Using Artificial States in Modeling Dynamic Systems: Turning Malpractice into Good Practice

Dirk Zimmer
DLR, Institute of System Dynamics and Control

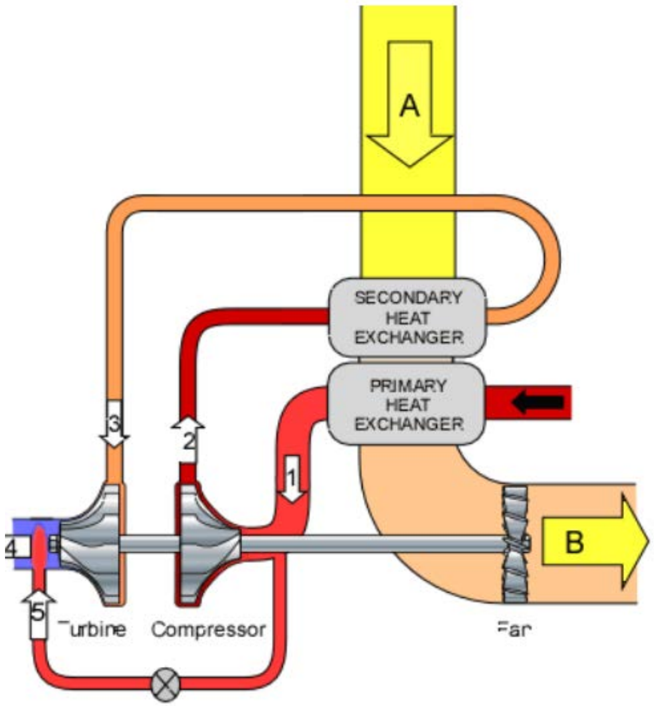
DLR

A New Way to Model Non-linearities

- This presentation offer a new way for the modeler to describe his systems of non-linear equations so that they can be solved more robustly.
- To this end, we introduce a new operator and present a corresponding algorithm
- The idea originated from many years of practical modeling experience.
- So let us start by a practical example.

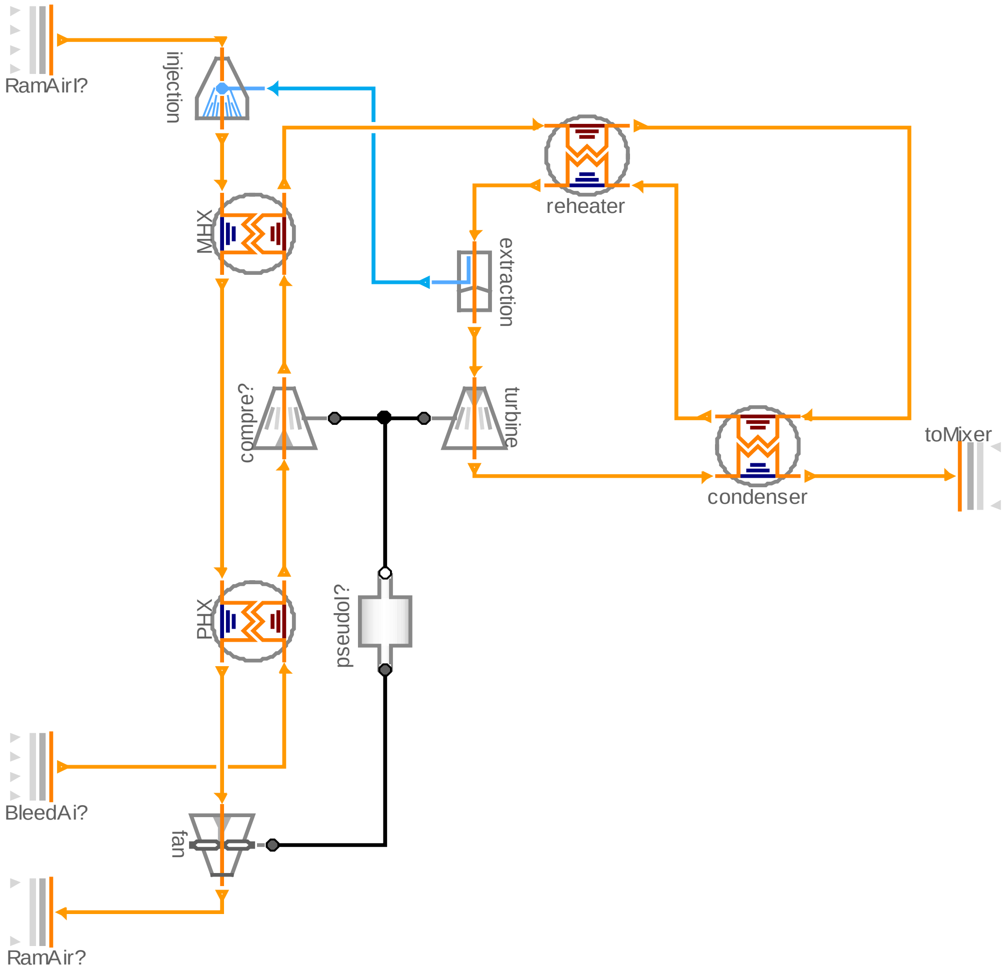
Example: ECS Air Cycle

- In aircraft, bleed air from the engine is used to pressurize the cabin.
- Bleed air is hot: 220° C
- Bleed air is at high pressure: 2.5 bar
- So it needs to be cooled down and expanded before it enters the cabin.
- One architecture to achieve this is the three wheel bootstrap circuit



