Omola models for Heat Recovery Steam Generators

Holmberg, Ulf

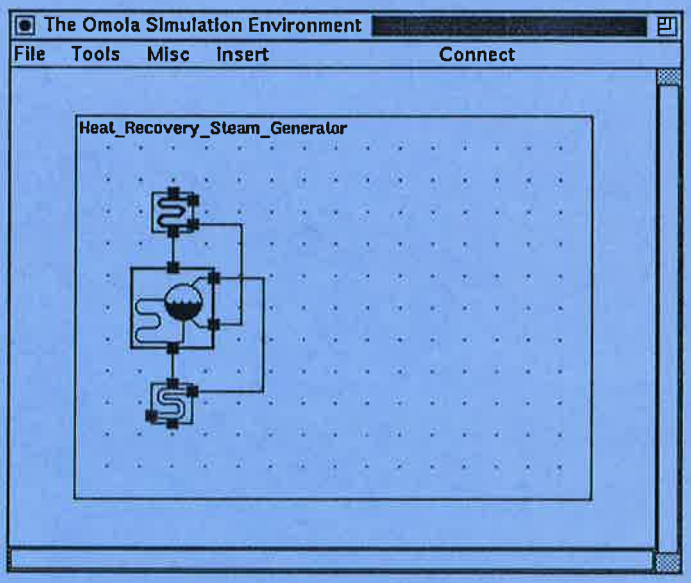
1992

Document Version:
Publisher's PDF, also known as Version of record

ISRN LUTFD2/TFRT--7485--SE

Omola Models For Heat Recovery Steam Generators

Ulf Holmberg



Department of Automatic Control, Lund Institute of Technology

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
INTERNAL REPORT

Date of issue
April 1992

Document Number
ISRN LUTFD2/TFRT--7485--SE

Author(s)
Ulf Holmberg

Supervisor

Sponsoring organisation
Sydkraft

Title and subtitle
Omola Models For Heat Recovery Steam Generators

Abstract

Three of the basic parts of a heat recovery steam generator have been implemented in Omola: the economizer, the superheater and the boiler. A simulator, Omsim, acts on the Omola code to produce equations that can be used for making dynamical simulations. Omsim is, however, far from a final product. This report aims to give some feedback from the use of Omola/Omsim. Difficulties and errors are reported.

Key words
Heat recovery steam generators, economizer, boiler, superheater, omola, simulations

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
25

Recipient's notes

Security classification

Omola Models For Heat Recovery Steam Generators

1. Introduction

This report documents some basic models that can be used for building up more complex models for *heat recovery steam generators*, (HRSG). The models are written in *Omola*— an object-oriented language for model representation, see [1]. The simulator, which is operating on the Omola code, is called *Omsim* by the CACE-group and in this report. Omsim typically checks for consistence, sort equations, *massages* equations (makes substitutions) and tries, if possible, to make index reduction, see [3]. Omsim is just under development and is far from a final product. One major problem with Omsim is that the models sometimes have been manipulated in an undesirable way. An example will be given to show where simulations can give absured answers. Another major problem is that there is no initializer to the integration routines. This means that initializations of all variables have to be made manually. That is even for a small example a hopeless situation. Therefore, at present, the package is only useful for model description (Omola), not for making dynamical simulation (unless the models are written in a Simnon-like fashion).

The attempt here is not to model a whole HRSG, but rather to demonstrate how a general HRSG can be built up by combining three of its basic parts: the economizer, the boiler and the superheater. One simple vertical configuration is shown in Figure 1. The water flow is downwards and the gas flow is upwards. First, the feed water is heated in the **economizer** before it is entering the **boiler**, where it is transformed into steam and finally the steam is heated in the **superheater**. Thereafter, the superheated steam is supposed to enter a steam generator, a part that is not treated here. Nor are other parts modeled, such as attenuators, valves, regulators, etc.

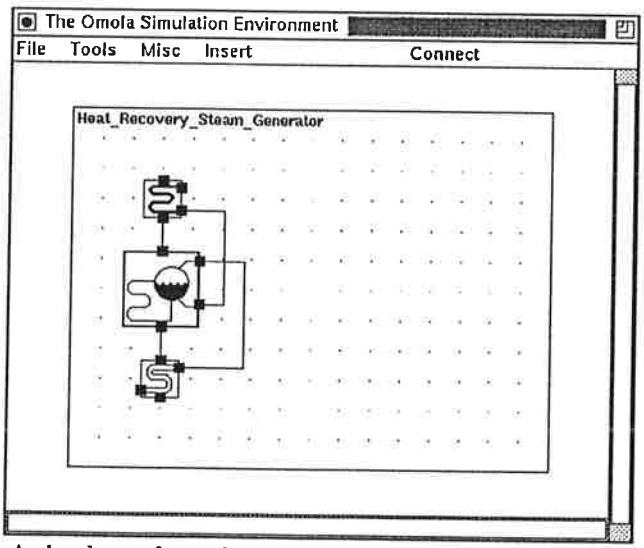


Figure 1. A simple configuration of the basic parts of a HRSG: the economizer, the boiler and the superheater.

2. Terminals

Each block in Figure 1 has four connections: influent gas, effluent gas, influent water or steam and effluent water or steam. Each connection, *terminal*, has three basic quantities: massflow, pressure and enthalpy. Then there are additional information about directions and contents. A terminal is therefore a record-terminal, consisting of subterminals for massflow, pressure and so on. A terminal library is listed below.

```
LIBRARY terminals;

PressureTerminal ISA SimpleTerminal WITH
  quantity := "pressure";
  unit := "Pa";
  direction := 'across;
END;

EnthalpyTerminal ISA SimpleTerminal WITH
  quantity   := "specific.enthalpy";
  unit       := "kJ/kg";
  direction  := 'across;
  default    := 0;
END;

MassFlowInTerminal ISA SimpleTerminal WITH
  quantity   := "mass.flow.rate";
  unit       := "kg/s";
  direction  := 'in;
END;

MassFlowOutTerminal ISA SimpleTerminal WITH
  quantity   := "mass.flow.rate";
  unit       := "kg/s";
  direction  := 'out;
END;

CompositionTerminal ISA SimpleTerminal WITH
  value   TYPE (Gas, Water, Steam);
  default TYPE (Gas, Water, Steam) := 'Gas;
  variability := 'Constant;
END;

InTerminal ISA RecordTerminal WITH
components:
  MassFlow ISA  MassFlowInTerminal;
  Enthalpy ISAN EnthalpyTerminal;
  Pressure ISA  PressureTerminal;
  content  ISA  CompositionTerminal;
END;

OutTerminal ISA RecordTerminal WITH
components:
  MassFlow ISA  MassFlowOutTerminal;
  Enthalpy ISAN EnthalpyTerminal;
  Pressure ISA  PressureTerminal;
  content  ISA  CompositionTerminal;
END;
```

The terminals in e.g. the economizer are used like this:

```
GasIn isa InTerminal with
  content:='Gas;
  Graphic isa Base::Layout with
    x_pos:=200;
    y_pos:=0;
  end;
end;
GasOut isa OutTerminal with
```

```
    content:='Gas;
  Graphic isa Base::Layout with
    x_pos:=200;
    y_pos:=300;
  end;
end;

WaterIn isa InTerminal with
  content:='Water;
  Graphic isa Base::Layout with
    x_pos:=400;
    y_pos:=240;
  end;
end;
WaterOut isa OutTerminal with
  content:='Water;
  Graphic isa Base::Layout with
    x_pos:=400;
    y_pos:=60;
  end;
end;
```

A *Graphic* object has been attached to specify the location of the entries on the block, see Figure 1.

Notations

Enthalpy and pressure are denoted h and p respectively. Massflow will have the notation w in the economizer and the superheater. In the boiler model, however, q is used for massflow. The formulas are then easier to recognize from [6]. Indices are used in short variable names to make formulas more readable. The short variable names are typically consisting of three letters. The rules are that the first letter specifies *w, h or p* whether it is massflow, enthalpy or pressure. Then, the second letter specifies *w, s or g* whether it is water, steam or gas. The third letter indicates *e or l* whether it is entering or leaving. Example: *hwl* means enthalpy-water-leaving and is therefore connected to the terminal WaterOut.Enthalpy.

3. Heat Transfer

The models for heat transfer from gas to water/steam is similar in the economizer model and the superheater model. They both use conductance between tubes. It was therefore convenient to make *functions* calculating conductance. The model for heat transfer in the boiler model is simplified as much as possible to make it easier to simulate. However, when ever an equation solver is included in Omsim, corresponding modifications can be made in the boiler model as well to make it more accurate.

The heat transfer in *economizer.om* is modeled below. Two parameters **Fsg**, a slagging factor, and **Aht**, the heat transfer area of the tubes, are used. The variables are **h**, conductance, and **dTm**, logarithmic mean temperature difference, which are both calculated by use of functions imported from the library file *functions.om*. Notice the somewhat awkward *function-calls*.

```
% Heat transfer ------------
qT = (1-Fsg)*Aht*h*dTm;
% h=Kuc*wge^0.6; In MMS, convection depends on flow rate,
```

```
% but here we calculate Kuc from the geometry, see Tyllered
h ISA GasConductance WITH  % from LIBRARY functions
  Dt:=outer::Dt;
  w:=outer::wge;
END;
% logarithmic mean temperature difference
dTm ISA LogMean WITH % from LIBRARY functions
  x:=Tge-Twl;      % Temp. diff at hot side
  y:=Tgl-Twe;      % Temp. diff at cold side
END;
```

The heat transfer in *superheater.om* is somewhat more complicated since the conductance is depending also on steam properties. The modifications are given below:

```
% Heat transfer ------------
qT = (1-Fsg)*Aht*h*dTmean;

h=1/(1/hgas+1/hsteam); % conductance between gas and steam
hgas ISA GasConductance WITH  % from LIBRARY functions
  Dt:=outer::Dt;
  w:=outer::wge;
END;
hsteam ISA SteamConductance WITH  % from LIBRARY functions
  Dt:=outer::Dt;
  w:=outer::wse;
  T:=Tsl;
  p:=psl;
END;

% logarithmic mean temperature difference
dTmean ISA LogMean WITH % from LIBRARY functions
  x:=Tge-Tsl;      % Temp. diff at hot side
  y:=Tgl-Tse;      % Temp. diff at cold side
END;
```

It would be desirable to use a similar heat transfer in the boiler model. However, to make it easier to simulate in the present implementation of Omsim (lacking the equation solver), *boilerD.om* uses the more simple expression below.

```
% Heat transfer from gas to riser steam
Pow=Cpow*(Tgl-Ts);
```

