Simulink Report: Vol_Isobaric_2D_

Christian Müller

2008-03-06

Model - Vol_Isobaric_2D_

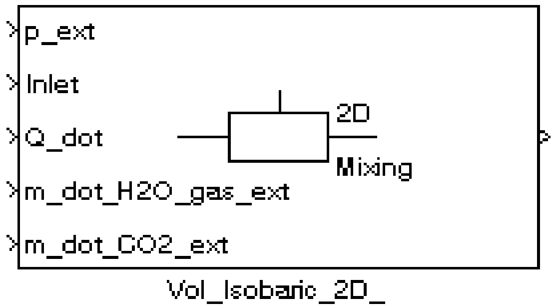


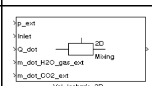
Tabelle 1.1. Vol_Isobaric_2D_ Simulation Parameters

Solver ode14x	*ZeroCross* on	*StartTime* 0.0*StopTime* 10.0
RelTol 1e-3	*AbsTol* auto	*Refine* 1
InitialStep auto	*FixedStep* auto	*MaxStep* auto

Tabelle 1.2. Vol_Isobaric_2D_ Summary Information

NumModelInputs	N/A	*NumModelOutputs*	N/A
NumVirtualSubsystems	N/A	*NumNonvirtSubsystems*	N/A
NumNonVirtBlocksInModel	N/A	*NumBlockTypeCounts*	N/A