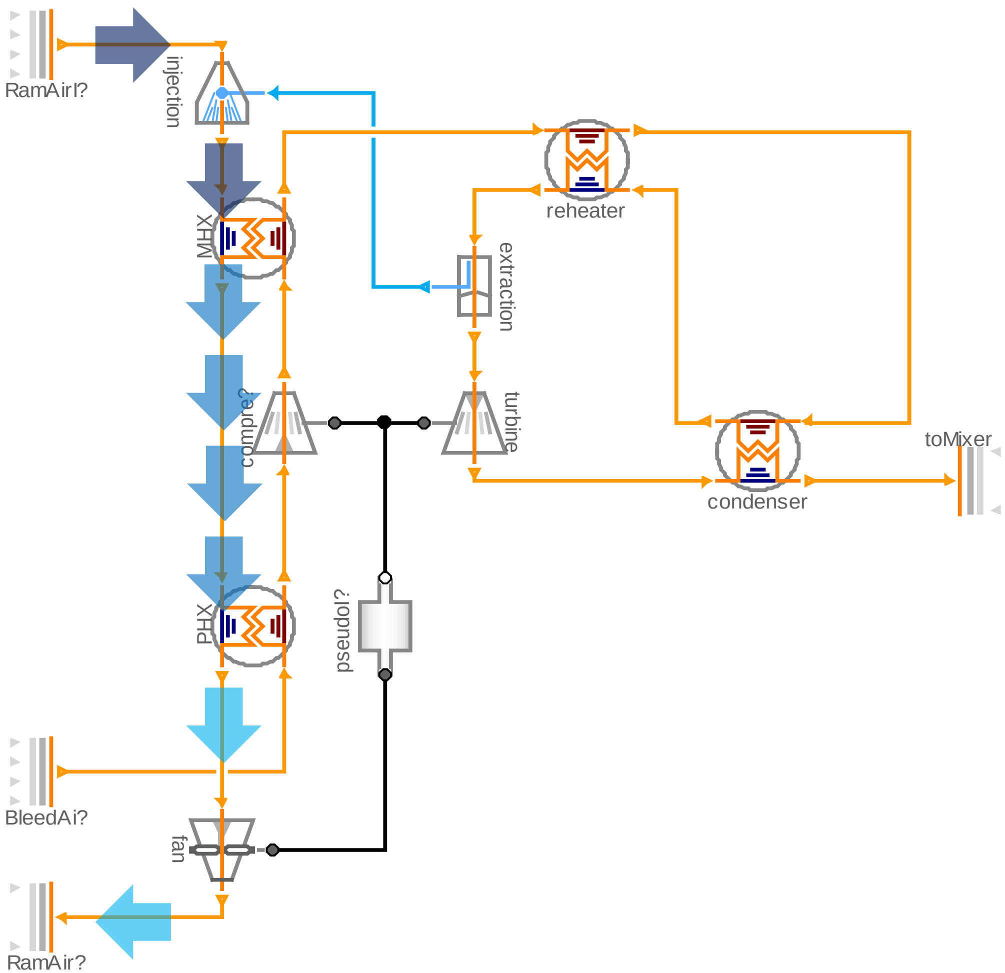
Source: ECS Blog

Three Wheel Bootstrap Circuit



- At DLR, we modeled this circuit using Modelica:

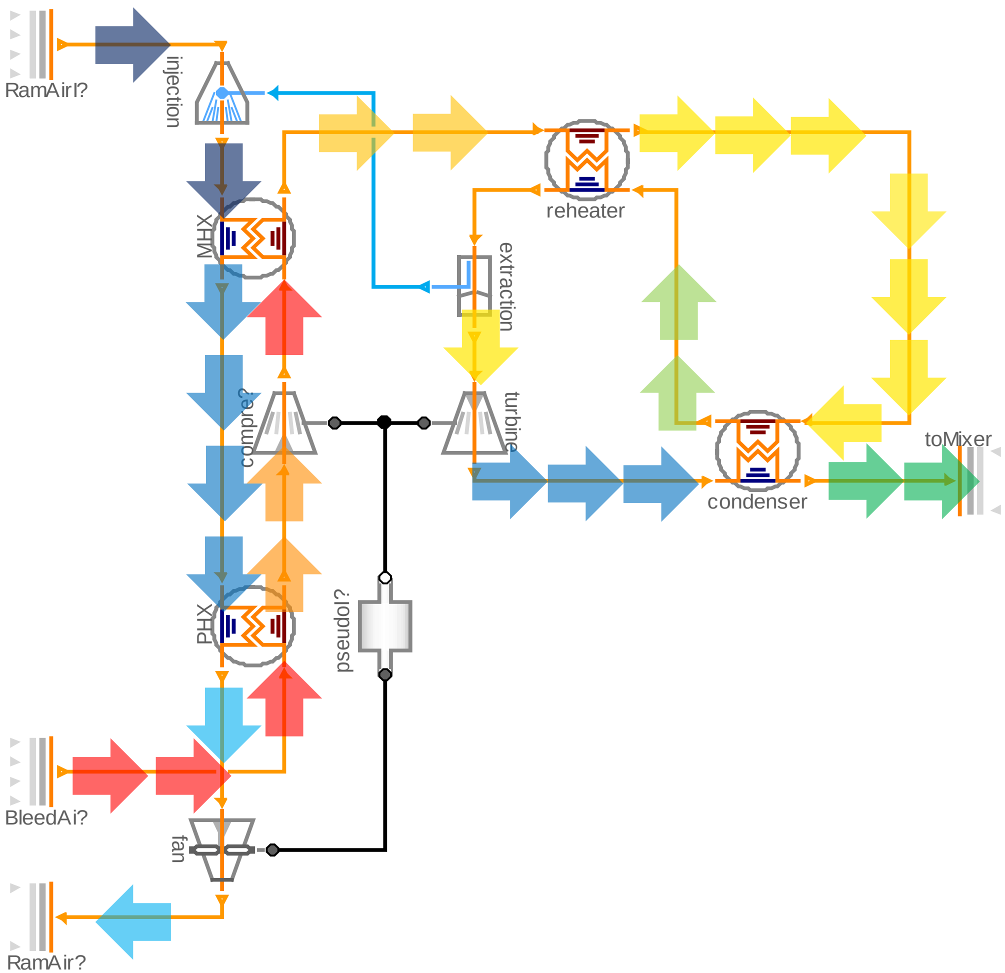
Three Wheel Bootstrap Circuit



- At DLR, we modeled this circuit using Modelica:

