Initialization of Thermofluid models

Francesco Casella

(francesco.casella@polimi.it)

Dipartimento di Elettronica e Informazione

Politecnico di Milano

OpenOffice.org

Introduction

- Initialization is critical for models with dynamics
  - solution of ODE/DAE depends on initial conditions
  - if consistent initial conditions cannot be found, the model is useless
- Thermofluid models often strongly nonlinear
- Steady-state initialization involves solving large, strongly nonlinear systems of equations
  - sensitivity to initial guess values
  - sensitivity to selection of iteration variables
  - tracing root of problem difficult
    - guess values?
    - errors in the model (e.g. wrong parameters)?
- Troubleshooting is not user friendly
  - understanding of the symbolic/numeric methods
  - understanding the equations of the original model (not encrypted!)
  - equations are rearranged / symbolically solved
  - no avaliable Modelica tool currently gives user friendly GUI support
  - unpredictable time-to-go

Specifying initial conditions - defaults

- Each variable in Modelica has a `start` attribute

```
model SimpleSystem
  parameter Time tau;
  parameter Real x_start = 2;
  Real x(start = x_start);
  Real u;
equation
  u = if time < 1 then 1 else 2;
  tau * der(x) = - x^2 + u;
end SimpleSystem
```

- By default, Modelica tools use start values to assign initial values to state variables
- It is possible to log these additional initial equations
  (Simulation | Setup | Translation | Log selected default initial conditions)

Specifying initial conditions – fixed attribute

- Setting the fixed attribute to true one indicates explicitly that the initial value of the variable is the initial value

```
model SimpleSystem2
  parameter Time tau;
  parameter Real x_start = 2;
  Real x(start = x_start, fixed = true);
  Real y;
  Real u;
equation
  u = if time < 1 then 1 else 2;
  tau * der(x) = - x^2 + u;
  y = 3*x^2;
end SimpleSystem2
```

Specifying initial conditions – initial equations

- In Modelica 1.x, initialiation was mainly thought as a tool issue
- Dymola had a run-time interface to specify initial conditions a define free and fixed variables (using data from fixed and start attributes)
- Since Modelica 2.0 (2002): explict initial equations with arbitrary structure (the old fixed = true mechanism is still available)

```
model SimpleSystem
  parameter Time tau;
  parameter Real x_start = 2;
  Real x(start = x_start);
  Real y;
  Real u;
equation
  u = if time < 1 then 1 else 2;
  tau * der(x) = - x^2 + u;
  y = 3*x^2;
initial equation
  x = x_start;
end SimpleSystem
```

```
model SimpleSystem
  parameter Time tau;
  parameter Real x_start = 2;
  Real x(start = x_start);
  Real y;
  Real u;
equation
  u = if time < 1 then 1 else 2;
  tau * der(x) = - x^2 + u;
  y = 3*x^2;
initial equation
  der(x) = 0;
end SimpleSystem
```

- Initial equations are combined with dynamic equations
- Nonlinear solvers use start attribute for guess values

Specifying initial conditions – Unknown parameters

- Parameters can be freed and determined by further initial equations (beware of initial equation count...)

```
model SimpleSystem
  parameter Time tau;
  parameter Real x_start = 2;
  parameter Real y_start = 10;
  parameter Real u_start(fixed = false, start = 2);
  Real x(start = x_start);
  Real y;
  Real u;
equation
  u = if time < 1 then u_start else u_start + 1;
  tau * der(x) = - x^2 + u;
  y = 3*x^2;
initial equation
  der(x) = 0;
  y = y_start;
end SimpleSystem
```

- Fixed attributes can be changed at instantiation (Show Component feature in the Dymola GUI)

```
System S(u_start(start = 2, fixed = false),
         y(start = 3, fixed = true));
```

Meaningful initial conditions for thermofluid systems

- In some cases, the initial state of a thermofluid system conveniently expressed by giving the values of initial states
- Example: direct steam generation solar plant just before dawn
  - temperature distribution in the collector pipes = ambient temperature
  - given temperatures in the storage volumes, depending on length of shutdown period
  - zero (or negligible) flows (beware of singular conditions)

- In other contexts, starting from an “off-state” is problematic/not required
- Example: power plant model for load change or primary frequency studies
  - range of validity 30-100%
  - piping and instrumentation for startup not included in the model (turbine bypass systems, steam vents, etc.)
- In these cases, the typical transient is the response to a disturbance starting from an equilibrium state (steady-state)
- Fixing approximated initial states can lead to unphysical transients
  - out-of-range mass & heat flow values
  - flow reversal
  - trip conditions
  - ...