NumBlockSignals	N/A	*NumBlockParams*	N/A
NumZCEvents	N/A	*NumNonsampledZCs*	N/A

Systems

Name	Parent	Snapshot	Blocks	Signals
Vol_Isobaric_2D_	<root>		Vol_Isobaric_2D_	Vol_Isobaric_2D_<1>

Blocks

Tabelle 1.3. Block Type Count

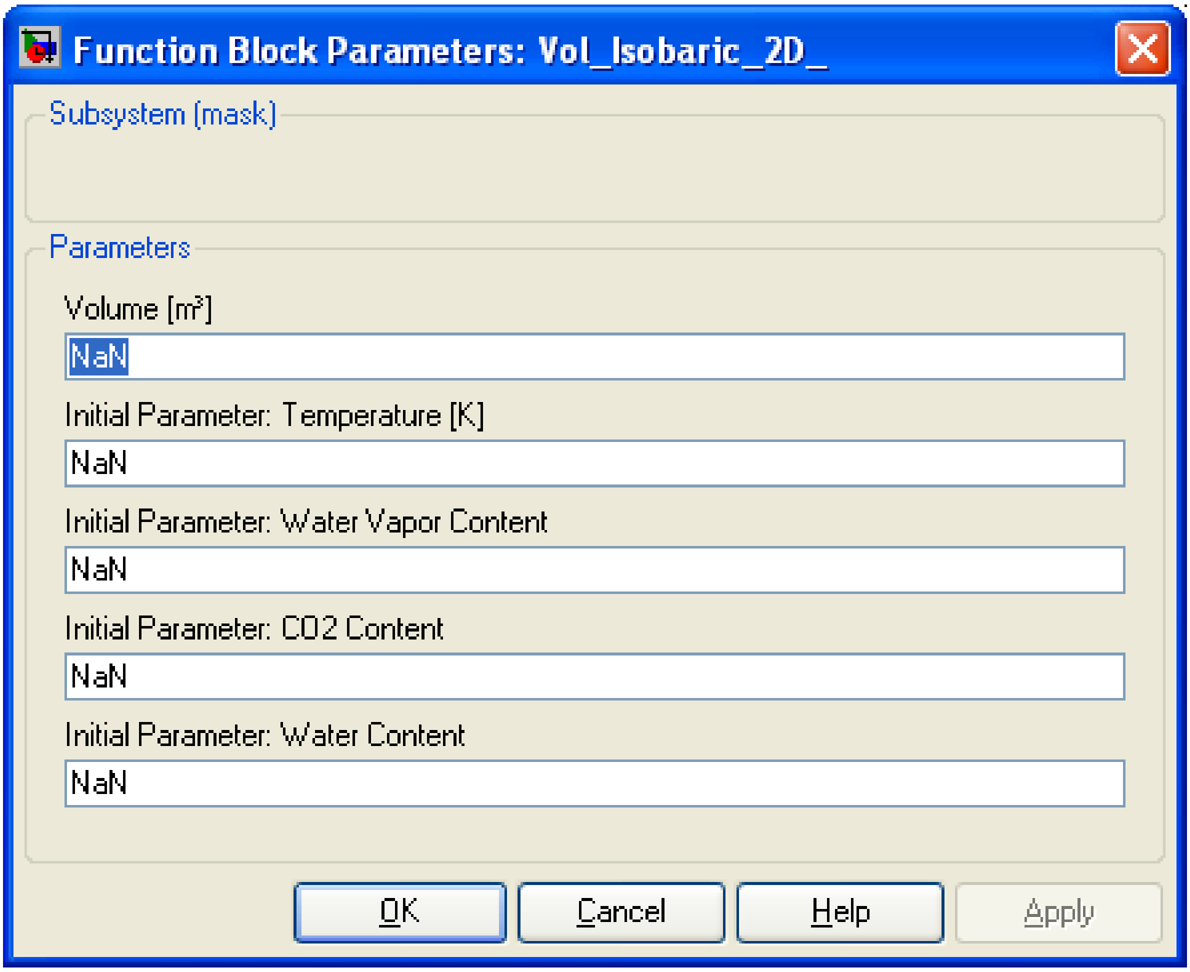
BlockType	Count	Block Names
Inport	34	p_vol, In, Q_dot, m_dot_H2O_gas_human, m_dot_CO2_human, V, p, T_init, x_H2O_gas_init, x_CO2_init, x_H2O_liq_init, T, m_air, x_H2O_gas, x_CO2, x_H2O_liq, T_in, m_dot_air_in, x_H2O_gas_in, x_CO2_in, x_H2O_liq_in, Q_dot,