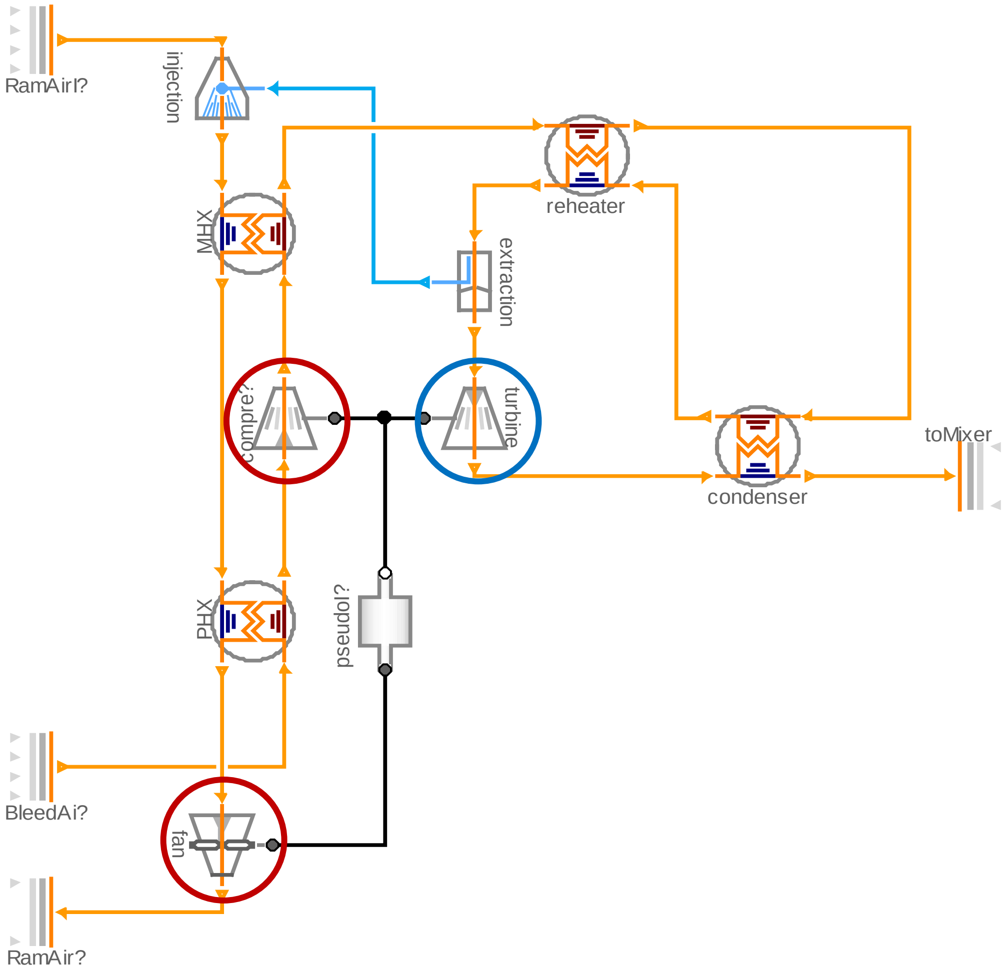
- The ram air channel provides the cooling reservoir

Three Wheel Bootstrap Circuit



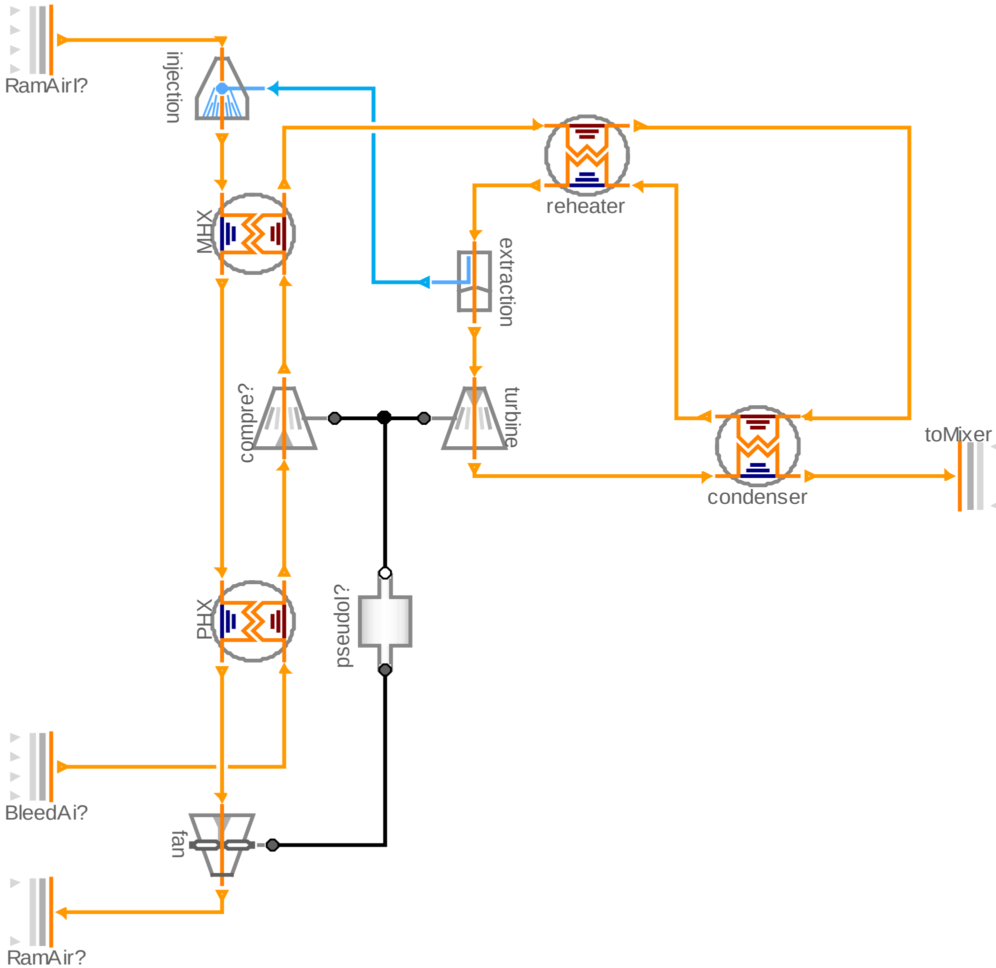
- At DLR, we modeled this circuit using Modelica:
- The ram air channel provides the cooling reservoir
- The air is then cooled and expanded. It passes four heat exchangers

