The Modelica.Fluid library

Francesco Casella
(francesco.casella@polimi.it)

Dipartimento di Elettronica e Informazione
Politecnico di Milano
Brief history

• Development started around 2002 (?) as Modelica_Fluid

• Goal: become part of the MSL for basic support to thermofluid system modelling

• Connector concept based on coupled flow/effor variables for energy (and partial mass) balances not completely satisfactory for numerical reasons

• Design of the Modelica_Fluid connectors (→ affecting the whole library!) swunged back and forth many times

• Eventually (2009): definition of stream connectors in Modelica 3.1

• Modelica.Fluid becomes part of MSL 3.1 (Aug. 2009)
Goals and scope

• Support the modelling of 0D-1D thermofluid systems with purely convective heat transport across ports
  – thermal power plants (fossil-fired, biomass, solar, nuclear)
  – heating systems
  – air conditioning and ventilation systems
  – no thermal conduction across ports (liquid metals at low flows)
  – no gasdynamics (supersonic flows, shock phenomena etc.)

• Use Modelica.Media for medium property computations
• Define common interfaces for cross-library compatibility
• Provide most commonly used components (sources, valves, pumps, …)

• Provide a wide range of ready-made components?

• Discussion
FluidPort connector

- Applicability:
  - purely convective heat and mass transport, (no heat conduction nor mass diffusion across ports)
  - one or two phases
  - one or more substances

- Discussion on the meaning of the variables

```modelica
connector FluidPort
  replaceable package Medium = Modelica.Media.Interfaces.PartialMedium
    "Medium model";
  flow Medium.MassFlowRate m_flow
    "Mass flow rate from the connection point into the component";
  Medium.AbsolutePressure p
    "Thermodynamic pressure in the connection point";
  stream Medium.SpecificEnthalpy h_outflow
    "Specific enthalpy close to connection point if m_flow < 0";
  stream Medium.MassFraction Xi_outflow[Medium.nXi]
    "Independent mixture mass fractions close to connection point if m_flow < 0";
  stream Medium.ExtraProperty C_outflow[Medium.nC]
    "Properties c_i/m close to the connection point if m_flow < 0";
end FluidPort;
```

- Only specific enthalpy discussed in the following for simplicity
Stream variables – First step

• Simple fluid port design – no flow reversal allowed

connector FluidPortA “Port for entering flow”
  flow MassFlowRate m_flow “Flow into connector”;
  AbsolutePressure p “Thermodynamic pressure at the connector”;
  input SpecificEnthalpy h “Specific enthalpy of incoming fluid”;
end FluidPortA;

connector FluidPortB “Port for outgoing flow”
  flow MassFlowRate m_flow “Flow into connector”;
  AbsolutePressure p “Thermodynamic pressure at the connector”;
  output SpecificEnthalpy h “Specific enthalpy of outgoing fluid”;
end FluidPortB;

• Limitations:
  – no support for flow reversal
  – only one FluidPortB allowed in the connection set
  – explicit mixing junctions required
Stream variables – Second step

• ThermoPower connector design – flow reversal allowed

    connector FluidPortA “Type-A port”
        flow MassFlowRate m_flow “Flow into connector”; 
        AbsolutePressure p “Thermodynamic pressure at the connector”;
        output SpecificEnthalpy hAB “Specific enthalpy of outgoing fluid”;
        input SpecificEnthalpy hBA “Specific enthalpy of incoming fluid”;
    end FluidPortA;

    connector FluidPortB “Type-B port”
        flow MassFlowRate m_flow “Flow into connector”; 
        AbsolutePressure p “Thermodynamic pressure at the connector”;
        input SpecificEnthalpy hAB “Specific enthalpy of incoming fluid”;
        output SpecificEnthalpy hBA “Specific enthalpy of outgoing fluid”;
    end FluidPortB;

• Limitations:
    – only one-to one connections allowed
    – explicit mixing junctions and flow splitters required
    – two complementary ports required with the same semantics
Stream variables – The final idea

- Stream variables: specific properties transported by the flow variable via purely convective transport
- The stream variable describe the property of outgoing fluid, *irrespective of the actual direction of the flow* (i.e. assuming m_flow < 0)
- Same role as the output variables in the previous designs
- No connection equations are generated

```plaintext
cconnector FluidPort “Generic fluid port”
    flow MassFlowRate m_flow “Flow into connector”;
    AbsolutePressure p “Thermodynamic pressure at the connector”;
    stream SpecificEnthalpy h_outgoing “Specific enthalpy of outgoing fluid”;
end FluidPort;
```

- Values of stream variables for incoming flow obtained via *operators*:
- **inStream(v):** value of v assuming entering flow (m_flow > 0) *irrespective of actual flow direction*
- Same role as input variables in the previous designs
- **actualStream(v):** actual value of v inside the component close to the interface, depending on flow directions

```plaintext
actualStream(port.h_outflow) = if port.m_flow > 0
    then inStream(port.h_outflow)
    else port.h_outflow;
```
Definition of `inStream()`

- Assume N fluid connectors mj.c are connected together
- Assume only inside connections for simplicity (for the general case: see Modelica Specification)
- *For each port*, `inStream(mj.c.h_outflow)` is the mixing quantity at the connection point *assuming entering flow*
- `inStream(mj.c.h_outflow)` is different at each port j

- **Declarative definition:**

  \[
  \text{inStream(mi.c.h_outflow)} = h_{\text{mix ini}}; \\
  0 = \text{sum(mj.c.m_flow for j in 1:N)}; \\
  0 = \text{sum(mj.c.m_flow*} \\
  \text{(if mj.c.m_flow > 0 or j==i then h_{mix ini} else mj.c.h_outflow for j in 1:N});}
  \]
Definition of inStream() - cont'd