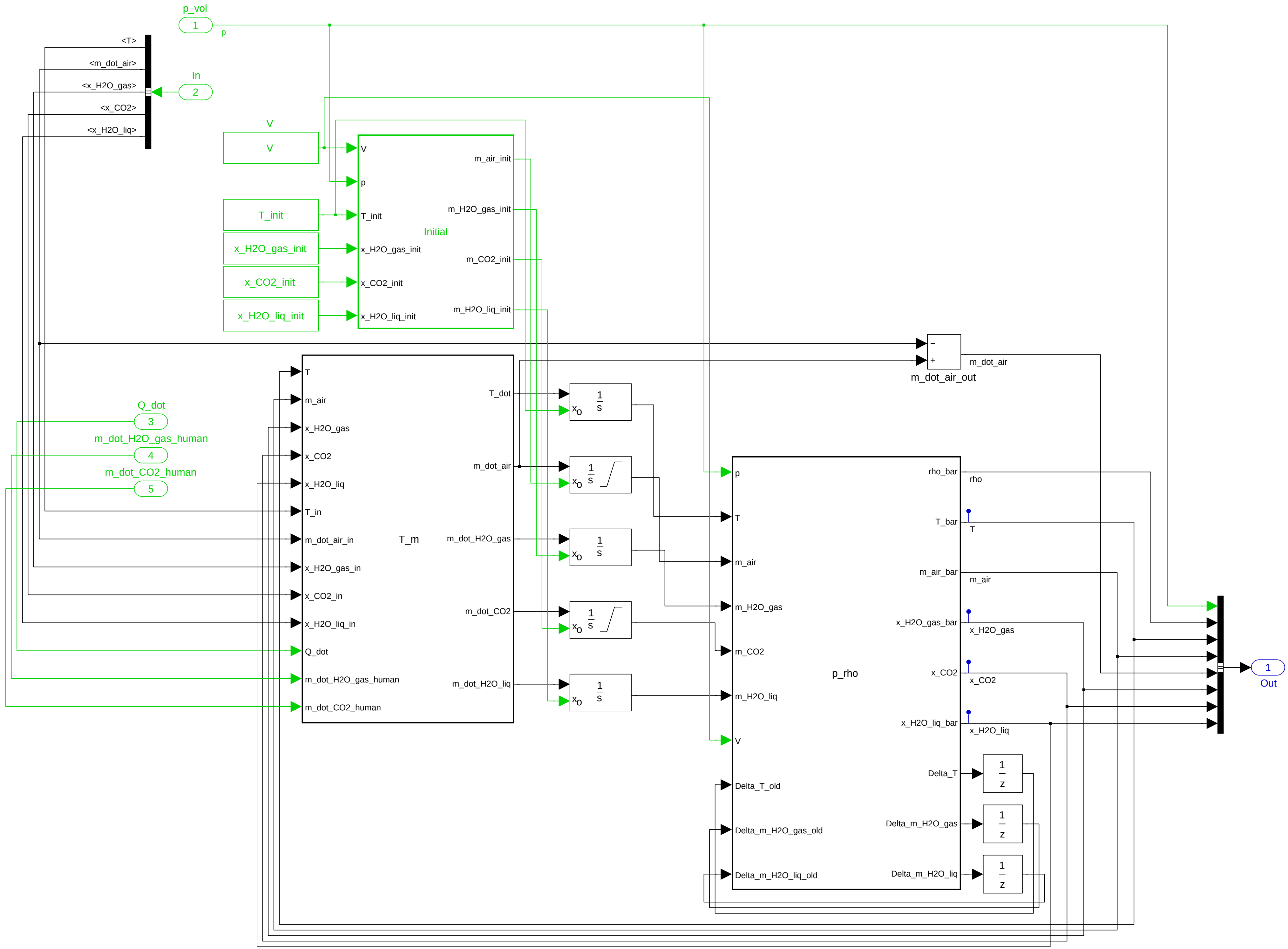
BlockType	Count	Block Names
		m_dot_H2O_gas_human, m_dot_CO2_human, p, T, m_air, m_H2O_gas, m_CO2, m_H2O_liq, V, Delta_T_old, Delta_m_H2O_gas_old, Delta_m_H2O_liq_old
Outport	19	m_air_init, m_H2O_gas_init, m_CO2_init, m_H2O_liq_init, T_dot, m_dot_air, m_dot_H2O_gas, m_dot_CO2, m_dot_H2O_liq, rho_bar, T_bar, m_air_bar, x_H2O_gas_bar, x_CO2, x_H2O_liq_bar, Delta_T, Delta_m_H2O_gas, Delta_m_H2O_liq, Out
Integrator	5	Integrator2, Integrator3, Integrator4, Integrator5, Integrator6
Constant	5	T_init, V, x_CO2_init, x_H2O_gas_init, x_H2O_liq_init
UnitDelay	3	Unit Delay, Unit Delay1, Unit Delay2
Terminator	3	Terminator , Terminator , Terminator
Stateflow (m)	3	Embedded MATLAB Function, Embedded_MATLAB_Function_1, Embedded_MATLAB_Function_2
S-Function	3	SFunction , SFunction , SFunction
Demux	3	Demux , Demux , Demux
Sum	1	m_dot_air_out
SubSystem	1	Vol_Isobaric_2D_
BusSelector	1	Bus Selector6
BusCreator	1	Bus Creator2