Three Wheel Bootstrap Circuit



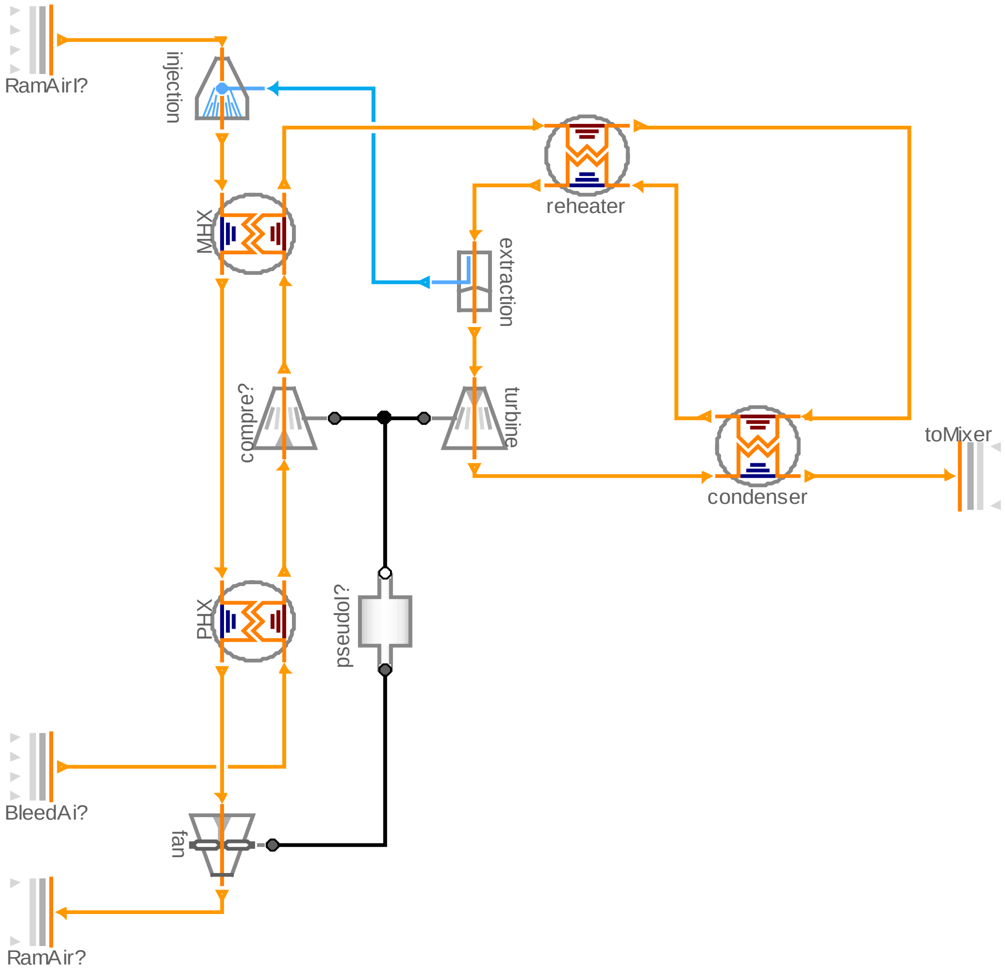
- At DLR, we modeled this circuit using Modelica:
- The **ram air channel** provides the cooling reservoir
- The bleed air is then cooled and expanded. It passes **four heat exchangers**
- The energy gained in **the turbine is used to power compressor and fan** by the drive shaft.

Three Wheel Bootstrap Circuit



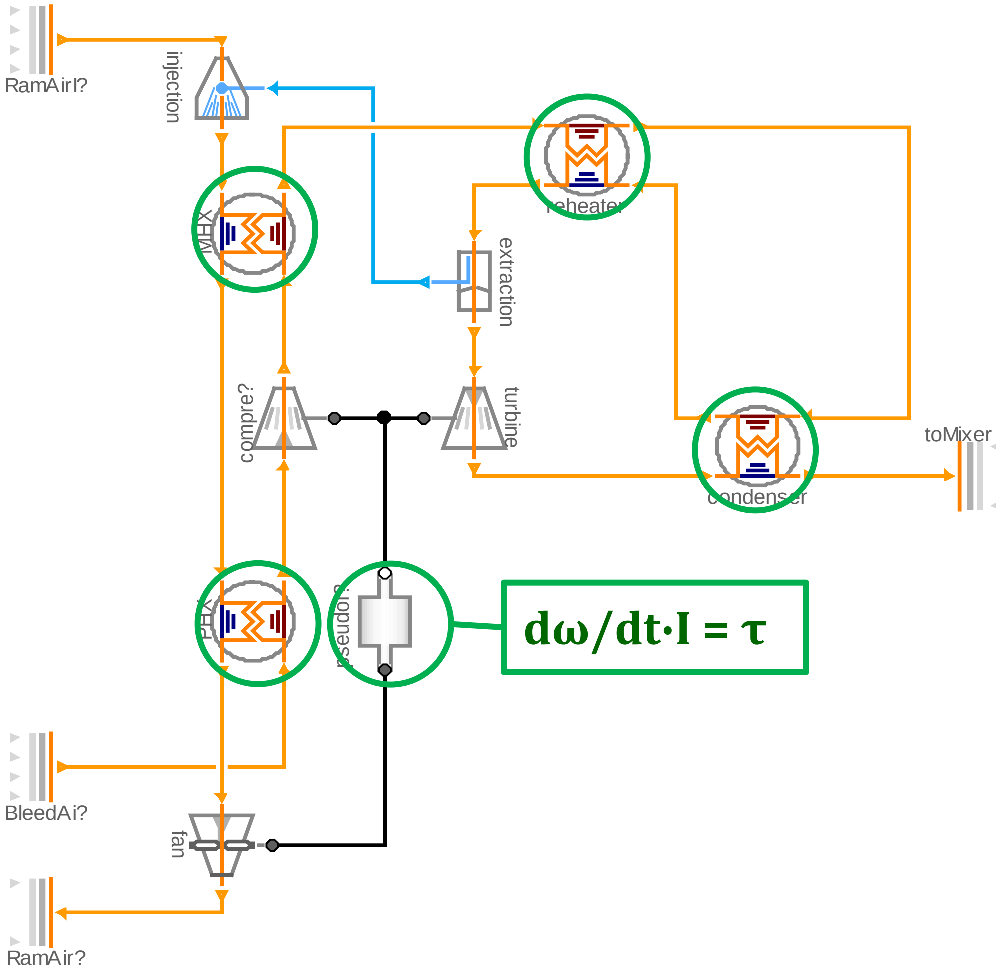
- The model is actually conceived as stateless and models no dynamics.
- Instead we look for the equilibrium point:
- Balance of mechanical energy at the drive shaft
- Balance of thermal energy at the heat exchangers