The functions used to calculate conductances and logarithmic mean difference are in the library file *functions.om* listed below.

```
library functions;

% Auxiliary function defining the logarithmic mean function
Logmean ISA Class WITH
   x,y,value TYPE Real;
   value = if abs(x-y) > 0.05*max(x,y)
           then (x-y)/ln(x/y)
           else (x+y)/2*(1-sqr(x-y)/(12*x*y)*(1-sqr(x-y)/(2*x*y)));
END;

TubeConductance ISA Model WITH
 parameters:
  Dt ISA Parameter; % Diameter of tubes
  Ag ISA Parameter with value:=3.14*Dt*Dt/4; end; % tube cross section area
  k  ISA Parameter; % relation constant
  n1 ISA Parameter; % relation exponent
  n2 ISA Parameter; % relation exponent

 variables:
```

```
  value TYPE Real;
  w ISA Variable; % mass flow

  % Quantities specific for the media
  lambda ISA Variable; % thermal conductivity [W/m/K]
  eta    ISA Variable; % dynamic viscosity  [kg/s/m]
  Cp     ISA Variable; % spec. heat capacity [J/kg/K]
  Re     ISA Variable; % Reynolds number
  Pr     ISA Variable; % Prandtl number
  Nu     ISA Variable; % Nussel number

 equations:
  Re=w*Dt/Ag/eta;
  Pr=Cp*eta/lambda;
  Nu=k*Re^n1*Pr^n2;
  value=Nu*lambda/Dt;
END; % TubeConductance

GasConductance ISA TubeConductance WITH
 k:=0.33;
 n1:=0.6;
 n2:=0.33;
 lambda:=0.045; % W/m/K, thermal conductivity of air
 eta:=2.86E-5;  % kg/s/m, dynamical viscosity of air
 Cp:=1040.0;    % J/kg/K, Spec. heat capacity of air
END; % GasConductance

SteamConductance ISA TubeConductance WITH
 variables:
 T ISA Variable;
 p ISA Variable;

 k:=0.023;
 n1:=0.8;
 n2:=0.4;
 lambda=5.7E-2*(1.1-exp(-T/1E2-p/1E5)); % Approximate or use steam tables
 eta=2.3E-5*(1.1-exp(-T/1E2-p/1E5));
 Cp=3000*(1.1-exp(-T/1E2-p/1E5));
END; % SteamConductance
```

Notice that approximate expressions have been used here for thermal conductivity, **lambda**, dynamical viscosity, **eta** and specific heat capacity, **Cp**. A future extension could be to use real steam tables functions here.

4. Water/Steam- (and gas-) Tables

Some water/steam tables are installed. These are listed below.

THP(h,p)	temperature as a function of enthalpy and pressure
VHP(h,p)	spec. volume as a function of enthalpy and pressure
HTP(T,p)	enthalpy as a function of temp. and pressure
HTP(T,p,gasmix)	enthalpy as a function of temp., pressure and gas mixture
RTP(T,p,gasmix)	gas constant/mole weight as a function of T,p and gas mixture
H1P(p)	enthalpy as a function of pressure for saturated water
V1P(p)	spec. volume as a function of pressure for saturated water
TP(p)	temperature as a function of pressure
H2P(p)	enthalpy as a function of pressure for saturated steam
V2P(p)	spec. volume as a function of pressure for saturated steam

When a third argument *gasmix* is present the calculation is made for a specific gas mixture rather than for steam. The vector *gasmix* is defined as below:

```
gasmix Type Row [7] := [0.6911, 0.078, 0.1697, 0., 0., 0., 0.0341];
                       % [   N2 ,  CO2 , O2    , Ar,SO2,NO ,  H2O  ]
```

The tables can be introduced already in the variable declaration. Below, en example is given, which is the variable declaration for the economizer model. The tables are introduced for gas enthalpy, gas constant, water temperature and water density (inverse volume).

```
variables:
  wwe     ISA Variable; % Mass flow of water entering
  wwl     ISA Variable; % Mass flow of water leaving
  wge     ISA Variable; % Mass flow of gas entering
  wgl     ISA Variable; % Mass flow of gas leaving
   % Specific enthalpy of ...
  hwe     ISA Variable; % water entering
  hwl     ISA Variable; % water leaving
  hge     ISA Variable WITH value=HTP(Tge,pge,gasmix)+NIL;END; % gas entering
  hgl     ISA Variable WITH value=HTP(Tgl,pgl,gasmix);END; % gas leaving
   % Temperature of ...
  Twe     ISA Variable WITH value:=THP(hwe,pwe);END; % water entering
  Twl     ISA Variable WITH value:=THP(hwl,pwl);END; % water leaving
  Tge     ISA Variable;
  Tgl     ISA Variable;
   % Desity of ...
  rhowe   ISA Variable WITH value=1/VHP(hwe,pwe);END; % water entering
  rhowl   ISA Variable WITH value:=1/VHP(hwl,pwl);END; % water leaving
  rhogl   ISA Variable; % gas leaving
   % Pressure of ...
  pwe     ISA Variable; % water entering
  pwl     ISA Variable; % water leaving
  pge     ISA Variable; % gas entering
  pgl     ISA Variable; % gas leaving

  qT      ISA Variable; % Heat transfer rate from gas to water
  gasmix Type Row [7] := [0.6911, 0.078, 0.1697, 0., 0., 0., 0.0341];
                         % [   N2 ,  CO2 , O2    , Ar,SO2,NO ,  H2O  ]
  R ISA Variable WITH  % J/kgK, gas const/mole weight
    initial:=290;
    value=RTP(Tgl,pgl,gasmix);
  END;
  NIL ISA Model WITH  % Trick to fool DASSL believe an equation is dynamical
   zero ISA Parameter WITH default:=0;END;
   value Type Real;
   value :=zero*Base::Time;
  END;
```

The use of NIL above is a trick to make the equation be sorted under dynamical equations. Then the integration routine DASSL can solve it (provided that all variables are initiated properly). Hopefully, when ever an equation solver is included, this trick will not be needed anymore. The use of = sometimes and := occasionally may look strange. And it is! A rule is to use := when ever it is possible. Then the causality is clear and fewer variables need to be initialized. On the other hand, if Omsim complains about an equation system containing assignment one has to change to = and initialize some variables.

5. The Economizer Model

Most of the economizer model is adopted from [2] and the same variable naming rules have been kept here. The following assumptions are made: Metal and water has equal temperature. Flue gas is modeled as an ideal gas (air) with atmospheric pressure. Radiant heat transfer is neglected. Convective heat transfer depends on gas flow rate.

As in [2], water temperature and density are calculated by use of water/steam table functions. In [2] gas temperature depends on enthalpy and

moisture content via a fitted expression. Here, instead, we let the relations between enthalpy, pressure and temperature depend on the total gas mixture, not only the moisture content. The difference from [2] is also that parameter K_{uc} in the conductance relation $h = K_{uc} w_{ge}^{0.6}$ is calculated from the geometry, see [5]. Also, R (gas constant/mole weight) is calculated as a function of temperature, pressure and gas mixture. A listing of the equations is given below:

```
equations:
  % Mass balance --------
  wwe=wwl;   % water
  wge=wgl;   % flue gas

  % Energy balance for water and metal (0.11=cp metal/cp water) ----
  (rhowl*Vw+0.11*Mm)*dot(hwl)=wwe*hwe-wwl*hwl+qT;

  % Energy balance for gas --------
  Vg*rhogl*dot(hgl)=k*(wge*hge-wgl*hgl-qT); % k=cp/cv for air

  % Heat transfer ------------
  qT = (1-Fsg)*Aht*h*dTm;
  % h=Kuc*wge^0.6; In MMS, convection depends on flow rate,
  % but here we calculate Kuc from the geometry, see Tyllered
  h ISA GasConductance WITH  % from LIBRARY functions
    Dt:=outer::Dt;
    w:=outer::wge;
  END;
  % logarithmic mean temperature difference
  dTm ISA LogMean WITH % from LIBRARY functions
    x:=Tge-Twl;      % Temp. diff at hot side
    y:=Tgl-Twe;      % Temp. diff at cold side
  END;

  % Fluid mechanics --------------
  rhowe*(pwe-pwl)=abs(wwe)*wwe/sqr(Cf)+NIL; % pressure drop for water

  pge=pgl;                                  % no pressure drop for gas

  % Fluid properties ----------------
  rhogl=pgl/R/Tgl;            % flue gas density (=rhoge) (ideal gas)
```

Comments of the model

The dependents on **dot(rhowl)** and **dot(rhogl)** have been neglected in the energy balance equations. The following approximation for the rate of change of the energy stored in the metal is made:

$$\frac{d}{dt}(M_m c_{pm} T_m) = \frac{d}{dt}(M_m c_{pm} T_{wl}) = \frac{d}{dt}(M_m (c_{pm}/c_w) h_{wl})$$

where $c_{pm}/c_w = 0.11$. The expression **abs(wwe)*wwe** in the pressure drop equation, is used rather than what is often seen **sqr(wwe)**. The equation will then work even though the mass flow change direction. Besides, it is much better for the numerics. The NIL trick, once again, is temporary and will hopefully be removed when an equation solver is included.

6. The Superheater Model

The superheater model is taken partly form [2] and partly from [4]. The difference form the economizer model is that there is a temperature difference

between metal and steam, see [4]. Also, the conductance become more complicated as has been shown earlier. The mass flow of steam has been modeled as constant according to [4] and opposed to [2]. Otherwise, the superheater model looks similar the the economizer model. It is listed below:

```
equations:
  % Mass balance --------
  wse=wsl;   % steam (acc. to Lindahl, MMS uses drho/dt # 0)
  wge=wgl;   % flue gas

  % Energy balance for steam and metal (Lindahl)
  rhosl*Vs*dot(hsl)+Cpm*Mm*dot(Tm)=wse*hse-wsl*hsl+qT;
  Tm = Tsl -(wse*hse-wsl*hsl)*cTslm; % metal temperature (Lindahl)

  % Energy balance for gas --------
  Vg*rhogl*dot(hgl)=k*(wge*hge-wgl*hgl-qT);

  % Heat transfer ------------
  qT = (1-Fsg)*Aht*h*dTmean;

  h=1/(1/hgas+1/hsteam); % conductance between gas and steam
  hgas ISA GasConductance WITH  % from LIBRARY functions
    Dt:=outer::Dt;
    w:=outer::wge;
  END;
  hsteam ISA SteamConductance WITH  % from LIBRARY functions
    Dt:=outer::Dt;
    w:=outer::wse;
    T:=Tsl;
    p:=psl;
  END;

  % logarithmic mean temperature difference
  dTmean ISA LogMean WITH % from LIBRARY functions
    x:=Tge-Tsl;      % Temp. diff at hot side
    y:=Tgl-Tse;      % Temp. diff at cold side
  END;

  % Fluid mechanics --------------
  rhose*(pse-psl)=abs(wse)*wse/sqr(Cf)+NIL; % pressure drop for water

  pge=pgl;                                  % no pressure drop for gas

  % Fluid properties ----------------
  rhogl=pgl/R/Tgl;           % flue gas density (=rhoge) (ideal gas)
```

Comments of the model

See comments of the economizer model and the heat transfer section.