Data and Functions

Tabelle 1.4. Model Functions

Function Name	Parent Blocks	Calling string
NaN	Vol_Isobaric_2D_	`NaN`


Function Block Parameters: Vol_Isobaric_2D_
Subsystem (mask)
Parameters
Volume [m³]
NaN
Initial Parameter: Temperature [K]
NaN
Initial Parameter: Water Vapor Content
NaN
Initial Parameter: CO2 Content
NaN
Initial Parameter: Water Content
NaN
OK
Cancel
Help
Apply


p_vol
1
p
<T>
<m_dot_air>
<x_H2O_gas>
<x_CO2>
<x_H2O_liq>
In
2
V
V
T_init
x_H2O_gas_init
x_CO2_init
x_H2O_liq_init
V
p
T_init
x_H2O_gas_init
x_CO2_init
x_H2O_liq_init
Initial
m_air_init
m_H2O_gas_init
m_CO2_init
m_H2O_liq_init
m_dot_air
m_dot_air_out
Q_dot
3
m_dot_H2O_gas_human
4
m_dot_CO2_human
5
T
m_air
x_H2O_gas
x_CO2
x_H2O_liq
T_in
m_dot_air_in
x_H2O_gas_in
x_CO2_in
x_H2O_liq_in
Q_dot
m_dot_H2O_gas_human
m_dot_CO2_human
T_m
T_dot
m_dot_air
m_dot_H2O_gas
m_dot_CO2
m_dot_H2O_liq
1
s
p
T
m_air
m_H2O_gas
m_CO2
m_H2O_liq
V
Delta_T_old
Delta_m_H2O_gas_old
Delta_m_H2O_liq_old
p_rho
rho_bar
rho
T_bar
T
m_air_bar
m_air
x_H2O_gas_bar
x_H2O_gas
x_CO2
x_CO2
x_H2O_liq_bar
x_H2O_liq
Delta_T
Delta_m_H2O_gas
Delta_m_H2O_liq
1
z
1
Out

```
function [m_air_init,m_H2O_gas_init,m_CO2_init,m_H2O_liq_init]= Initial(V,p,T_init,↙
x_H2O_gas_init,x_CO2_init,x_H2O_liq_init)

% *************************************************************************
% * Definition of a isobaric with 2 apertures
% *
% * Number of Inputs:        5
% *
% * Parameter : Volume:      V
% *
% *
% * Relevant input variables of Vol_Isobaric_2D:
% *
% * Temperature:             T
% * Mass flow dry air:       m_dot_air
% * Content water vapor:     x_H2O_gas
% * Content CO2:             x_CO2
% * Content water:           x_H2O_liq
% * External Heat Flow:      Q_dot
% *
% *
% * Relevant output variables of Vol_Isobaric_2D:
% *
% * Pressure:                p
% * Density:                 rho
% * Temperature:             T
% * Mass dry air:            m_air
% * Content water vapor:     x_H2O_gas
% * Content CO2:             x_CO2
% * Content water:           x_H2O_liq
% *
% ************************************************************************
% * Embedded MATLAB Function Initial:
% *
% * Calculations:
% * 1. Definition specific gas constants.
% * 2. Calculation of the saturation density by a given temperature.
% * 3. Calculation of the saturation mass.
% * 4. Initialisation.
% *
% *
% *
% * Assumptions:
% * 1. The gas mixture inside the volume consists of water vapor (H2O_gas),
% *    CO2 and water (H2O_liq, state: fog).
% * 2. The saturation density is a function of the temperature. The used
% *    function is described in literature.
% *
% *
% * Last modification : 15.03.2008
% * Author : Christian Müller(HAW)
% *
% ************************************************************************

% * 1. Definition specific gas constants
R_air             = 287.058;
```

```
R_H2O_gas         = 461.523;
R_CO2             = 188.924;
% **********************************************************************

% * 2. Calculation of the saturation density by a given temperature
rho_H2O_gas_sat   = 4.44259*exp(15.05703*(T_init-273.15)/(208.07254+(T_init-273.15)))↙
/1000;
% **********************************************************************

% * 3. Calculation of the saturation mass
m_H2O_gas_sat     = V*rho_H2O_gas_sat;
% **********************************************************************

% * 4. Initialisation
m_air_init        = p*V/(T_init*(R_air+x_H2O_gas_init*R_H2O_gas+x_CO2_init*R_CO2));
m_H2O_gas_init    = m_air_init*x_H2O_gas_init;

if m_H2O_gas_init > m_H2O_gas_sat
   m_H2O_gas_init = m_H2O_gas_sat;
   m_air_init     = ((p*V/T_init)-(m_H2O_gas_init*R_H2O_gas))/(R_air+x_CO2_init*R_CO2);
end

m_CO2_init        = m_air_init*x_CO2_init;
m_H2O_liq_init    = m_air_init*x_H2O_liq_init;
% **********************************************************************