Three Wheel Bootstrap Circuit



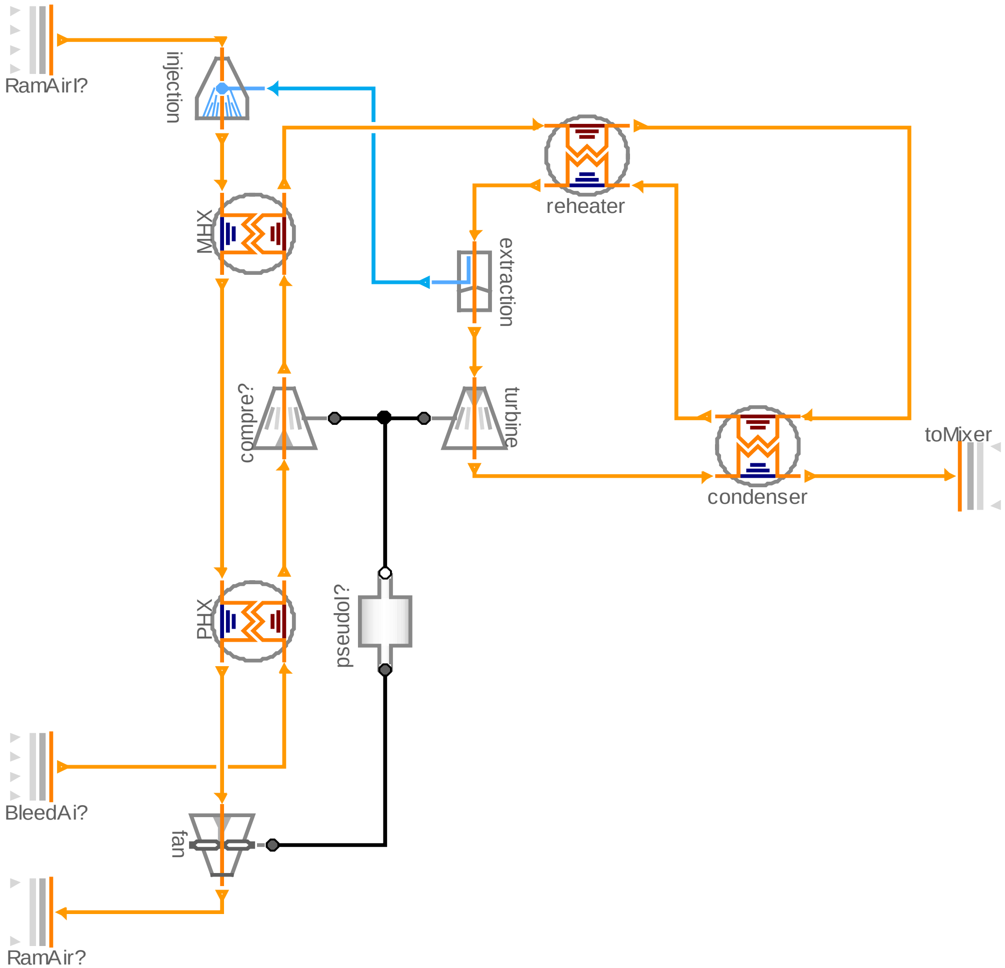
- Formulating these balances leads to a system with more than **200 non-linear equations**
- Dymola tries its best but there remain **more than 40 iteration variables**.
- It is very difficult to find a solution
- It is computationally very expensive

Three Wheel Bootstrap Circuit



- How does physics reach the balance point?
- We add an inertia to the drive shaft.
- We add thermal inertia to the heat exchangers
- So we have added 5 additional states to our system.
- These are **artificial states** since we are not interested in the corresponding dynamcis

Three Wheel Bootstrap Circuit



- With 5 additional states, there remain only single non-linear equations that can be solved sequentially.

- The system can be solved in a robust way.

- Simulation with 5 states is still faster than with a system of 40 iteration variables.

Review: The Method of Artificial States

We have applied a common method to cope with a system of non-linear equations: **The method of artificial states**

- We have torn the system apart by introducing artificial states
- Instead of prescribing the balance law directly, we are now describing how to reach the balance point as quasi steady state solution.
- We can do this, because we know the physics of our system.
- In this way, we abuse time-integration as solver for non-linear systems

Classes of Artificial States

The method of artificial states appears in many disguises

- **Adding storages of energy**
  like micro-capacitances, small inertias, spring-dampers, intermediate volumes for fluids, etc.

- **Adding Controllers**
  Integration-based controllers lead to the equilibrium point

- **Signal Filters**
  used to tear apart algebraic loops.

DLR

Is it Malpractice?

Unfortunately, artificial states involve many disadvantages

- **Stiffness** is added to the system
    - limits step-size
    - impairs real-time capability
    - local problem creates global damage

- Simulation **results are polluted** by artificial dynamics

- **Loss of precision**

- Time-constants are hard to retrieve and result in **fudge parameters**

Are these disadvantages inevitable?

What is the Problem?

When using artificial states, the modeler evidently makes a distinction between

- Dynamic processes that are relevant of the system under study.

- Dynamic processes that describe how to solve a non-linear system of equations.