Quick review of Modelica model transformations

- Steps of model transformation
  - flattening
  - index reduction
  - elimination of trivial equations / alias variables
  - BLT transformation
    - dynamic problem equations (initial values and parameters)
    - initialization problem (derivatives, algebraic variables)
  - Tearing applied to implicit systems of equations to reduce the number of iteration variables
- Equations solved as explicit assignments → no problem
- Implicit linear equations → solved without problem if non-singular
- Implicit nonlinear equations → iterative methods → reasonable guesses required for the iteration variables
- Guess values provided in Modelica by start attributes
  - defined by types
  - defined by modifiers in library models, possibly through parameters
  - defined by direct modifiers on the simulation model

Quick review of Modelica model transformations (cont'd)

- Guess values provided in Modelica by start attributes
- Defined by types

```
type Temperature = Real(unit = 'K', start = 300);
```

- Defined by modifiers in library models, possibly through parameters

```
SI.Pressure p_start;
SI.Pressure p(start = p_start);
```

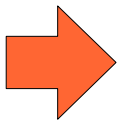
- Defined by direct modifiers on the simulation model

```
model PlantWithStartAttributes
  extends Plant(
    turbine(p_in(start = 1.27e8)),
    drumHP(p(start = 1.32e8)));
end PlantWithStartAttributes;
```

Tearing / Alias variables

- A small fraction of model variables require a guess value to ensure solver convergence
- ~60% alias variables (x = ± y)
- ~30% assigned in assignment section of torn systems

- Selection of alias and iteration variables in torn systems not unique!
- Alias variable selection in Dymola:
  - highest priority to direct modifiers,
  - intermediate priority to modifiers in libraries
  - lowest priority to type-defined start values
- Tearing variable selection
  - optimal selection: NP-complete problem
  - proprietary heuristics
  - changes w/ small modifications of model
  - changes w/ tool version



Not really object oriented!

Understanding initialization failures in Dymola

- Interactive demo

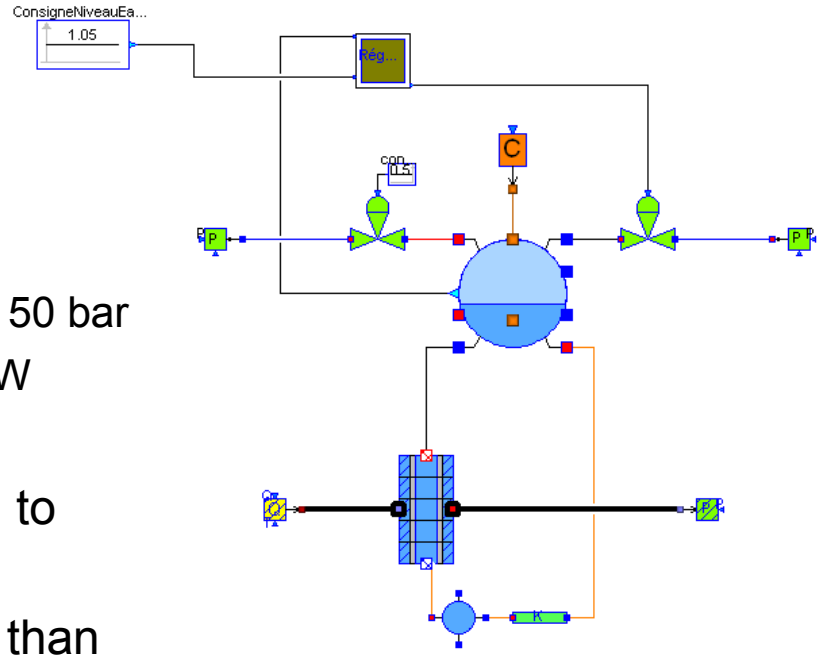
- dsmodel.mof: how to generate, how to read
- dslog.txt: how to generate, how to read

Strategies for steady-state initialization

- Divide and conquer
  - break plant in smaller modules
  - initialize each model w/ suitable boundaries
  - make sure each model is correctly parameterized
- Check dsmodel.mof/dslog.txt and improve start values
- Fix initial states, run transient until equilibrium
  - requires asymptotically stable equilibrium
    (if not, use simple feedback controllers to stabilize)
  - transient might go wrong because of abnormal mass and energy flows due to unphisical values of initial states
  - example: evaporator with high heat transfer coefficient
    - small error in delta-T fluid-wall → large error in heat flow → condensation/evaporation → bad flow rates (possibly flow reversal)
- Important remark: if the parametrization of the model is incorrect, a steady-state solution might not exist!