7. The Boiler Model

The boiler model is taken from [6] with water/steam table functions instead of polynomial approximations. The gas part has been made as simple as possible, but can probably be changed to something similar to that in the superheater model once the equation solver is included. There have been a lot of difficulties to make this model work. The model consists of just 4 balance equations and some other relations between variables. It is shown in [6] how these equations can be written as a third order system with the states beeing drum pressure, p, drum water volume, Vw, and steam quality at riser outlet, xr. The equations can be manipulated by hand to produce the derivative of these states on the left hand side, as in Simnon. This has been done in *boilerD.om*, and gives

a model that works well. At least, it can reproduce the same step responses that are made in [6]. However, if the equations are written down in an Omola spirit, like balance equations, strange things will happen. Such a model is made in *boilerERROR1.om*, but without the gas part. The equations from this file is listed below:

```
    % From Steam Tables
    Hs       ISA Variable WITH value=H2P(p);END;     % Steam enthalphy
    rs       ISA Variable WITH value=1/V2P(p);END;   % Steam density
    Hw       ISA Variable WITH value=H1P(p);END;     % Water enthalphy
    rw       ISA Variable WITH value=1/V1P(p);END;   % Water density
    Ts       ISA Variable WITH value=TP(p);END;      % Steam temperature
    Hfw      ISA Variable WITH value=HTP(Tfw,p);END; % Feedwater enthalphy

realizations:
    equations:
    % Drum energy balance
    dot(rs*Vst*Hs+rw*Vwt*Hw+m*cp*Tm)=Pow+qfw*Hfw-qs*Hs;
      Vst = Vdrum - Vw + am*Vr;              % Total steam volume
      Vwt = Vw + Vdc + (1-am)*Vr;            % Total water volume
      Tm  = Ts;       % metal temp=sat. steam temp corresponing to p

    % Drum mass balance
    dot(rw*Vst+rw*Vwt)=qfw-qs;

    % Riser energy balance
    dot(rs*am*Vr*Hs+rw*(1-am)*Vr*Hw)=Pow+qdc*Hw-xr*qr*Hs-(1-xr)*qr*Hw;

    % Riser mass balance
    dot(am*Vr*rs+(1-am)*Vr*rw)=qdc-qr;

    % Momentum balance for Downcomer
    am*Vr*(rw-rs)=0.5*k*abs(qdc)*qdc;

    % Average steam-water volume ratio in the risers (am)
    am=rw/(rw-rs)*(1-rs/(rw-rs)/xr*ln(1+(rw/rs-1)*xr));

    dl=(Vw+am*Vr)/Adrum; % Drum water level
```

This model is accepted by Omsim/Omola. However, the most disturbing fact is that a simulation gives the **WRONG** answer. For example $V_w = -40$, a negative water volume! In the present implementation of Omsim the unix command *setenv OMSIM_GAUSSMAX 10* must be made before starting Omsim. Otherwise, Omsim will complain about index reduction of the water/steam table functions above. If the water/steam table functions are substituted into the equations, i.e. the variables rs, rw, Hs, Hw, Ts and Hfw are eliminated, then the problem with index reduction disappear. This is tried in another Omola implementation in the file, *boilerERROR2.om*. However, the same simulation error will occur. Therefore, it seems that the error is caused by unfortunate manipulations of the equations rather than by the index reduction procedure.

Another Omola code for the same model is given in the file *boiler.om*. Here, the differentiation has been made by hand. A lot of auxiliary variables need then to be introduced. The Omola code is still quite nice and readable. And fortunately, a simulation gives the correct answer. However, after including the gas part, it was hard to find the steady state which is necessary in order to start DASSL. Thus, we have to wait for the equation solver before this model can be used. The realization part of *boiler.om* is listed below:

```
%   dot(rs*Hs*Vst+rw*Hw*Vwt+m*cp*Tm)=Pow+qfw*Hfw-qs*Hs;
    DrumEnergyBalance ISA SetOFDAE WITH
     dEdrum,dEs,dEw,dEm ISA Variable;
     %-------------------------------------------------------------
     dEdrum=Pow+qfw*Hfw-qs*Hs;              % Drum energy balance
```

```
    %-----------------------------------------------------------
    dEdrum=dEs+dEw+dEm;
    dEs= drs*Hs*Vst+rs*dHs*Vst+rs*Hs*dVst; % steam part
    dEw= drw*Hw*Vwt+rw*dHw*Vwt+rw*Hw*dVwt; % water part
    dEm= m*cp*dTs;                         % metal part
    Vst = Vdrum - Vw + am*Vr;              % Total steam volume
    Vwt = Vw + Vdc + (1-am)*Vr;            % Total water volume
    Tm  = Ts;       % metal temp=sat. steam temp corresponing to p
  END; % DrumEnergyBalance

%   dot(rs*Vst+rw*Vwt)=qfw-qs; % global mass balance
    DrumMassBalance ISA SetOfDAE WITH
    dMdrum,dMs,dMw ISA Variable;
    %-----------------------------------------------------------
    dMdrum=qfw-qs;                         % Rate of mass in drum
    %-----------------------------------------------------------
    dMdrum=dMs+dMw;
    dMs=drs*Vst+rs*dVst;                   % steam part
    dMw=drw*Vwt+rw*dVwt;                   % water part
  END; % DrumMassBalance

%   dot(rs*Hs*am*Vr+rw*Hw*(1-am)*Vr)=Pow+qdc*Hw-xr*qr*Hs-(1-xr)*qr*Hw;
    RiserEnergyBalance ISA SetOfDAE WITH
    dEriser,dErs,dErw ISA Variable;
    %-----------------------------------------------------------
    dEriser=Pow+qdc*Hw-xr*qr*Hs-(1-xr)*qr*Hw;  % Riser energy balance
    %-----------------------------------------------------------
    dEriser=dErs+dErw;
    dErs=drs*Hs*am*Vr+rs*dHs*am*Vr+rs*Hs*dam*Vr;          % steam part
    dErw=drw*Hw*(1-am)*Vr+rw*dHw*(1-am)*Vr+rw*Hw*(-dam)*Vr; % water part
  END; % RiserEnergyBalance

%   dot(rs*am*Vr+rw*(1-am)*Vr)=qdc-qr;
    RiserMassBalance ISA SetOfDAE WITH
    dMriser,dMrs,dMrw ISA Variable;
    %-----------------------------------------------------------
    dMriser=qdc-qr;                        % Riser mass balance
    %-----------------------------------------------------------
    dMriser=dMrs+dMrw;
    dMrs=drs*am*Vr+rs*dam*Vr;
    dMrw=drw*(1-am)*Vr+rw*(-dam)*Vr;
  END; % RiserMassBalance

  % Momentum balance for Downcomer
    am*Vr*(rw-rs)=0.5*k*abs(qdc)*qdc;

  % Average steam-water volume ratio in the risers (am)
  am=rw/(rw-rs)*(1-rs/(rw-rs)/xr*ln(1+(rw/rs-1)*xr));
  damdxr=-am/xr+rw/(rs+(rw-rs)*xr);
  dam=damdxr*dot(xr);

  dl=(Vw+am*Vr)/Adrum; % Drum water level

  %  --------Gas side----------
  wgl=wge;   % mass balance
  pgl=pge;   % no pressure drop for gas

  % Energy balance for gas --------
  Vg*rhogl*dot(hgl)=1.4*(wge*hge-wgl*hgl-Pow);  % 1.4=cp/cv

  % Heat transfer from gas to riser steam
  Pow=Cpow*(Tgl-Ts);

  % Fluid properties ----------------
  rhogl=pgl/R/Tgl;           % flue gas density (=rhoge) (ideal gas)

  % Partial derivatives
  dHs=dHsdp*dot(p);
  drs=drsdp*dot(p);
  dHw=dHwdp*dot(p);
  drw=drwdp*dot(p);
```

```
dTs=dTsdp*dot(p);
dVst=-dot(Vw)+Vr*damdxr*dot(xr);
dVwt=-dVst;
```

8. How and where to start

/Run

One of the cosmetic capabilities is the possibility to make *icons*. Rumors say that the icons must be in the same directory in which Omsim is started up. Therefore, the following file configuration has been chosen. In the **Run**-directory there are files like *RunEconomizer*, *RunBoilerD*, *RunBoiler*, *RunSuperheater* and *RunHRSG*. Also, there are the bitmap-files of the icons. Thus, it is here Omsim should be started up. This is done by simply making the unix command *source* for one of the above Run-files, e.g. *source RunEconomizer* starts Omsim and load the appropriate files for running an economizer. To start the whole thing like in Figure 1, *RunHRSG* is used. Use *RunBoilerD* when simulating the boiler. This uses a model that has been manipulated by hand such that the state derivatives are to the left hand side, just like Simnon. *RunBoiler* uses a more Omola-like model. However, a lot of auxiliary variables have been introduced in order to get it to work. Consequently, a lot more variables also need to be initialized here to simulate this model. Since the present implementation of Omsim has no equation solver that initialize the states before the integration routine starts this model cannot be simulated right now (however, it can be instantiated).

/Lib

All Omola models loaded from the **Run**-directory are in the **Lib**-directory. Here are following library files: *terminals.om*, *economizer.om*, *boilerD.om*, *boiler.om* and *superheater.om*. There are also *Test*-files for running the different models. For example *TestEconomizer* is loaded by *RunEconomizer* and defines the appropriate terminals and gives initial values in order to start a simulation.

There are no user defined functions in Omola, but *function-like* objects are defined in a function library file: *function.om*. These *object-functions* can by used like functions, even if it is a bit awkward.

