Multibody**
- Few states
- Very efficient
- Non-linear equation systems
- Not feasible for contacts

**Performance**
Reformulation of multibody dynamics

Again all non-linear computations are explicit.

The implicit system, is a linear system of dimension 8. Easy to solve.

Despite the hard-impulses, we can use explicit integration method. This system was simulated with RKFIX3 at only 100Hz.

Perfectly suited for hard real-time simulation (this was 300x too fast).

No need for complicated tooling to handle structural changes anymore....
Reformulation of multibody dynamics

More fun with hard contacts…

(again using explicit integration methods with fixed step size)
How about mechanics

**Mass-Spring Approach**
- No non-linear equation systems
- Many states
- Stiffness
- High-Frequency Oscillations

**Dialectic Mechanics**

**Rigid Multibody**
- Few states
- Very efficient
- Non-linear equation systems
- Not feasible for contacts
Let us tackle the problem at its root…
Take home lessons I

In the Modelica Association (but also in the M&S community) we have misattributed our efforts:

- Too much focus on languages, tools, and computation
- Too little focus on modelling, underlying principles and information

There is a great potential to drastically reduce complexity in the simulation of classic physical systems but you have to deviate from classic textbook idealization.
## Take home lessons II

**Find the DLR ThermoFluid Stream Library on GitHub:**
https://github.com/DLR-SR/ThermofluidStream

Feel free to contribute!!

<table>
<thead>
<tr>
<th>Release</th>
<th>Dymola</th>
<th>OpenModelica</th>
<th>Modelon Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>v0.1-beta</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>v0.2-beta</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Latest main</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

Papers on dialectic mechanics will be coming soon...
(and probably some sort of free library)