He is forced to mix up these dynamics since M&S Frameworks provide no means to make a proper distinction

Hence we propose on tool: balance dynamics equations.

Balance Dynamics Equations

- The main idea is to give the modeler a way to express his non-linear system as a result of an idealization.

- When we added the inertia, we used the following differential equation

$$\mathrm{der}(\omega)\cdot I = \tau$$

where I is the fudge parameter.

- Ideally we want $I \rightarrow 0$ and reach the steady-state solution $0 = \tau$ infinitely fast. To express this we use the new balance operator instead of the derivative operator.

$$\mathrm{balance}(\omega) = \tau$$

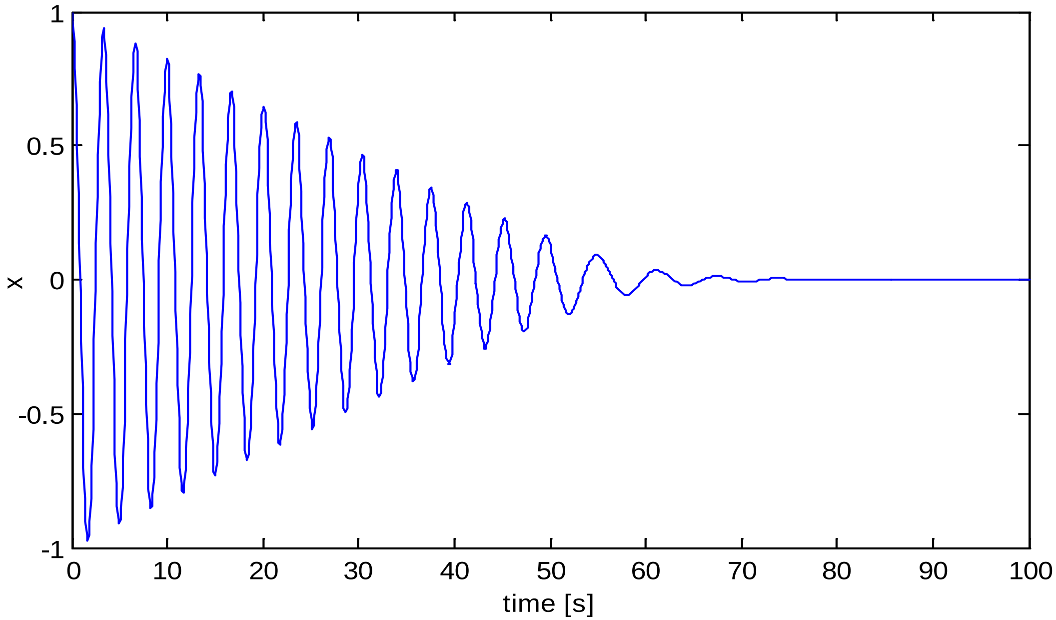
- The fudge parameter is gone

Small Application Example

- Let us simulate the following system:

$$\mathrm{d}x/\mathrm{d}t = y$$
$$\mathrm{d}y/\mathrm{d}t = -0.1 \cdot a - 0.4 \cdot \mathrm{y}$$
$$s(a) = 10 \cdot x$$

- This is the simulation result:



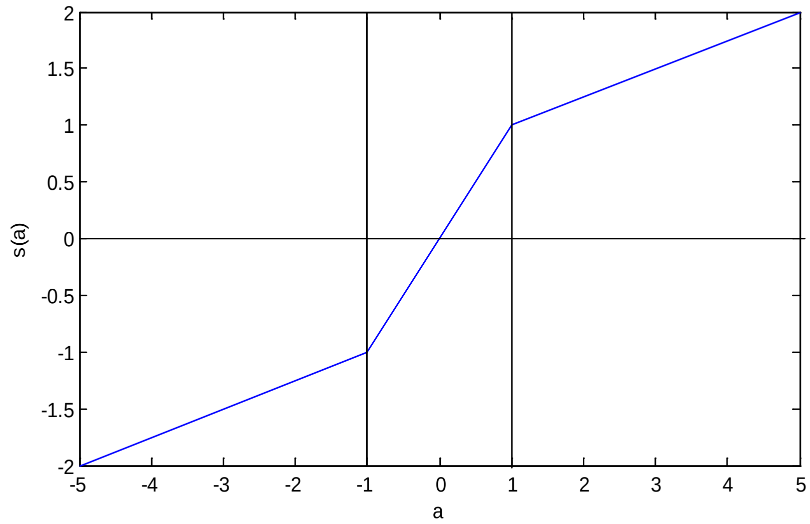
The Problematic Non-linear Equation

- x and y are states and we need to solve the last equation for a with $s(a)$ being defined as:

- for $a < -1$: $s(a) = a/4 - 3/4$

- for $a > 1$: $s(a) = a/4 + 3/4$

- else: $s(a) = a$



- Since the solution by Newton's method is tricky when the start values jumps over 1 (resp. -1), we have to choose a very small step-size (here 0.01s with Heun).

How to Use Balance Dynamics Equations?

- But we know that $s(a)$ is a monotonic increasing function. So we can use a balance dynamics equation to get to the solution.

$$\mathrm{d}x/\mathrm{d}t = y$$
$$\mathrm{d}y/\mathrm{d}t = -0.1 \cdot a - 0.4 \cdot y$$
$$\mathrm{balance}(a) = 10 \cdot x - \mathrm{s}(a)$$

- The balance dynamics equation can be interpreted as control law:

$$\mathrm{der}(a) \cdot \mathrm{T} = 10 \cdot x - \mathrm{s}(a)$$

- If a is too small, a is increased and if a is too high, a is decreased.

How to Interprete Balance Dynamics Equations?

$$\mathrm{d}x/\mathrm{d}t = y$$
$$\mathrm{d}y/\mathrm{d}t = -0.1\cdot a - 0.4\cdot y$$
$$\mathrm{balance}(a) = 10\cdot x - \mathrm{s}(a)$$

- This system can now be interpreted in two ways:

To solve the ODE

$$\mathrm{d}x/\mathrm{d}t = \mathrm{y}$$
$$\mathrm{d}x/\mathrm{d}t = -0.1\cdot a - 0.4\cdot y$$
$$0 = 10\cdot x - \mathrm{s}(a)$$

To solve the non-linear equation

$$x = \mathrm{const}$$
$$\mathrm{d}a/\mathrm{d}t = 10\cdot x - \mathrm{s}(a)$$

- The idea is to use a sub-simulation to solve the non-linear equation and get the solution of the steady-state.