9. Conclusions

Three of the basic parts of a heat recovery steam generator has been implemented in Omola: the economizer, the superheater and the boiler. A final evaluation of Omola/Omsim is not made. The purpose of this work was to find the weak points of Omola/Omsim such that these can be eliminated in the developing of the programs. Let us focus on two major difficulties: starting of the integration routine and manipulations of the equations.

The integration routine DASSL have to be given a starting point. Without an equation solver this is a hopeless situation for the user. The inclusion of an

equation solver has therefore high priority. However, it seems that it is more than the initialization that is causing the start problem. A simulation of each of the models at a time works fine. But when connecting the models, as in Figure 1, the simulator is almost stuck. It will take several minutes just to take one single step. It doesn't seem to matter if all variables are initialized. There is obviously a need to look deeper into what DASSL is doing in the starting procedure.

The manipulations of the equations are another big problem. The user is not allowed to take part of how the manipulations are made. A horrible example has been given showing that manipulations of equations can destroy numeric properties such that simulations give wrong answers. Because of this, a nice boiler model, written in an Omola spirit, couldn't be used. In stead, the only boiler model that works well right now is a model written in a Simnon-like spirit. There is obviously a lot to be done to make Omola/Omsim work.

10. References

[1] Andersson, M. (1990): "Omola—An Object-Oriented Language for Model Representation," Lic. Thesis, TFRT-3208, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

[2] Babcock & Wilcox, "MMS, Part 1a, Fossile and Nuclear Components," Section 3.2.

[3] Mattsson, S. E. & G. Söderlind (1991): "Index Reduction in Differential-Algebraic Equations Using Dummy Derivatives," Lic. Thesis, TFRT-7477, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

[4] Lindahl, S. (1976): "A Non-Linear Drum Boiler-Turbine Model," Lic. Thesis, TFRT-3132, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

[5] Tyllered, G. O. (1972): "Termodynamik," Compendium from the department of Thermodynamics, Lund Institute of Technology, Sweden.