An example of model without steady-state solution

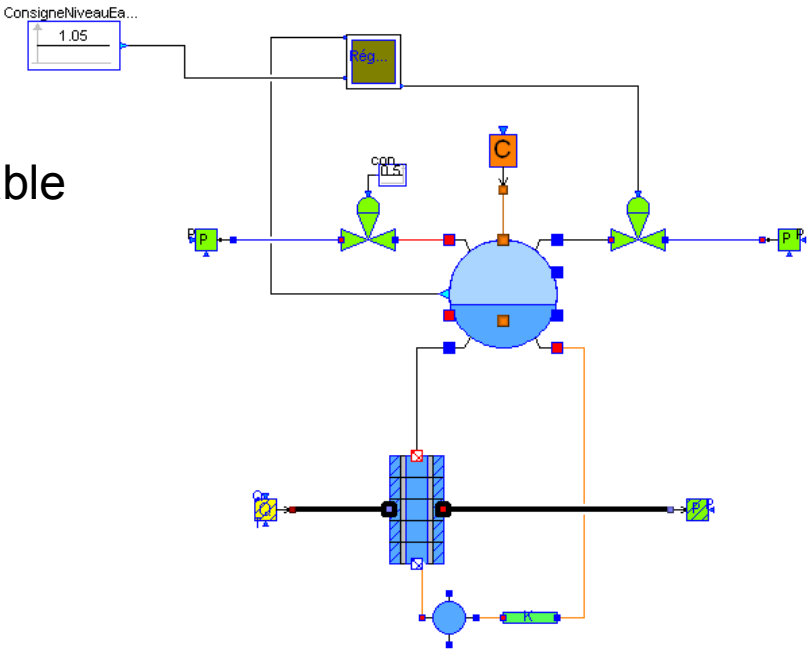
- Natural circulation steam generator
- Nominal operating data:
  - drum pressure 127 bar
  - feedwater pressure 137 bar
  - steam valve outlet pressure 50 bar
  - power from flue gas: 100 MW
  - steam flow: 80 kg/s
- Steam flow rate roughly proportional to steam valve Cv and drum pressure
- Assume now Cv is just 10% smaller than the correct value
  - in order to evacuate 80 kg/s drum pressure should be 140 bar
  - feedwater cannot flow from source at lower pressure!



Solution does not exist!

Troubleshooting

- Set initial values of pressures and enthalpies close enough to nominal operating point
- The system has an asymptotically stable equilibrium thanks to level controller
- Start the simulation



- Drum pressure increases steadily
- Steam flow << feedwater flow
- → Too small steam valve
- → Increase steam valve Cv

- To find the correct Cv:
  - set Cv(fixed = false)
  - set steam_flow(start = 80, fixed = true)

Homotopy-based initialization: Motivations

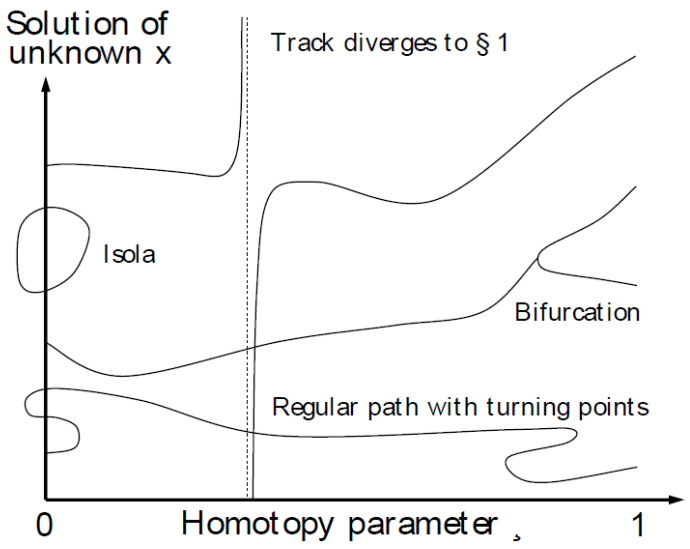
- Setting start values
  - tedious
  - selection of iteration variables can change
  - how to determine good enough values?
  - not object oriented
  - not robust
  - requires high-end skills to troubleshoot models

- A more robust method required
  - guarantee convergence
  - use only well-known design data
  - reasonable defaults for iteration variables
  - no need of directly setting start values
  - object-oriented

Homotopy-based initialization

Homotopy-based initialization: basic ideas

- Simplified problem solved first
  - same variables, slightly modified equations
  - close enough to the actual one
  - steady-state (otherwise structure can change too much!)
  - simpler to solve ("less nonlinar" → less sensistive to guess values)
  - guess values possibly provided by default
- Solve convex combination of simplified and actual problem
- Caveat: the problem changes continuously, the solution might not
  - turning points (can be handled by specialized solvers)
  - asymptotes (failure)
  - bifurcations (requires interaction with expert user)
- Specification in Modelica 3.2: homotopy operator
  - `homotopy(actual, simplified)`
  - during solution of initialization returns λ*actual + (1 – λ) *simplified