```

```
function [T_dot,m_dot_air,m_dot_H2O_gas,m_dot_CO2,m_dot_H2O_liq] = T_m(T,m_air,↙
x_H2O_gas,x_CO2,x_H2O_liq,T_in,m_dot_air_in,x_H2O_gas_in,x_CO2_in,x_H2O_liq_in,Q_dot,↙
m_dot_H2O_gas_human,m_dot_CO2_human)

% *************************************************************************
% * Definition of a isobaric with 2 apertures
% *
% * Number of Inputs:        5
% *
% * Parameter : Volume:      V
% *
% *
% * Relevant input variables of Vol_Isobaric_2D:
% *
% * Temperature:             T
% * Mass flow dry air:       m_dot_air
% * Content water vapor:     x_H2O_gas
% * Content CO2:             x_CO2
% * Content water:           x_H2O_liq
% * External Heat Flow:      Q_dot
% *
% *
% * Relevant output variables of Vol_Isobaric_2D:
% *
% * Pressure:                p
% * Density:                 rho
% * Temperature:             T
% * Mass dry air:            m_air
% * Content water vapor:     x_H2O_gas
% * Content CO2:             x_CO2
% * Content water:           x_H2O_liq
% *
% ************************************************************************
% * Embedded MATLAB Function T_m:
% *
% * Calculations:
% * 1. Definition specific gas constants.
% * 2. Definition of the outflowing mass flow
% * 3. Modification of the mass inside the system.
% * 4. Modification of the temperature inside the system.
% *
% *
% * Assumptions:
% * 1. The gas mixture inside the volume consists of water vapor (H2O_gas),
% *    CO2 and water (H2O_liq, state: fog).
% * 2. In the case of outgoing mass flow, the characteristic variables of
% *    the mass flow are defined by the state variables of the volume.
% * 3. The increase or decrease of the mass inside the volume is defined
% *    by the incoming (sign: plus) or outgoing (sign: minus) mass flow.
% * 4. The Enthalpy equation of the gas inside the volume is used. the
% *    enthalpy of the liquid water inside the volume is neglected.
% *    Assuming that the liquid water is always in thermal equilibrium with
% *    the gas.
% *
% *
% * Last modification : 15.03.2008
```

```
% * Author : Christian Müller(HAW)
% *
% ***********************************************************************

% * 1. Definition specific gas constants
R_air           = 287.058;
R_H2O_gas       = 461.523;
R_CO2           = 188.924;
c_p_air         = 1005;
c_p_H2O_gas     = 1870;
c_p_CO2         = 830;
% ***********************************************************************

% * 2. Definition of the outflowing mass flow
if m_dot_air_in < 0
   m_dot_air_in = 0;
end

m_dot_air_out   = (Q_dot+m_dot_air_in*↙
(c_p_air+x_H2O_gas_in*c_p_H2O_gas+x_CO2_in*c_p_CO2)↙
*T_in+m_dot_H2O_gas_human*c_p_H2O_gas*T+m_dot_CO2_human*c_p_CO2*T)/↙
((c_p_air+x_H2O_gas*c_p_H2O_gas+x_CO2*c_p_CO2)*T);
m_dot_air_out   = -m_dot_air_out;
% ***********************************************************************

% * 3. Modification of the mass inside the system
m_dot_air       = m_dot_air_in+m_dot_air_out;
m_dot_H2O_gas   =↙
m_dot_air_in*x_H2O_gas_in+m_dot_air_out*x_H2O_gas+m_dot_H2O_gas_human;
m_dot_CO2       = m_dot_air_in*x_CO2_in+m_dot_air_out*x_CO2+m_dot_CO2_human;
m_dot_H2O_liq   = m_dot_air_in*x_H2O_liq_in+m_dot_air_out*x_H2O_liq;
% ***********************************************************************

% * 4. Modification of the temperature inside the system
m               = m_air+m_air*x_H2O_gas+m_air*x_CO2;
m_dot           = m_dot_air+m_dot_H2O_gas+m_dot_CO2;
T_dot           = -(T/m)*m_dot;
% ***********************************************************************
```

```
function [rho_bar,T_bar,m_air_bar,x_H2O_gas_bar,x_CO2,x_H2O_liq_bar,Delta_T,↙
Delta_m_H2O_gas,Delta_m_H2O_liq] = p_rho(p,T,m_air,m_H2O_gas,m_CO2,m_H2O_liq,V,↙
Delta_T_old,Delta_m_H2O_gas_old,Delta_m_H2O_liq_old)

% ***************************************************************************