[6] Åström, K. J. & R. Bell (1988): "Simple Drum-Boiler Models," TFRT-7402, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

```
LIBRARY terminals;                    % Some terminal definitions

PressureTerminal ISA SimpleTerminal WITH
  quantity := "pressure";
  unit := "Pa";
  direction := 'across;
END;

EnthalpyTerminal ISA SimpleTerminal WITH
  quantity   := "specific.enthalpy";
  unit       := "kJ/kg";
  direction  := 'across;
  default    := 0;
END;

MassFlowInTerminal ISA SimpleTerminal WITH
  quantity   := "mass.flow.rate";
  unit       := "kg/s";
  direction  := 'in;
END;

MassFlowOutTerminal ISA SimpleTerminal WITH
  quantity   := "mass.flow.rate";
  unit       := "kg/s";
  direction  := 'out;
END;

CompositionTerminal ISA SimpleTerminal WITH
  value   TYPE (Gas, Water, Steam);
  default TYPE (Gas, Water, Steam) := 'Gas;
  variability := 'Constant;
END;

InTerminal ISA RecordTerminal WITH
components:
  MassFlow ISA  MassFlowInTerminal;
  Enthalpy ISAN EnthalpyTerminal;
  Pressure ISA  PressureTerminal;
  content  ISA  CompositionTerminal;
END;

OutTerminal ISA RecordTerminal WITH
components:
  MassFlow ISA  MassFlowOutTerminal;
  Enthalpy ISAN EnthalpyTerminal;
  Pressure ISA  PressureTerminal;
  content  ISA  CompositionTerminal;
END;
```

```
library functions;

% Auxiliary function defining the logaritmic mean function
Logmean ISA Class WITH
   x,y,value TYPE Real;
   value = if abs(x-y) > 0.05*max(x,y)
           then (x-y)/ln(x/y)
           else (x+y)/2*(1-sqr(x-y)/(12*x*y)*(1-sqr(x-y)/(2*x*y)));
END;

% Flue gas temperature as a function of Mf & hgl, see MMS
FunctionOfMoistureAndEnthalpy ISA Class WITH
   value,TF,Moisturefraction,enthalpy  TYPE Real;
   hMMSunits ISA Variable WITH initial:=80;END; % Start value to DASSL
   zero ISA Parameter WITH default:=0;END; % Fix to make DASSL believe
   % that this is a dynamical equation !!!
   hMMSunits=enthalpy/2326.0; % Btu/lbm
   TF = ((( 3.7646e-7 - (1.7871e-7)*Moisturefraction)*hMMSunits
            +(-1.0344e-3 + (1.6434e-4)*Moisturefraction))*hMMSunits
          +(4.16-3*Moisturefraction))*hMMSunits+80+zero*Base::Time;
   value = 5.0/9.0*(TF-32)+273.15; % Kelvin
END;

TubeConductance ISA Model WITH
 parameters:
  Dt ISA Parameter; % Diameter of tubes
  Ag ISA Parameter with value:=3.14*Dt*Dt/4; end; % tube cross section area
  k  ISA Parameter; % relation constant
  n1 ISA Parameter; % relation exponent
  n2 ISA Parameter; % relation exponent

 variables:
  value TYPE Real;
  w ISA Variable; % mass flow

  % Quantities specific for the media
  lambda ISA Variable; % thermal conductivity [W/m/K]
  eta    ISA Variable; % dynamic viscosity  [kg/s/m]
  Cp     ISA Variable; % spec. heat capacity [J/kg/K]
  Re     ISA Variable; % Reynolds number
  Pr     ISA Variable; % Prandtl number
  Nu     ISA Variable; % Nussel number

 equations:
  Re=w*Dt/Ag/eta;
  Pr=Cp*eta/lambda;
  Nu=k*Re^n1*Pr^n2;
  value=Nu*lambda/Dt;
END; % TubeConductance

GasConductance ISA TubeConductance WITH
 k:=0.33;
 n1:=0.6;
 n2:=0.33;
 lambda:=0.045; % W/m/K, thermal conductivity of air
 eta:=2.86E-5;  % kg/s/m, dynamical viscosity of air
 Cp:=1040.0;    % J/kg/K, spec. heat capacity of air
END; % GasConductance

SteamConductance ISA TubeConductance WITH
 variables:
 T ISA Variable;
 p ISA Variable;

 k:=0.023;
 n1:=0.8;
 n2:=0.4;
 lambda=5.7E-2*(1.1-exp(-T/1E2-p/1E5)); % Approximate or use steam tables
 eta=2.3E-5*(1.1-exp(-T/1E2-p/1E5));
 Cp=3000*(1.1-exp(-T/1E2-p/1E5));
END; % SteamConductance
```

```
LIBRARY economizer;  USES terminals, functions;

EconomizerModel ISA Model WITH
  % Economizer model adopted from MMS (Babcock and Wilcox),
  % Part 1a, Fossile and Nuclear Components, ECON/EC, Section 3.2.
  %
  % Subcooled water is heated before entering the drum-system.
  %
  % Metal and water has equal temperature. Flue gas is modelled as an
  % ideal gas (air) with atmospheric pressure. Radiant heat transfer
  % neglected. Convective heat transfer depends on gas flow rate.
  % Steam table functions for water temperature and density.
  % Gas temperature depends on enthalpy and moisture content.
  %
  % Ulf Holmberg, Dept. of Aut. Control,
  % Lund Inst. of Techn., Lund Sweden

  terminals:       % Gas and water in and outlet
  GasIn isa InTerminal with
    content:='Gas;
    Graphic isa Base::Layout with
      x_pos:=200;
      y_pos:=0;
    end;
  end;
  GasOut isa OutTerminal with
     content:='Gas;
    Graphic isa Base::Layout with
      x_pos:=200;
      y_pos:=300;
    end;
  end;

  WaterIn isa InTerminal with
    content:='Water;
    Graphic isa Base::Layout with
      x_pos:=400;
      y_pos:=240;
    end;
  end;
  WaterOut isa OutTerminal with
    content:='Water;
    Graphic isa Base::Layout with
      x_pos:=400;
      y_pos:=60;
    end;
  end;
icon:
 Graphic isa Layout with bitmap TYPE String:="econ";  end;
  parameters:
    Vw   ISA Parameter with default := 0.02; end; % Volume of water
    Vg   ISA Parameter with default := 1; end; % Volume of flue gas
    Mm   ISA Parameter with default := 20; end; % Metal mass
    Aht  ISA Parameter with default := 1.6; end; % Heat transfer area of tubes
    Dt   ISA Parameter with default := 0.05; end; % Diameter of tubes
    Cf   ISA Parameter with default := 0.03; end;
     % Valve constant for pressure drop.
     % Should possibly be expression with length and diameter of tubes.

  % fixed parameters
    k    ISA Parameter WITH value := 1.4; END; % cp/cv for water
    Fsg  ISA Parameter WITH value := 0.1; END; % slagging factor, [0,0.3] (MMS)

  variables:
    wwe    ISA Variable; % Mass flow of water entering
    wwl    ISA Variable; % Mass flow of water leaving
    wge    ISA Variable; % Mass flow of gas entering
    wgl    ISA Variable; % Mass flow of gas leaving
     % Specific enthalpy of ...
    hwe    ISA Variable; % water entering
    hwl    ISA Variable; % water leaving
    hge    ISA Variable WITH value=HTP(Tge,pge,gasmix)+NIL;END; % gas entering
    hgl    ISA Variable WITH value=HTP(Tgl,pgl,gasmix);END; % gas leaving
     % Temperature of ...
    Twe    ISA Variable WITH value:=THP(hwe,pwe);END; % water entering
    Twl    ISA Variable WITH value:=THP(hwl,pwl);END; % water leaving
    Tge    ISA Variable;
    Tgl    ISA Variable;
     % Desity of ...
    rhowe  ISA Variable WITH value=1/VHP(hwe,pwe);END; % water entering
    rhowl  ISA Variable WITH value:=1/VHP(hwl,pwl);END; % water leaving
    rhogl  ISA Variable; % gas leaving
     % Pressure of ...
    pwe    ISA Variable; % water entering
    pwl    ISA Variable; % water leaving
    pge    ISA Variable; % gas entering
    pgl    ISA Variable; % gas leaving

    qT     ISA Variable; % Heat transfer rate from gas to water
    gasmix Type Row [7] := [0.6911, 0.078, 0.1697, 0., 0., 0., 0.0341];
                        %  [   N2 ,  CO2 , O2    , Ar,SO2,NO ,  H20  ]
    R ISA Variable WITH  % J/kgK, gas const/mole weight
      initial:=290;
      value=RTP(Tgl,pgl,gasmix);
    END;
    NIL ISA Model WITH  % Trick to fool DASSL believe an equation is dynamical
     zero ISA Parameter WITH default:=0;END;
     value Type Real;
     value :=zero*Base::Time;
    END;

    equations:
      % Mass balance --------
      wwe=wwl;   % water
      wge=wgl;   % flue gas

      % Energy balance for water and metal (0.11=cp metal/cp water) ----
      (rhowl*Vw+0.11*Mm)*dot(hwl)=wwe*hwe-wwl*hwl+qT;

      % Energy balance for gas --------
      Vg*rhogl*dot(hgl)=k*(wge*hge-wgl*hgl-qT);

      % Heat transfer ------------
      qT = (1-Fsg)*Aht*h*dTm;
      % h=Kuc*wge^0.6; In MMS, convection depends on flow rate,
      % but here we calculate Kuc from the geometry, see Tyllered
      h ISA GasConductance WITH  % from LIBRARY functions
        Dt:=outer::Dt;
        w:=outer::wge;
      END;
      % logarithmic mean temperature difference
      dTm ISA LogMean WITH % from LIBRARY functions
        x:=Tge-Twl;      % Temp. diff at hot side
        y:=Tgl-Twe;      % Temp. diff at cold side
      END;

      % Fluid mechanics --------------
      rhowe*(pwe-pwl)=abs(wwe)*wwe/sqr(Cf)+NIL; % pressure drop for water

      pge=pgl;                                  % no pressure drop for gas
```

```
      % Fluid properties ----------------
      rhogl=pgl/R/Tgl;           % flue gas density (=rhoge) (ideal gas)

  connections:
    WaterIn.Pressure   = pwe;
    WaterIn.MassFlow   = wwe;
    WaterIn.Enthalpy   = hwe;

    WaterOut.Pressure  = pwl;
    WaterOut.MassFlow  = wwl;
    WaterOut.Enthalpy  = hwl;

    GasIn.Pressure     = pge;
    GasIn.MassFlow     = wge;
    GasIn.Enthalpy     = hge;

    GasOut.Pressure    = pgl;
    GasOut.MassFlow    = wgl;
    GasOut.Enthalpy    = hgl;

END; % EconomizerModel
```

```
LIBRARY superheater;  USES terminals, functions;

SuperheaterModel ISA Model WITH
  % Superheater model adopted from MMS (Babcock and Wilcox),
  % Part 1a, Fossile and Nuclear Components, SPRHTR/HS, Section 3.2,
  % and from Lindahl, TFRI-3132, pp. 72--76.
  %
  % Saturated steam/water enters the superheater and is heated to
  % superheated state.
  %
  % Metal and steam has a small temperature difference determined by the
  % heat transfer. Flue gas is modelled as an
  % ideal gas (air) with atmospheric pressure. Radiant heat transfer
  % neglected. Convective heat transfer depends on gas flow rate.
  % Steam table functions for steam temperature and density.
  %
  % Ulf Holmberg, Dept. of Aut. Control,
  % Lund Inst. of Techn., Lund Sweden

  terminals:       % Steam and gas in and outlet
  GasIn isa InTerminal with
    content:='Gas;
    Graphic isa Base::Layout with
      x_pos:=200;
      y_pos:=0;
    end;
  end;
  GasOut isa OutTerminal with
     content:='Gas;
    Graphic isa Base::Layout with