• Solution (might need regularization in 0/0 cases):

\[
inStream(mi.c.h_outflow) := \\
\frac{\sum(\max(-mj.c.m_flow, 0) \times mj.c.h_outflow \text{ for } j \text{ in } \text{cat}(1, 1:i-1, i+1:N))}{\sum(\max(-mj.c.m_flow, 0) \text{ for } j \text{ in } \text{cat}(1, 1:i-1, i+1:N))};
\]

• Note: does not become singular when \( mi.c.m_flow = 0 \)
• Note: terms corresponding to ports with \( m\_flow.min = 0 \) (flow never goes out of port) can be removed a priori
Suggested implementation

• The basic one-to-one case corresponds to the ThermoPower design
• In simpler cases 0/0 indeterminacy can be removed symbolically
• When N>2 and all flows go towards zero, regularization introduced to avoid 0/0
• Setting attribute min = 0 to the flow variable simplifies the computation when flow reversal support is not required

\[ N = 1 \] (unconnected port)
\[ \text{inStream}(m1.c.h\_outflow) = m1.c.h\_outflow; \]

\[ N = 2 \] (one-to-one connection):
\[ \text{inStream}(m1.c.h\_outflow) = m2.c.h\_outflow; \]
\[ \text{inStream}(m2.c.h\_outflow) = m1.c.h\_outflow; \]

All other cases:
\[ \text{if } mj.c.m\_flow.min \geq 0 \text{ for all } j = 1:N \text{ with } j \neq i \text{ then} \]
\[ \text{inStream}(mi.c.h\_outflow) = mi.c.h\_outflow; \]
\[ \text{else} \]
\[ si = \text{sum(max}(-mj.c.m\_flow, 0) \text{ for } j \in \text{cat}(1, 1:i-1, i+1:N)); \]
\[ \text{inStream}(mi.c.h\_outflow) = \]
\[ \text{sum(positiveMax}(-mj.c.m\_flow, si)\*mj.c.h\_outflow)/\text{sum(positiveMax}(-mj.c.m\_flow, si)) \]
\[ \text{for } j \in 1:N \text{ and } i \neq j \text{ and } mj.c.m\_flow.min < 0 \]
Representative models

model CV “Control volume with mass and energy storage”
    FluidPort pa, pb;
    ...
    equation
        dM_dP*der(p) + dM_dh*der(h) = pa.m_flow + pb.m_flow;
        dE_dP*der(p) + dE_dh*der(h) = pa.m_flow*actualStream(pa.h_outflow) +
                                     pb.m_flow*actualStream(pb.h_outflow);
    pa.p = p;
    pb.p = p;
    pa.h_outflow = h;
    pb.h_outflow = h;
end CV;

model FM
    FluidPort pa, pb;
    ...
    equation
        pa.m_flow = f(pa.p - pb.p, rho);
        rho = f_r(pa.p, pb.p, ha, hb, dp_small);
        ha = inStream(pa.h_outflow);
        hb = inStream(pb.h_outflow);
        pa.m_flow + pb.m_flow = 0;
        pb.h_outflow = inStream(pa.h_outflow);
        pa.h_outflow = inStream(pb.h_outflow);
end FM;
Physical meaning of FluidPorts

- FluidPorts corresponds to an infinitesimally short pipe protruding from the component
- Allows hierarchical and device-oriented modelling without ambiguities
- Beyond mandatory CV-FM-CV structure
- CV-CV connections are allowed
  - same pressure (index reduction)
  - different enthalpy/temperature (infinitesimal pipe in-between)
- CV with two FMs connected to same port
  - additional algebraic equations created to describe flow pattern like this
  - desired semantics might be different
- Solution in Modelica.Fluid + Dymola
  - vector of ports
  - the GUI automatically connects to first free element and increases nPorts

```modelica
parameter Integer nPorts=0 "Number of ports"
annotation(Evaluate=true, Dialog(__Dymola_connectorSizing=true);

VesselFluidPorts_b ports[nPorts](redeclare each package Medium = Medium) "Fluid inlets and outlets";
```
The System Object

All models require an outer system model (like the MultiBody World) containing system defaults (can be overridden locally)
Mathematical structure of typical cases

• **dynCV-dynFM-dynCV**
  – no problems at initialization if initial states fixed
  – wave dynamics, might trigger fast and persistent oscillations

• **dynCV-FM-dynCV**
  – no problems at initialization if initial states fixed

• **FM-FM**
  – nonlinear algebraic equations, possible problems at initialization

• **CV-CV**
  – index reduction (same pressure)
  – possibly nonlinear algebraic equations as a result

• **Three-way connections**
  – nonlinear algebraic equations (enthalpy changes @ flow reversal)
  – can be removed by setting min attribute on connector flow variables (allowFlowReversal parameter w/ global default)
Components with replaceable Media

- Generic components: defined for a class of medium models, specified by interface (add Modelica code)

```modelica
replaceable package Medium =
    Modelica.Media.Interfaces.PartialMedium "Medium in the component";

replaceable package Medium =
    Modelica.Media.Interfaces.PartialTwoPhaseMedium
        "Medium in the component";
```

- Need to redeclare medium on all elements of a circuit (can be done through GUI)

```modelica
    redeclare package Medium = Modelica.Media.Water.StandardWater);
```

- Components with default concrete medium:

```modelica
    constrainedby Modelica.Media.Interfaces.PartialMedium
        "Medium in the component";
```

- Medium is used to define type of connector variables → automatic check of inconsistencies.

- Automatic medium propagation requires type inference (Modelica 4?)