% * Definition of a isobaric with 2 apertures
% *
% * Number of Inputs:        5
% *
% * Parameter : Volume:      V
% *
% *
% * Relevant input variables of Vol_Isobaric_2D:
% *
% * Temperature:             T
% * Mass flow dry air:       m_dot_air
% * Content water vapor:     x_H2O_gas
% * Content CO2:             x_CO2
% * Content water:           x_H2O_liq
% * External Heat Flow:      Q_dot
% *
% *
% * Relevant output variables of Vol_Isobaric_2D:
% *
% * Pressure:                p
% * Density:                 rho
% * Temperature:             T
% * Mass dry air:            m_air
% * Content water vapor:     x_H2O_gas
% * Content CO2:             x_CO2
% * Content water:           x_H2O_liq
% *
% **************************************************************************
% * Embedded MATLAB Function p_rho:
% *
% * Calculations:
% * 1. Redefinition of the input variables.
% * 2. Definition specific gas constants.
% * 3. Calculation of the water vapor content, CO2 content and
% *    water content.
% * 4. Calculation of the density.
% * 5. Condensation/Evaporation
% *
% *
% * Assumptions:
% * 1. The gas mixture inside the volume consists of water vapor (H2O_gas),
% *    CO2 and water (H2O_liq, state: fog).
% * 3. The differnt contents are defined in respect of the mass of the
% *    dry air.
% *
% *
% * Last modification : 15.03.2008
% * Author : Christian Müller(HAW)
% *
% **************************************************************************
```

```
% * 1. Redefinition of the input variables
T                              = T+Delta_T_old;

if T < 0
   T                           = 0;
end

m_H2O_gas                      = m_H2O_gas+Delta_m_H2O_gas_old;

if m_H2O_gas < 0
   m_H2O_gas                   = 0;
end

m_H2O_liq                      = m_H2O_liq+Delta_m_H2O_liq_old;

if m_H2O_liq < 0
   m_H2O_liq                   = 0;
end
% **************************************************************************

% * 2. Definition specific gas constants
R_air                          = 287.058;
R_H2O_gas                      = 461.523;
R_CO2                          = 188.924;
c_p_air                        = 1005;
c_p_H2O_gas                    = 1870;
c_p_CO2                        = 830;
c_p_H2O_liq                    = 4173;
c_p_H2O_ice                    = 2050;
r_0                            = 2500000;
r_ice                          = 333000;
% **************************************************************************

% * 3. Calculation of the water vapor content, CO2 content and
% *    water content
x_H2O_gas                      = m_H2O_gas/m_air;
x_CO2                          = m_CO2/m_air;
x_H2O_liq                      = m_H2O_liq/m_air;
% **************************************************************************

% * 4. Calculation of the density
rho                            = (p/T)*((1+x_H2O_gas+x_CO2)/↙
(R_air+x_H2O_gas*R_H2O_gas+x_CO2*R_CO2));
m_air_bar                      = (p*V/T)*(1/(R_air+x_H2O_gas*R_H2O_gas+x_CO2*R_CO2));
% **************************************************************************

% * 5. Condensation/Evaporation
rho_bar                        = rho;
T_bar                          = T;
x_H2O_gas_bar                  = x_H2O_gas;
x_H2O_liq_bar                  = x_H2O_liq;
rho_H2O_gas_sat                = 4.44259*exp(15.05703*(T-273.15)/(208.07254+(T-273.15)))↙
/1000;
x_H2O_gas_sat                  = rho_H2O_gas_sat/(m_air_bar/V);
evap                           = 0;
cond                           = 0;
```

```
if x_H2O_gas_sat > x_H2O_gas
   evap                        = 1;
end

if x_H2O_gas_sat < x_H2O_gas
   cond                        = 1;
end

if evap > 0
   test                        = 1;
   iter                        = 0;

   while test > 0
         rho_H2O_gas_sat_evap = 4.44259*exp(15.05703*(T_bar-273.15)/(208.07254+(T_bar-↙