      x_pos:=200;
      y_pos:=300;
    end;
  end;

  SteamIn isa InTerminal with
    content:='Steam;
    Graphic isa Base::Layout with
      x_pos:=400;
      y_pos:=240;
    end;
  end;
  SteamOut isa OutTerminal with
    content:='Steam;
    Graphic isa Base::Layout with
      x_pos:=0;
      y_pos:=60;
    end;
  end;
icon:
 Graphic isa Layout with bitmap TYPE String:="super";  end;
  parameters:
    Vs   ISA Parameter WITH default := 0.02; END; % Volume of steam
    Vg   ISA Parameter WITH default := 1;    END; % Volume of flue gas
    Mm   ISA Parameter WITH default := 20;   END; % Metal mass
    Aht  ISA Parameter WITH default := 1.6;  END; % Heat transfer area of tubes
    Dt   ISA Parameter WITH default := 0.05; END; % Diameter of tubes
    Cf   ISA Parameter WITH default := 0.03; END;
     % Valve constant for pressure drop.
     % Should possibly be expression with length and diameter of tubes.

  % fixed parameters
    k     ISA Parameter WITH value := 1.4; END; % cp/cv for water
    Fsg   ISA Parameter WITH value := 0.1; END; % slagging factor,[0,0.3](MMS)
    Cpm   ISA Parameter WITH value := 460.0; END; % J/kg/K, heat cap. for steel

    cTslm ISA Parameter WITH value := 4E-6; END; % heat transfer coeff.

  variables:
     % Mass flow of ...
    wse    ISA Variable; % steam entering
    wsl    ISA Variable; % steam leaving
    wge    ISA Variable; % gas entering
    wgl    ISA Variable; % gas leaving
     % Specific enthalpy of ...
    hse    ISA Variable; % steam entering
    hsl    ISA Variable; % steam leaving
    hge    ISA Variable WITH value=HTP(Tge,pge,gasmix)+NIL;END; % gas entering
    hgl    ISA Variable WITH value=HTP(Tgl,pgl,gasmix);END; % gas leaving
     % Temperature of ...
    Tse    ISA Variable WITH value=THP(hse,pse);END; % steam entering
    Tsl    ISA Variable WITH value=THP(hsl,psl);END; % steam leaving
    Tge    ISA Variable; % gas entering
    Tgl    ISA Variable; % gas leaving
    Tm     ISA Variable; % metal
     % Density of ...
    rhose  ISA Variable WITH value=1/VHP(hse,pse);END; % steam entering
    rhosl  ISA Variable WITH value=1/VHP(hsl,psl);END; % steam leaving
    rhogl  ISA Variable; % gas leaving
     % Pressure of ...
    pse    ISA Variable; % steam entering
    psl    ISA Variable; % steam leaving
    pge    ISA Variable; % gas entering
    pgl    ISA Variable; % gas leaving

    qT     ISA Variable; % Heat transfer rate from gas to steam
    h      ISA Variable; % conductance between gas and steam
    gasmix Type Row [7] := [0.6911, 0.078, 0.1697, 0., 0., 0., 0.0341];
                           % [   N2 ,  CO2 , O2    , Ar,SO2,NO ,  H20  ]
    R ISA Variable WITH  % J/kgK, gas const/mole weight
      initial:=290;
      value=RTP(Tgl,pgl,gasmix);
  END;
  NIL ISA Model WITH  % Trick to fool DASSL believe an equation is dynamical
   zero ISA Parameter WITH default:=0;END;
   value Type Real;
   value :=zero*Base::Time;
  END;

  equations:
    % Mass balance --------
    wse=wsl;   % steam (acc. to Lindahl, MMS uses drho/dt # 0)
    wge=wgl;   % flue gas

    % Energy balance for steam and metal (Lindahl)
    rhosl*Vs*dot(hsl)+Cpm*Mm*dot(Tm)=wse*hse-wsl*hsl+qT;
    Tm = Tsl -(wse*hse-wsl*hsl)*cTslm; % metal temperature (Lindahl)

    % Energy balance for gas --------
    Vg*rhogl*dot(hgl)=k*(wge*hge-wgl*hgl-qT);

    % Heat transfer ------------
    qT = (1-Fsg)*Aht*h*dTmean;

    h=1/(1/hgas+1/hsteam); % conductance between gas and steam
    hgas ISA GasConductance WITH  % from LIBRARY functions
      Dt:=outer::Dt;
      w:=outer::wge;
    END;
    hsteam ISA SteamConductance WITH  % from LIBRARY functions
      Dt:=outer::Dt;
```

```
        w:=outer::wse;
        T:=Tsl;
        p:=psl;
      END;

      % logarithmic mean temperature difference
      dTmean ISA LogMean WITH % from LIBRARY functions
        x:=Tge-Tsl;      % Temp. diff at hot side
        y:=Tgl-Tse;      % Temp. diff at cold side
      END;

      % Fluid mechanics --------------
      rhose*(pse-psl)=abs(wse)*wse/sqr(Cf)+NIL; % pressure drop for water

      pge=pgl;                                  % no pressure drop for gas

      % Fluid properties -----------------
      rhogl=pgl/R/Tgl;            % flue gas density (=rhoge) (ideal gas)

  connections:
    SteamIn.MassFlow   = wse;
    SteamIn.Enthalpy   = hse;
    SteamIn.Pressure   = pse;

    SteamOut.MassFlow  = wsl;
    SteamOut.Enthalpy  = hsl;
    SteamOut.Pressure  = psl;

    GasIn.MassFlow     = wge;
    GasIn.Enthalpy     = hge;
    GasIn.Pressure     = pge;

    GasOut.MassFlow    = wgl;
    GasOut.Enthalpy    = hgl;
    GasOut.Pressure    = pgl;

END; % SuperheaterModel
```

```
% SimpleDrumBoilerModel
% Reference: K J Astrom and R D Bell TFRT-7402 (1988)
% Author: Ulf Holmberg, 1991

LIBRARY boiler; Uses terminals;

BoilerModel ISA Model WITH
 terminals:
  GasIn isa InTerminal with
    content:='Gas;
    Graphic isa Base::Laycut with
      x_pos:=200;
      y_pos:=0;
    end;
  end;
  GasOut isa OutTerminal with
     content:='Gas;
    Graphic isa Base::Layout with
      x_pos:=200;
      y_pos:=300;
    end;
  end;
  WaterIn isa InTerminal with
    content:='Water;
    Graphic isa Base::Layout with
      x_pos:=400;
      y_pos:=90;
    end;
  end;
  SteamOut isa OutTerminal with
    content:='Steam;
    Graphic isa Base::Layout with
      x_pos:=400;
      y_pos:=260;
    end;
  end;
icon:
 Graphic isa Layout with bitmap TYPE String:="Boiler";  end;

 parameters:
  Adrum ISA Parameter WITH default := 20.0; END; % Drum wet area
  Vdrum ISA Parameter WITH default := 40.0; END; % Drum volume
  Vr    ISA Parameter WITH default := 37.0; END; % Riser volume
  Vdc   ISA Parameter WITH default := 19.0; END; % Downcomer volume
  k     ISA Parameter WITH default := 0.01; END; % friction coefficient
%  cp    ISA Parameter WITH default := 1.0;  END; % specific heat of metal
%  m     ISA Parameter WITH default := 0.0;  END; % mass of metal
 variables:
  Pow   ISA Variable; % Power from fuel [W]
  qfw   ISA Variable; % Feedwater flow  [kg/s]
  Tfw   ISA Variable; % Feedwater temp  [deg C]
  qs    ISA Variable; % Steam flow      [kg/s]

  dl    ISA Variable; % Drum level      [m]
  am    ISA Variable; % Steam quality volume ratio
  p     ISA Variable; % Drum pressure   [pa]
  Vw    ISA Variable; % Drum water volume [m^3]
  Vwt   ISA Variable; % Total water volume [m^3]
  xr    ISA Variable; % Steam quality at riser outlet
  Vst   ISA Variable; % Total steam volume [m^3]
%  Tm    ISA Variable; % average metal temperature [deg C]
  qdc   ISA Variable;
  qr    ISA Variable;

  % From Steam Tables

  Hs       ISA Variable WITH value:=H2P(p);END; % Steam enthalphy
  dHsdp    ISA Variable WITH value:=(H2P(p+0.1)-H2P(p))/0.1;END; % dHs/dp
  rs       ISA Variable WITH value:=1/V2P(p);END; % Steam density
  drsdp    ISA Variable WITH value:=(V2P(p+0.1)-V2P(p))/0.1*(-sqr(1/V2P(p)));END; %$
                                                                        ^ rs/dp
  Hw       ISA Variable WITH value:=H1P(p);END; % Water enthalphy
  dHwdp    ISA Variable WITH value:=(H1P(p+0.1)-H1P(p))/0.1;END; % dHw/dp
  rw       ISA Variable WITH value:=1/V1P(p);END; % Water density
  drwdp    ISA Variable WITH value:=(V1P(p+0.1)-V1P(p))/0.1*(-sqr(1/V1P(p)));END; %$
                                                                      ^ drw/dp
  Ts       ISA Variable WITH value:=TP(p);END; % Steam temperature
%  dTsdp    ISA Variable WITH value:=(TP(p+0.1)-TP(p))/0.1;END; % dTs/dp
  Hfw      ISA Variable WITH value=HTP(Tfw,p);END; % Feedwater enthalphy

  Vg  ISA Parameter WITH default:= 1;END; % Volume of flue gas
  rhogl  ISA Variable; % density of gas leaving
  wge ISA Variable;
  wgl ISA Variable;
  pge ISA Variable;
  pgl ISA Variable;
  Tge ISA Variable;
  Tgl ISA Variable;
  hge isa variable with value=HTP(Tge,101325.0,gasmix)+NIL;end;
  hgl isa variable with value=HTP(Tgl,101325.0,gasmix);end;
  gasmix Type Row [7] := [0.6911, 0.078, 0.1697, 0., 0., 0., 0.0341];
                         % [   N2 ,  CO2 , O2    , Ar,SO2,NO ,  H2O  ]
  R ISA Variable WITH value=RTP(Tgl,pgl,gasmix);END; % J/kgK, gas const/mole weight

  NIL ISA Model WITH  % Trick to fool DASSL believe an equation is dynamical
   zero ISA Parameter WITH default:=0;END;
   value Type Real;
   value :=zero*Base::Time;
  END;
  Cpow isa parameter with default:=2.5069e7;end;

  % auxiliary variables
   damdx,Hc,e11,e12,e13,e21,e22,e23,e31,e32,e33,b1,b2,b3 ISA Variable;
   p1,p2,p3,e221,e231,e321,e331,e332,b21,b31,b32 ISA Variable;
realizations:
  equations:
   Vst = Vdrum - Vw + am*Vr;   % Total steam volume
   Vwt = Vw + Vdc + (1-am)*Vr; % Total water volume

   % E-matrix in the description Edot(x)=B(x,u)
   e11=Vst*(Hs*drsdp+rs*dHsdp)+Vwt*(Hw*drwdp+rw*dHwdp);
   e12=Hw*rw-Hs*rs;
   e13=(Hs*rs-Hw*rw)*Vr*damdx;
   b1=Pow+qfw*Hfw-qs*Hs;
   e21=Vst*drsdp+Vwt*drwdp;
   e22=rw-rs;
   e23=(rs-rw)*Vr*damdx;
   b2=qfw-qs;
   Hc=Hs-Hw;
   e31=((1-xr)*Hc*drsdp+rs*dHsdp)*am*Vr+(rw*dHwdp-xr*Hc*drwdp)*(1-am)*Vr;
   e32=0;
   e33=((1-xr)*rs+xr*rw)*Hc*Vr*damdx;
   b3=Pow-qdc*xr*Hc;

   % solve linear equation for derivatives of state vector
   p1=e21/e11;
   e221=e22-e12*p1;
   e231=e23-e13*p1;
   b21=b2-b1*p1;

   p2=e31/e11;
```

```
    e321=-e12*p2;
    e331=e33-e13*p2;
    b31=b3-b1*p2;

    p3=e321/e221;
    e332=e331-e231*p3;
    b32=b31-b21*p3;

    % state equations
    dot(xr)=b32/e332;
    dot(Vw)=(b21-e231*dot(xr))/e221;
    dot(p)=(b1-e12*dot(Vw)-e13*dot(xr))/e11;

    % Riser flow
    qr=qdc-(am*drsdp+(1-am)*drwdp)*Vr*dot(p)+(rw-rs)*Vr*damdx*dot(xr);

    % Momentum balance for Downcomer
    % ------------------------------
%    am*Vr*(rw-rs)=0.5*k*abs(qdc)*qdc;    % where k= friction coeff.
    qdc=sqrt(2.0*am*Vr*(rw-rs)/k);