How to Perform such a Sub-simulation?

- If we want to simulate a system $\mathrm{d}\boldsymbol{x} = \mathrm{f}(\boldsymbol{x})$ just to get to the steady state then this is a special case. Which integration method to use?
    - Since the system is supposed to be stable, we take an implicit solver
    - Since the steady state solution is insensitive to the local integration error, an order 1 method is sufficient.
- Hence we end up using Backward Euler. One step of BE:

$$\boldsymbol{x}_{\mathrm{t+h}} = \boldsymbol{x}_{\mathrm{t}} + \mathrm{h}\cdot\mathrm{f}(\boldsymbol{x}_{\mathrm{t+h}})$$

- requires us to solve:

$$\mathrm{g}(\boldsymbol{x}_{\mathrm{t+h}}) = \boldsymbol{x}_{\mathrm{t}} - \boldsymbol{x}_{\mathrm{t+h}} + \mathrm{h}\cdot\mathrm{f}(\boldsymbol{x}_{\mathrm{t+h}})$$

Turning Sub-simulation into a Continuation Method

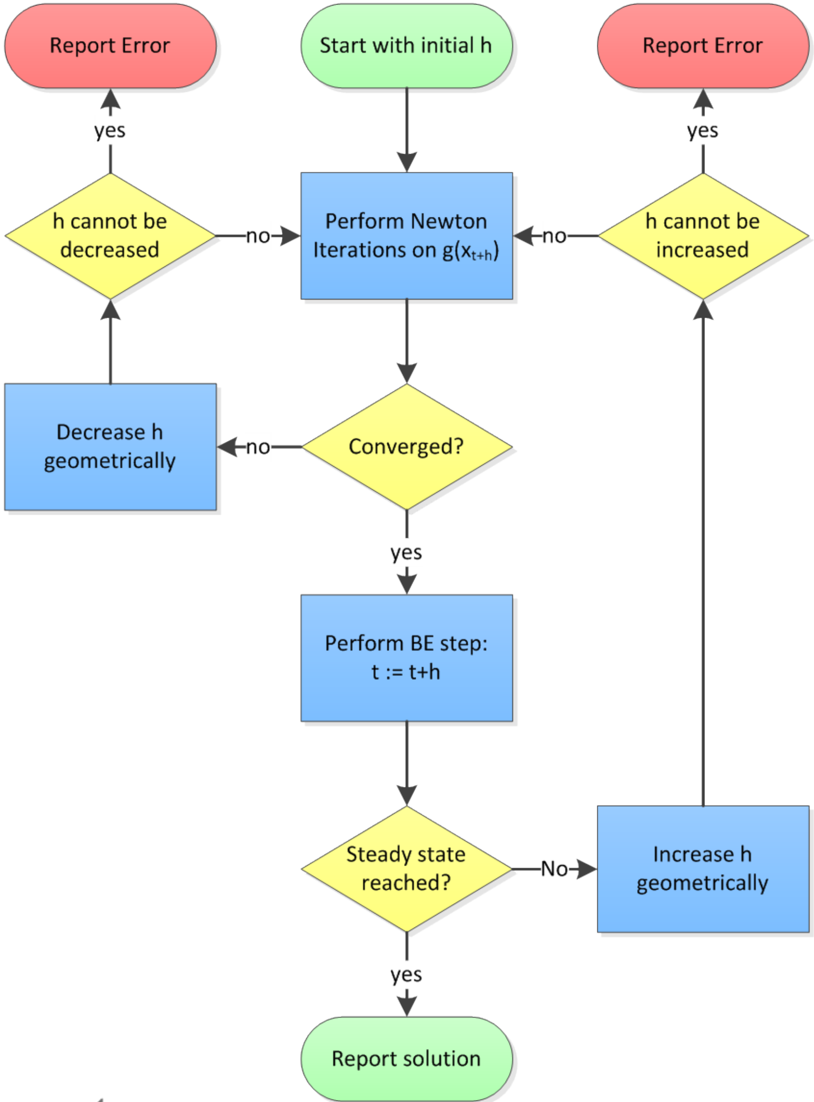
- Evidently for $\mathrm{h} \to \infty$, solving $\mathrm{g}(\boldsymbol{x}_{\mathrm{t+h}})$ becomes equivalent to solving $\mathrm{f}(\boldsymbol{x})$ directly.

- But we have won one important degree of freedom: we can choose h and there will always be an h small enough to stay in the convergence interval

- In this way, we have transformed the problem into a numerical continuation problem (as used in homotopy solvers)

- We can adapt the natural parameter continuation for our purposes

Continuation Solver for Balance Dynamics



- This algorithm becomes part of the main simulation loop and replaces the former direct solver for $0 = f(\boldsymbol{x})$