273.15)))/1000;
         x_H2O_gas_sat_evap   = rho_H2O_gas_sat_evap/(m_air_bar/V);

         if (x_H2O_gas_sat_evap-x_H2O_gas) < x_H2O_liq
            x_H2O_gas_bar      = x_H2O_gas_sat_evap;
         else
            x_H2O_gas_bar      = x_H2O_gas+x_H2O_liq;
         end

         h_air_T               = (c_p_air+x_H2O_gas*c_p_H2O_gas+x_CO2*c_p_CO2)*T;
         h_H2O_liq_T           = x_H2O_liq*c_p_H2O_liq*T;
         Q_lat                 = r_0*(x_H2O_gas_bar-x_H2O_gas);
         Z_T_bar               = (h_air_T+h_H2O_liq_T-Q_lat)/↙
(c_p_air+x_H2O_gas_bar*c_p_H2O_gas+x_CO2*c_p_CO2+(x_H2O_liq-(x_H2O_gas_bar-x_H2O_gas))↙
*c_p_H2O_liq);

         if T < 273.15
            h_H2O_ice_T        = x_H2O_liq*c_p_H2O_ice*T+(x_H2O_gas_bar-x_H2O_gas)↙
*r_ice;
            Z_T_bar            = (h_air_T+h_H2O_ice_T-Q_lat)/↙
(c_p_air+x_H2O_gas_bar*c_p_H2O_gas+x_CO2*c_p_CO2+(x_H2O_liq-(x_H2O_gas_bar-x_H2O_gas))↙
*c_p_H2O_ice);
         end

         Z_T_bar               = (Z_T_bar+T_bar)/2;

         if abs(Z_T_bar-T_bar) < 0.1
            test               = 0;
         end
         if iter > 10
            test               = 0;
         end

         T_bar                 = Z_T_bar;
         iter                  = iter+1;
   end

   if T_bar >= T
      T_bar                    = T;
      rho_bar                  = rho;
      x_H2O_gas_bar            = x_H2O_gas;
```

```
      x_H2O_liq_bar            = x_H2O_liq;
   else
     x_H2O_liq_bar             = x_H2O_liq-(x_H2O_gas_bar-x_H2O_gas);
     R_avg                     = (R_air+x_H2O_gas_bar*R_H2O_gas+x_CO2*R_CO2)/↙
(1+x_H2O_gas_bar+x_CO2);
     rho_bar                   = p/(R_avg*T_bar);
   end
end

if cond>0
   test                        = 1;
   iter                        = 0;
   x_H2O_gas_sat_cond          = 0;

   while test > 0
         rho_H2O_gas_sat_cond = 4.44259*exp(15.05703*(T_bar-273.15)/(208.07254+(T_bar-↙
273.15)))/1000;
         x_H2O_gas_sat_cond    = rho_H2O_gas_sat_cond/(m_air_bar/V);
         h_air_T               = (c_p_air+x_H2O_gas*c_p_H2O_gas+x_CO2*c_p_CO2)*T;
         h_H2O_liq_T           = x_H2O_liq*c_p_H2O_liq*T;
         Q_lat                 = r_0*(x_H2O_gas-x_H2O_gas_sat_cond);
         Z_T_bar               = (h_air_T+h_H2O_liq_T+Q_lat)/↙
(c_p_air+x_H2O_gas_sat_cond*c_p_H2O_gas+x_CO2*c_p_CO2+(x_H2O_liq+(x_H2O_gas-↙
x_H2O_gas_sat_cond))*c_p_H2O_liq);

         if T < 273.15
            h_H2O_ice_T        = x_H2O_liq*c_p_H2O_ice*T-(x_H2O_gas-x_H2O_gas_sat_cond)↙
*r_ice;
            Z_T_bar            = (h_air_T+h_H2O_ice_T+Q_lat)/↙
(c_p_air+x_H2O_gas_sat_cond*c_p_H2O_gas+x_CO2*c_p_CO2+(x_H2O_liq+(x_H2O_gas-↙
x_H2O_gas_sat_cond))*c_p_H2O_ice);
         end

         Z_T_bar               = (Z_T_bar+T_bar)/2;

         if abs(Z_T_bar-T_bar) < 0.1
            test               = 0;
         end

         if iter > 10
            test               = 0;
         end

         T_bar                 = Z_T_bar;
         iter                  = iter+1;
   end

   if T_bar <= T
      T_bar                    = T;
      rho_bar                  = rho;
      x_H2O_gas_bar            = x_H2O_gas;
      x_H2O_liq_bar            = x_H2O_liq;
   else
      x_H2O_gas_bar            = x_H2O_gas_sat_cond;
      x_H2O_liq_bar            = x_H2O_liq+(x_H2O_gas-x_H2O_gas_sat_cond);
      R_avg                    = (R_air+x_H2O_gas_bar*R_H2O_gas+x_CO2*R_CO2)/↙
```

```
(1+x_H2O_gas_bar+x_CO2);
      rho_bar                 = p/(R_avg*T_bar);
   end
end

Delta_T                       = Delta_T_old+T_bar-T;
Delta_m_H2O_gas               = Delta_m_H2O_gas_old+m_air_bar*(x_H2O_gas_bar-↙
x_H2O_gas);
Delta_m_H2O_liq               = Delta_m_H2O_liq_old+m_air_bar*(x_H2O_liq_bar-↙
x_H2O_liq);
% ************************************************************************
```