    % Average steam-water volume ratio in the risers (am)
    % ---------------------------------------------------
    am=rw/(rw-rs)*(1-rs/(rw-rs)/xr*ln(1+(rw/rs-1)*xr));
    damdx=-am/xr+rw/(rs+(rw-rs)*xr);

    dl=(Vw+am*Vr)/Adrum; % Drum water level

    %  --------Gas side----------
    wgl=wge;    % mass balance
    pgl=pge;    % no pressure drop for gas

    % Energy balance for gas --------
    Vg*rhogl*dot(hgl)=1.4*(wge*hge-wgl*hgl-Pow);  % 1.4=cp/cv

    % Heat transfer from gas to riser steam
    Pow=Cpow*(Tgl-Ts);

    % Fluid properties ----------------
    rhogl=pgl/R/Tgl;            % flue gas density (=rhoge) (ideal gas)

  connections:
    WaterIn.MassFlow   = qfw;
    WaterIn.Enthalpy   = Hfw;
    WaterIn.Pressure   = p;

    SteamOut.MassFlow  = qs;
    SteamOut.Enthalpy  = Hs;
    SteamOut.Pressure  = p;

    GasIn.MassFlow     = wge;
    GasIn.Enthalpy     = hge;
    GasIn.Pressure     = pge;

    GasOut.MassFlow    = wgl;
    GasOut.Enthalpy    = hgl;
    GasOut.Pressure    = pgl;

END; % BoilerModel
```

```
% SimpleDrumBoilerModel
% Reference: K J Astrom and R D Bell TFRT-7402 (1988)
% Author: Ulf Holmberg, 1991

LIBRARY SimpleDrumBoilerModel;

BoilerModel ISA Model WITH
 terminals:
 parameters:
  Adrum ISA Parameter WITH default := 20.0; END; % Drum wet area
  Vdrum ISA Parameter WITH default := 40.0; END; % Drum volume
  Vr    ISA Parameter WITH default := 37.0; END; % Riser volume
  Vdc   ISA Parameter WITH default := 19.0; END; % Downcomer volume
  k     ISA Parameter WITH default := 0.01; END; % friction coefficient
  cp    ISA Parameter WITH default := 1.0;  END; % specific heat of metal
  m     ISA Parameter WITH default := 0.0;  END; % mass of metal
 variables:
  Pow   ISA Variable; % Power from fuel [MW]
  qfw   ISA Variable; % Feedwater flow  [kg/s]
  Tfw   ISA Variable; % Feedwater temp  [K]
  qs    ISA Variable; % Steam flow      [kg/s]

  dl    ISA Variable; % Drum level      [m]
  am    ISA Variable; % Steam quality volume ratio
  p     ISA Variable; % Drum pressure   [Pa]
  Vw    ISA Variable; % Drum water volume [m^3]
  Vwt   ISA Variable; % Total water volume [m^3]
  xr    ISA Variable; % Steam quality at riser outlet
  Vst   ISA Variable; % Total steam volume [m^3]
  Tm    ISA Variable; % Average metal temperature [deg C]
  qdc   ISA Variable; % Downcomer flow [kg/s]
  qr    ISA Variable; % Riser flow [kg/s]

  % From Steam Tables
  Hs       ISA Variable WITH value=H2P(p);END;     % Steam enthalpy
  rs       ISA Variable WITH value=1/V2P(p);END;   % Steam density
  Hw       ISA Variable WITH value=H1P(p);END;     % Water enthalpy
  rw       ISA Variable WITH value=1/V1P(p);END;   % Water density
  Ts       ISA Variable WITH value=TP(p);END;      % Steam temperature
  Hfw      ISA Variable WITH value=HTP(Tfw,p);END; % Feedwater enthalpy

realizations:
   equations:
   % Drum energy balance
   dot(rs*Vst*Hs+rw*Vwt*Hw+m*cp*Tm)=Pow+qfw*Hfw-qs*Hs;
     Vst = Vdrum - Vw + am*Vr;              % Total steam volume
     Vwt = Vw + Vdc + (1-am)*Vr;            % Total water volume
     Tm  = Ts;       % metal temp=sat. steam temp corresponing to p

   % Drum mass balance
   dot(rw*Vst+rw*Vwt)=qfw-qs;

   % Riser energy balance
   dot(rs*am*Vr*Hs+rw*(1-am)*Vr*Hw)=Pow+qdc*Hw-xr*qr*Hs-(1-xr)*qr*Hw;

   % Riser mass balance
   dot(am*Vr*rs+(1-am)*Vr*rw)=qdc-qr;

   % Momentum balance for Downcomer
   am*Vr*(rw-rs)=0.5*k*abs(qdc)*qdc;

   % Average steam-water volume ratio in the risers (am)
   am=rw/(rw-rs)*(1-rs/(rw-rs)/xr*ln(1+(rw/rs-1)*xr));

   dl=(Vw+am*Vr)/Adrum; % Drum water level

END; % BoilerModel

% Test of boiler

LIBRARY System; USES SimpleDrumBoilerModel;
Boiler ISA BoilerModel WITH
 parameters:
  t1   ISA Parameter with default:=5.0; end;
  t2   ISA Parameter with default:=1000.0; end;
  Pow1 ISA Parameter with default:=200.0E6; end;
  qs1  ISA Parameter with default:=130.7486; end;
 variables:
  t ISA Variable; % time
  t=Base::Time;
  % initials
  p.initial:=7.576E6;
  Vw.initial:=13.7711;
  xr.initial:=0.092354;
  am.initial:=0.425462;
  qdc.initial:=1473.771;
  qr.initial:=1473.771;
  rs.initial:=39.9378;
  rw.initial:=729.809;
  %Pow=If t<t1 then Pow1 else 220.0E6 end;
  Pow=Pow1;
  qfw=130.7486;
  %qs=if t<t2 then qs1 else 140 end;
  qs=qs1;
  Tfw=280.0+273.15; % K
END;
```

```
% SimpleDrumBoilerModel
% Reference: K J Astrom and R D Bell TFRT-7402 (1988)
% Author: Ulf Holmberg, 1991

LIBRARY boiler; Uses terminals;

BoilerModel ISA Model WITH
 terminals:
  GasIn isa InTerminal with
    content:='Gas;
    Graphic isa Base::Layout with
      x_pos:=200;
      y_pos:=0;
    end;
  end;
  GasOut isa OutTerminal with
     content:='Gas;
    Graphic isa Base::Layout with
      x_pos:=200;
      y_pos:=300;
    end;
  end;
  WaterIn isa InTerminal with
    content:='Water;
    Graphic isa Base::Layout with
      x_pos:=400;
      y_pos:=90;
    end;
  end;
  SteamOut isa OutTerminal with
    content:='Steam;
    Graphic isa Base::Layout with
      x_pos:=400;
      y_pos:=260;
    end;
  end;
icon:
 Graphic isa Layout with bitmap TYPE String:="Boiler";  end;
 parameters:
  Adrum ISA Parameter WITH default := 20.0; END; % Drum wet area
  Vdrum ISA Parameter WITH default := 40.0; END; % Drum volume
  Vr    ISA Parameter WITH default := 37.0; END; % Riser volume
  Vdc   ISA Parameter WITH default := 19.0; END; % Downcomer volume
  k     ISA Parameter WITH default := 0.01; END; % friction coefficient
  cp    ISA Parameter WITH default := 1.0;  END; % specific heat of metal
  m     ISA Parameter WITH default := 0.0;  END; % mass of metal
 variables:
  Pow   ISA Variable; % Power from fuel [MW]
  qfw   ISA Variable; % Feedwater flow  [kg/s]
  Tfw   ISA Variable; % Feedwater temp  [K]
  qs    ISA Variable; % Steam flow      [kg/s]

  dl    ISA Variable; % Drum level      [m]
  am    ISA Variable with initial:=0.425459;end; % Steam quality volume ratio
  p     ISA Variable; % Drum pressure   [Pa]
  Vw    ISA Variable; % Drum water volume [m^3]
  Vwt   ISA Variable; % Total water volume [m^3]
  xr    ISA Variable; % Steam quality at riser outlet
  Vst   ISA Variable; % Total steam volume [m^3]
  Tm    ISA Variable; % average metal temperature [deg C]
  qdc   ISA Variable; % Downcomer flow [kg/s]
  qr    ISA Variable; % Riser flow [kg/s]

  % From Steam Tables
  Hs       ISA Variable WITH value:=H2P(p);END; % Steam enthalphy
  dHsdp    ISA Variable WITH value:=(H2P(p+0.1)-H2P(p))/0.1;END; % dHs/dp
  rs       ISA Variable WITH value:=1/V2P(p);END; % Steam density
  drsdp    ISA Variable WITH value:=(V2P(p+0.1)-V2P(p))/0.1*(-sqr(1/V2P(p)));END; %$
                                                                        ^ rs/dp
  Hw       ISA Variable WITH value:=H1P(p);END; % Water enthalphy
  dHwdp    ISA Variable WITH value:=(H1P(p+0.1)-H1P(p))/0.1;END; % dHw/dp
  rw       ISA Variable WITH value:=1/V1P(p);END; % Water density
  drwdp    ISA Variable WITH value:=(V1P(p+0.1)-V1P(p))/0.1*(-sqr(1/V1P(p)));END; %$
                                                                      ^ drw/dp
  Ts       ISA Variable WITH value:=TP(p);END; % Steam temperature
  dTsdp    ISA Variable WITH value:=(TP(p+0.1)-TP(p))/0.1;END; % dTs/dp
  Hfw      ISA Variable WITH value=HTP(Tfw,p);END; % Feedwater enthalphy

  Vg  ISA Parameter WITH default:= 1;END; % Volume of flue gas
  rhogl  ISA Variable; % density of gas leaving
  wge ISA Variable;
  wgl ISA Variable;
  pge ISA Variable;
  pgl ISA Variable;
  Tge ISA Variable;
  Tgl ISA Variable;
  hge isa variable with value=HTP(Tge,101325.0,gasmix)+NIL;end;
  hgl isa variable with value=HTP(Tgl,101325.0,gasmix);end;
  gasmix Type Row [7] := [0.6911, 0.078, 0.1697, 0., 0., 0., 0.0341];
                         % [   N2 ,  CO2 , O2    , Ar,SO2,NO ,  H20  ]
  R ISA Variable WITH value=RTP(Tgl,pgl,gasmix);END; % J/kgK, gas c./mole w.
  NIL ISA Model WITH  % Trick to fool DASSL believe an equation is dynamical
     zero ISA Parameter WITH default:=0;END;
     value Type Real;
     value :=zero*Base::Time;
  END;
  Cpow isa parameter with default:=2.5069e7;end;

  % Temporary auxiliary variables
   dHs,dHw,drs,drw,dTs,dVst,dVwt,dam,damdxr ISA Variable;
realizations:
   equations:
%   dot(rs*Hs*Vst+rw*Hw*Vwt+m*cp*Tm)=Pow+qfw*Hfw-qs*Hs;
    DrumEnergyBalance ISA SetOfDAE WITH
     dEdrum,dEs,dEw,dEm ISA Variable;
     %-------------------------------------------------------------
     dEdrum=Pow+qfw*Hfw-qs*Hs;              % Drum energy balance
     %-------------------------------------------------------------
     dEdrum=dEs+dEw+dEm;
     dEs= drs*Hs*Vst+rs*dHs*Vst+rs*Hs*dVst; % steam part
     dEw= drw*Hw*Vwt+rw*dHw*Vwt+rw*Hw*dVwt; % water part
     dEm= m*cp*dTs;                         % metal part
     Vst = Vdrum - Vw + am*Vr;              % Total steam volume
     Vwt = Vw + Vdc + (1-am)*Vr;            % Total water volume
     Tm  = Ts;       % metal temp=sat. steam temp corresponing to p
    END; % DrumEnergyBalance

%   dot(rs*Vst+rw*Vwt)=qfw-qs; % global mass balance
    DrumMassBalance ISA SetOfDAE WITH
     dMdrum,dMs,dMw ISA Variable;
     %-------------------------------------------------------------
     dMdrum=qfw-qs;                         % Rate of mass in drum
     %-------------------------------------------------------------
     dMdrum=dMs+dMw;
     dMs=drs*Vst+rs*dVst;                   % steam part
     dMw=drw*Vwt+rw*dVwt;                   % water part
    END; % DrumMassBalance

%   dot(rs*Hs*am*Vr+rw*Hw*(1-am)*Vr)=Pow+qdc*Hw-xr*qr*Hs-(1-xr)*qr*Hw;
    RiserEnergyBalance ISA SetOfDAE WITH
```

```
      dEriser,dErs,dErw ISA Variable;
      %-------------------------------------------------------
      dEriser=Pow+qdc*Hw-xr*qr*Hs-(1-xr)*qr*Hw;  % Riser energy balance