Simplified models for thermofluid systems

- Linear momentum balance equations with constant coefficients based on nominal operating point
  - pressure loss components $w = w_{nominal} / \Delta p_{nominal} \cdot \Delta p$
  - static head term $\Delta p_{static} = \rho_{nominal} \cdot gH$
  - steam turbine $w = w_{nominal} / p_{nominal} \cdot p_{inlet}$
  - pump: linearized characteristic around nominal operating point

```
// Pressure loss component
pin - pout = homotopy(smooth(1, Kf*squareReg(w,wnom*wnf))/rho,
                      dpnom/wnom*w) "Flow characteristics";

// Valve for incompressible fluid
w = homotopy(FlowChar(theta)*Av*sqrt(rho)*sqrtR(dp),
             theta/thetanom*wnom/dpnom*dp);

// Pump
function df_dq = der(flowCharacteristic, q_flow);
head = homotopy(
  (n/n0)^2*flowChar(q*n0/(n+n_eps)),
  df_dq(q0)*(q-q0) + (2/n0*flowChar(q0) - q0/n0*df_dq(q0))*(n-n0) + head0);

// Turbine
w = homotopy(Kt*partialArc*sqrt(p_in*rho_in))*sqrtReg(1-(1/PR)^2),
             wnom/pnom*p_in);
```

Simplified models for thermofluid systems (cont'd)

- Energy balance equation: nonlinear wh terms
  - use nominal flow rate for energy balance in 1D models for heat exchangers
  - gets rid of most wh terms in a typical model
- Upstream specific enthalpy depending on flow direction
  - assume design direction of flow

```
hi = homotopy(if not allowFlowReversal then inStream(inlet.h_outflow)
                                       else actualStream(inlet.h_outflow),
              inStream(inlet.h_outflow));
```

- Flow-dependent heat transfer coefficients
  - assume nominal h.t.c. value

```
wall.gamma[j] = homotopy(gamma_nom*noEvent(abs(infl.m_flow/wnom)^kw),
                         gamma_nom);
```

- Temperature-specific enthalpy relationship
  - Not too nonlinear in most cases (save phase changes)
  - Do not change (difficult to make it consistent across models)
  - Set rough start values
    - liquid / two-phase / vapour for water
    - within 100-200 K for ideal gases

Simplified models for thermofluid systems (cont'd)

- Controllers acting on flows and controlling energy-related quantities
  - may introduce strong nonlinear coupling between hydraulic and thermal equations
  - simplified model: keep control variable fixed at nominal value (opened feedback loop)
- Controllers with saturations and anti-windup
  - Saturations are strong nonlinearities
  - If known to be in linear regime: remove saturations
  - Else: fix at 0% or 100%
- Control valves
  - Assume flow is nominal flow * opening (regardless of density and pressure drop)
- Initialization not at nominal operating point
  - Simplified model has all setpoints at nominal values
  - Homotopy brings the setpoints to the desired value
  - Different from relaxation transient: quasi-static transformation (less problems if initial guess not accurate – no weird transients)

Performing the homotopy transformation

- Suitable continuation solvers can be used
  - can track turning points
  - efficient interpolation-extrapolation
- Ideally
  - no turning points
  - no other singularities
- Singularities might arise if wrong parametrization
  - split system into subsystems
  - divide and conquer

Homotopy-based initialization: tool support

- Successful experiments at Politecnico and EDF
  - experimental solver linked to Dymola
  - by M.Sielemann, based on NOX/LOCA solver
  - definitely not user-friendly
- Small-size use cases
  - multibody systems with multiple steady-state configurations
  - analog electronic circuits
  - hydraulicciruits
  - air conditioning systems
- Large-sized use cases
  - complete combined-cycle power plant models
  - up to 670 iteration variables (!)
  - no need of manual setting of start values
- Preliminary implementation in Dymola 7.5 (not good enough yet)
- Hopefully available soon as standard feature in all Modelica tools

DISCUSSION

References

- M. Sielemann, F. Casella, M. Otter, C. Clauss, J. Eborn, S.E. Mattsson, H. Olsson: Robust Initialization of Differential-Algebraic Equations Using Homotopy. Proceedings of 8th International Modelica Conference, Dresden, Germany, 20-22 March 2011.
- F. Casella, M. Sielemann, L. Savoldelli: Steady-state Initialization of Object-Oriented Thermo-Fluid Models by Homotopy Methods, Proceedings of 8th International Modelica Conference, Dresden, Germany, 20-22 March 2011.