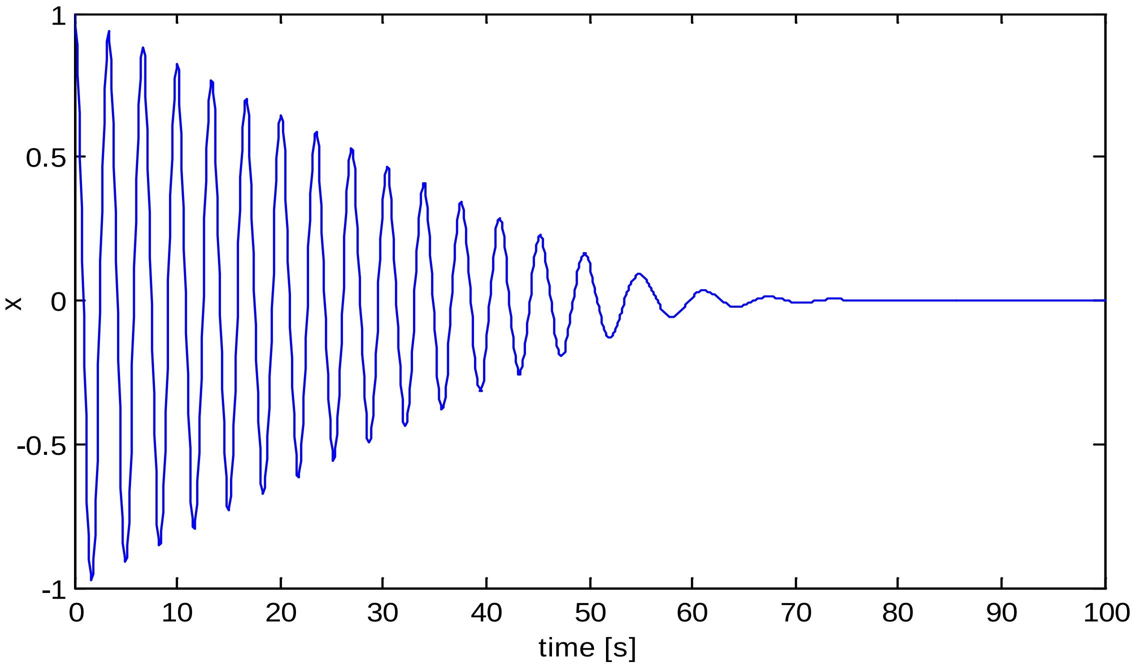
- The continuation method wraps Newton's method and thereby adds robustness

- Having a good initial guess and a good initial value for h, the overhead of the continuation solver is low.

- Let us apply this solver to our application example.

Application Example

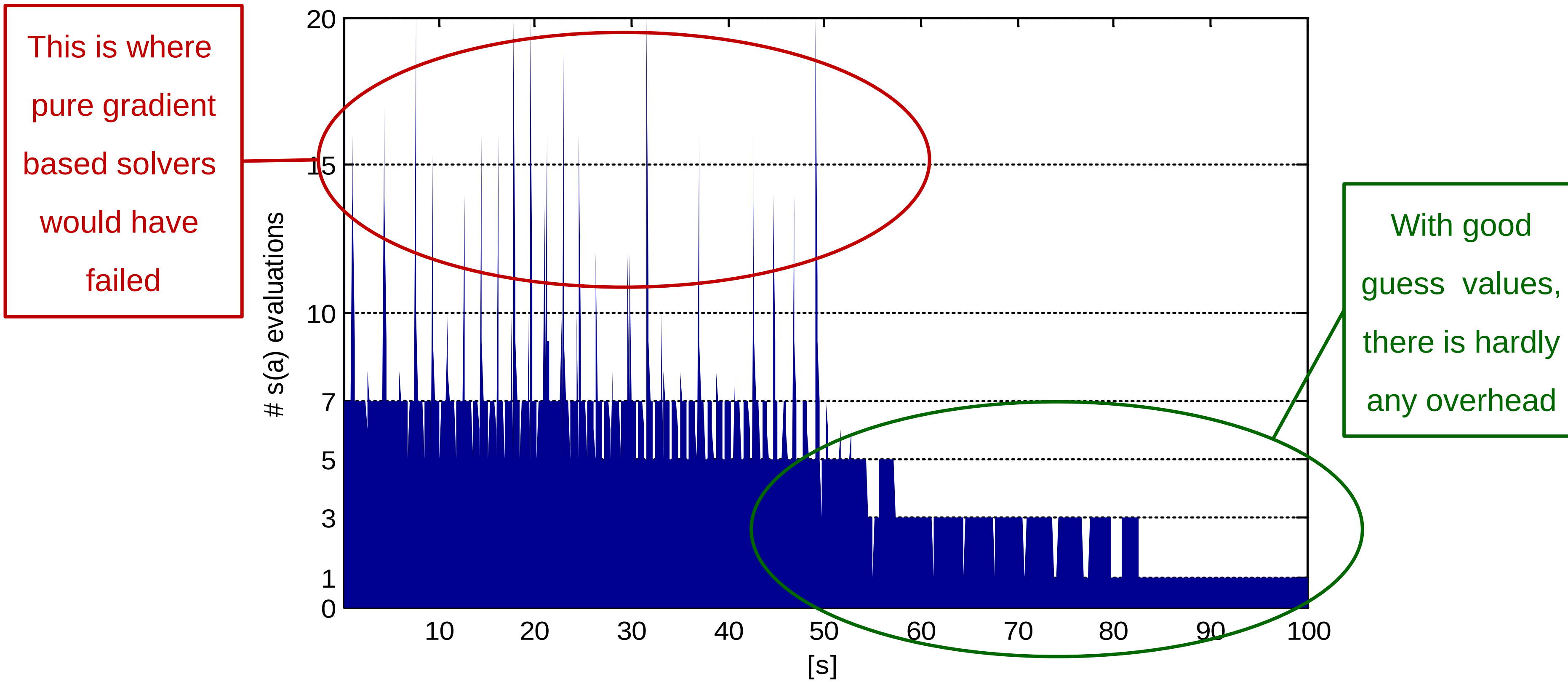
- Remember: this was the simulation result.



- With balance dynamics, we can afford to take much larger steps (here 0.1s with Heun but much larger is possible)

Application Example

- This diagram presents the number of function calls of $s(a)$ in each integration step using our continuation solver:



Conclusions

- Balance dynamics equations provide an elegant way to incorporate vital knowledge on how to solve systems of non-linear equations.

- Modelers can now apply the method of artificial states without having a bad conscience.

- Solving non-linear systems is still not for free but at least we can avoid creating a global damage to our system. The problem is kept local.

- The proposed operator is not the ultimate answer but it is good enough to continue the examination of this method.

What Needs to be Done?

- We need test implementations (for instance in Modelica tools) so that we can apply this method to more complex and realistic examples.

- We at DLR cannot do all of this work, so we are looking for people wanting to join this research task.

- There remain a number of interesting research question that wait to be answered:

    - How to generate code for balance dynamics solver?
    - How to best implement a continuation solver?
    - How to design the language for balance dynamics?

Questions?