      %-------------------------------------------------------
      dEriser=dErs+dErw;
      dErs=drs*Hs*am*Vr+rs*dHs*am*Vr+rs*Hs*dam*Vr;            % steam part
      dErw=drw*Hw*(1-am)*Vr+rw*dHw*(1-am)*Vr+rw*Hw*(-dam)*Vr; % water part
     END; % RiserEnergyBalance

%    dot(rs*am*Vr+rw*(1-am)*Vr)=qdc-qr;
     RiserMassBalance ISA SetOfDAE WITH
      dMriser,dMrs,dMrw ISA Variable;
      %-------------------------------------------------------
      dMriser=qdc-qr;                          % Riser mass balance
      %-------------------------------------------------------
      dMriser=dMrs+dMrw;
      dMrs=drs*am*Vr+rs*dam*Vr;
      dMrw=drw*(1-am)*Vr+rw*(-dam)*Vr;
     END; % RiserMassBalance

     % Momentum balance for Downcomer
      am*Vr*(rw-rs)=0.5*k*abs(qdc)*qdc;

     % Average steam-water volume ratio in the risers (am)
     am=rw/(rw-rs)*(1-rs/(rw-rs)/xr*ln(1+(rw/rs-1)*xr));
     damdxr=-am/xr+rw/(rs+(rw-rs)*xr);
     dam=damdxr*dot(xr);

     dl=(Vw+am*Vr)/Adrum; % Drum water level

     %  --------Gas side----------
     wgl=wge;    % mass balance
     pgl=pge;    % no pressure drop for gas

     % Energy balance for gas --------
     Vg*rhogl*dot(hgl)=1.4*(wge*hge-wgl*hgl-Pow);  % 1.4=cp/cv

     % Heat transfer from gas to riser steam
     Pow=Cpow*(Tgl-Ts);

     % Fluid properties ----------------
     rhogl=pgl/R/Tgl;            % flue gas density (=rhoge) (ideal gas)

     % Partial derivatives
     dHs=dHsdp*dot(p);
     drs=drsdp*dot(p);
     dHw=dHwdp*dot(p);
     drw=drwdp*dot(p);
     dTs=dTsdp*dot(p);
     dVst=-dot(Vw)+Vr*damdxr*dot(xr);
     dVwt=-dVst;

   connections:
     WaterIn.MassFlow    = qfw;
     WaterIn.Enthalpy    = Hfw;
     WaterIn.Pressure    = p;

     SteamOut.MassFlow   = qs;
     SteamOut.Enthalpy   = Hs;
     SteamOut.Pressure   = p;

     GasIn.MassFlow      = wge;
     GasIn.Enthalpy      = hge;
     GasIn.Pressure      = pge;

     GasOut.MassFlow     = wgl;
     GasOut.Enthalpy     = hgl;
     GasOut.Pressure     = pgl;

   END; % BoilerModel
```

```
% SimpleDrumBoilerModel
% Reference: K J Astrom and R D Bell TFRT-7402 (1988)
% Author: Ulf Holmberg, 1991

LIBRARY SimpleDrumBoilerModel;

BoilerModel ISA Model WITH
 terminals:
 parameters:
  Adrum ISA Parameter WITH default := 20.0; END; % Drum wet area
  Vdrum ISA Parameter WITH default := 40.0; END; % Drum volume
  Vr    ISA Parameter WITH default := 37.0; END; % Riser volume
  Vdc   ISA Parameter WITH default := 19.0; END; % Downcomer volume
  k     ISA Parameter WITH default := 0.01; END; % friction coefficient
  cp    ISA Parameter WITH default := 1.0;  END; % specific heat of metal
  m     ISA Parameter WITH default := 0.0;  END; % mass of metal
 variables:
  Pow   ISA Variable; % Power from fuel [MW]
  qfw   ISA Variable; % Feedwater flow  [kg/s]
  Tfw   ISA Variable; % Feedwater temp  [K]
  qs    ISA Variable; % Steam flow      [kg/s]

  dl    ISA Variable; % Drum level      [m]
  am    ISA Variable; % Steam quality volume ratio
  p     ISA Variable; % Drum pressure   [Pa]
  Vw    ISA Variable; % Drum water volume [m^3]
  Vwt   ISA Variable; % Total water volume [m^3]
  xr    ISA Variable; % Steam quality at riser outlet
  Vst   ISA Variable; % Total steam volume [m^3]
  Tm    ISA Variable; % average metal temperature [deg C]
  qdc   ISA Variable; % Downcomer flow [kg/s]
  qr    ISA Variable; % Riser flow [kg/s]

  % From Steam Tables
  Hs       ISA Variable WITH value=H2P(p);END;     % Steam enthalphy
  rs       ISA Variable WITH value=1/V2P(p);END;   % Steam density
  Hw       ISA Variable WITH value=H1P(p);END;     % Water enthalphy
  rw       ISA Variable WITH value=1/V1P(p);END;   % Water density
  Ts       ISA Variable WITH value=TP(p);END;      % Steam temperature
  Hfw      ISA Variable WITH value=HTP(Tfw,p);END; % Feedwater enthalphy

realizations:
   equations:
   % Drum energy balance
   dot(rs*Vst*Hs+rw*Vwt*Hw+m*cp*Tm)=Pow+qfw*Hfw-qs*Hs;
     Vst = Vdrum - Vw + am*Vr;              % Total steam volume
     Vwt = Vw + Vdc + (1-am)*Vr;            % Total water volume
     Tm  = Ts;       % metal temp=sat. steam temp corresponing to p

   % Drum mass balance
   dot(rw*Vst+rw*Vwt)=qfw-qs;

   % Riser energy balance
   dot(rs*am*Vr*Hs+rw*(1-am)*Vr*Hw)=Pow+qdc*Hw-xr*qr*Hs-(1-xr)*qr*Hw;

   % Riser mass balance
   dot(am*Vr*rs+(1-am)*Vr*rw)=qdc-qr;

   % Momentum balance for Downcomer
   am*Vr*(rw-rs)=0.5*k*abs(qdc)*qdc;

   % Average steam-water volume ratio in the risers (am)
   am=rw/(rw-rs)*(1-rs/(rw-rs)/xr*ln(1+(rw/rs-1)*xr));

   dl=(Vw+am*Vr)/Adrum; % Drum water level

END; % BoilerModel

% Test of boiler

LIBRARY System; USES SimpleDrumBoilerModel;
Boiler ISA BoilerModel WITH
 parameters:
  t1   ISA Parameter with default:=5.0; end;
  t2   ISA Parameter with default:=1000.0; end;
  Pow1 ISA Parameter with default:=200.0E6; end;
  qs1  ISA Parameter with default:=130.7486; end;
 variables:
  t ISA Variable; % time
  t=Base::Time;
  % initials
  p.initial:=7.576E6;
  Vw.initial:=13.7711;
  xr.initial:=0.092354;
  am.initial:=0.425462;
  qdc.initial:=1473.771;
  qr.initial:=1473.771;
  rs.initial:=39.9378;
  rw.initial:=729.809;
  %Pow=If t<t1 then Pow1 else 220.0E6 end;
  Pow=Pow1;
  qfw=130.7486;
  %qs=if t<t2 then qs1 else 140 end;
  qs=qs1;
  Tfw=280.0+273.15; % K
END;
```

```
library System; Uses terminals, functions, economizer, boiler, superheater;

Heat_Recovery_Steam_Generator isa model with
  Graphic ISA Base::Layout;
  C6 ISA Base::Connection WITH
    economizer.GasIn AT boiler.GasOut;
    bpoints TYPE Matrix [2, 2] := [75.0, 209.0; 75.0, 181.0];
  END;
  C7 ISA Base::Connection WITH
    boiler.GasIn AT superheater.GasOut;
    bpoints TYPE Matrix [2, 2] := [75.0, 118.0; 75.0, 90.0];
  END;
  C8 ISA Base::Connection WITH
    economizer.WaterOut AT boiler.WaterIn;
    bpoints TYPE Matrix [5, 2] := [90.0, 215.0; 100.0, 215.0; 127.0, 215.0; 127.0,$
                                                          ^ 137.0; 106.0, 137.0];
  END;
  C9 ISA Base::Connection WITH
    boiler.SteamOut AT superheater.SteamIn;
    bpoints TYPE Matrix [4, 2] := [106.0, 173.0; 144.0, 173.0; 144.0, 84.0; 90.0, 84.0];
  END;

submodels:
 Economizer isa EconomizerModel with
  Graphic isa super::Graphic with
   x_pos:=75;
   y_pos:=225;
  end;
    wwePAR ISA Parameter WITH default:= 130.749; END;
    hwePAR ISA Parameter WITH default:= 1.226E6; END;
    pwePAR ISA Parameter WITH default:= 7.60112E6; END;

    pwlPAR ISA Parameter WITH default:= 7.576E6; END;

    wgePAR ISA Parameter WITH default:= 640; END;
    hgePAR ISA Parameter WITH default:= 403011; END;

    pglPAR ISA Parameter WITH default:= 101325; END;

    WaterIn.MassFlow   = wwePAR; % kg/s
    WaterIn.Enthalpy   = hwePAR;  % J/kg

%    WaterOut.Pressure   = pwlPAR;              % take away when connecting

%    GasIn.MassFlow     = wgePAR;  % kg/s      % take away when connecting
%    GasIn.Enthalpy     = hgePAR;              % take away when connecting

    GasOut.Pressure    = pglPAR; % N/m^2, atmospheric pressure for gas

    % initial conditions
    hwe.initial := 1.226E6;
    pwe.initial := 7.60112E6;
    Twe.initial := 551.206;

    hwl.initial := 1.236E6;
    pwl.initial := 7.5765E6;
    Twl.initial := 553.113;

    hge.initial := 403011;
    Tge.initial := 572.318;
    hgl.initial := 401005;
    Tgl.initial := 570.473;

    rhowe.initial:= 756.186;
end;

 Boiler isa BoilerModel with
  Graphic isa super::Graphic with
   x_pos:=75;
   y_pos:=150;
  end;
    wwePAR ISA Parameter WITH default:= 130.7486; END;
    hwePAR ISA Parameter WITH default:= 1.23619E6; END;
    pwePAR ISA Parameter WITH default:= 130000; END;

    wslPAR ISA Parameter WITH default:= 130.7486; END;
    hslPAR ISA Parameter WITH default:= 8398.9; END;
    pslPAR ISA Parameter WITH default:= 130000; END;

    wgePAR ISA Parameter WITH default:= 640; END;
    hgePAR ISA Parameter WITH default:= 715511; END;
    pglPAR ISA Parameter WITH default:= 101325; END;

%    WaterIn.MassFlow   = wwePAR; % kg/s    % take away when connecting
%    WaterIn.Enthalpy   = hwePAR;  % J/kg   % take away when connecting

%    SteamOut.MassFlow   = wslPAR; % kg/s   % take away when connecting

%    GasIn.MassFlow     = wgePAR;  % kg/s   % take away when connecting
%    GasIn.Enthalpy     = hgePAR;           % take away when connecting

%    GasOut.Pressure    = pglPAR; % N/m^2,   % take away when connecting

    % initials
    p.initial:=7.576E6;
    Vw.initial:=13.7711;
    xr.initial:=0.0940547;
    hgl.initial:=403011;
    Tfw.initial:=553.15;
    Tge.initial:=850;
    Tgl.initial:=572.318;
    R.initial:=290;
end;

 Superheater isa SuperheaterModel with
  Graphic isa super::Graphic with
   x_pos:=75;
   y_pos:=75;
  end;
    wsePAR ISA Parameter WITH default:= 130.749; END;
    hsePAR ISA Parameter WITH default:= 2.76587E6; END;
    psePAR ISA Parameter WITH default:= 7.576E6; END;

    wslPAR ISA Parameter WITH default:= 130.749; END;

    wgePAR ISA Parameter WITH default:= 640; END;
    hgePAR ISA Parameter WITH default:= 7.36E5; END;

    pglPAR ISA Parameter WITH default:= 101325; END;

%    SteamIn.Enthalpy   = hsePAR;   % take away when connectin
%    SteamIn.Pressure   =  psePAR;  % take away when connectin

    SteamOut.MassFlow   = wslPAR;

    GasIn.MassFlow = wgePAR;
    GasIn.Enthalpy = hgePAR;

%    GasOut.Pressure = pglPAR;      % take away when connectin

    % initial conditions
```

```
    hse.initial := 2.76587E6;
    pse.initial := 7.576E6;
    Tse.initial := 564.34;

    hsl.initial := 2.85613E6;
    psl.initial := 7.1004E6;
    Tsl.initial := 578.231;

    hge.initial := 7.36E5;
    Tge.initial := 836.64;
    hgl.initial := 68156D;
    Tgl.initial := 820.7D5;

    Tm.initial  := 625.428;
    rhose.initial:= 39.9384;
  end;
END;
```