Simulink Report: Mixing_Unit_2D_

Christian Müller

2008-03-06

Model - Mixing_Unit_2D_

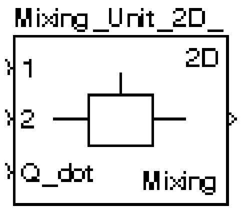


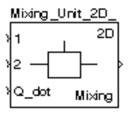
Tabelle 1.1. Mixing_Unit_2D_ Simulation Parameters

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. Mixing_Unit_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	*NonsampledZCs*	N/A

Systems

Name	**Parent**	**Snapshot**	**Blocks**	**Signals**
Mixing_Unit_2D_	<root>		Mixing_Unit_2D_	Mixing_Unit_2D_<1>

Blocks

Tabelle 1.3. Block Type Count

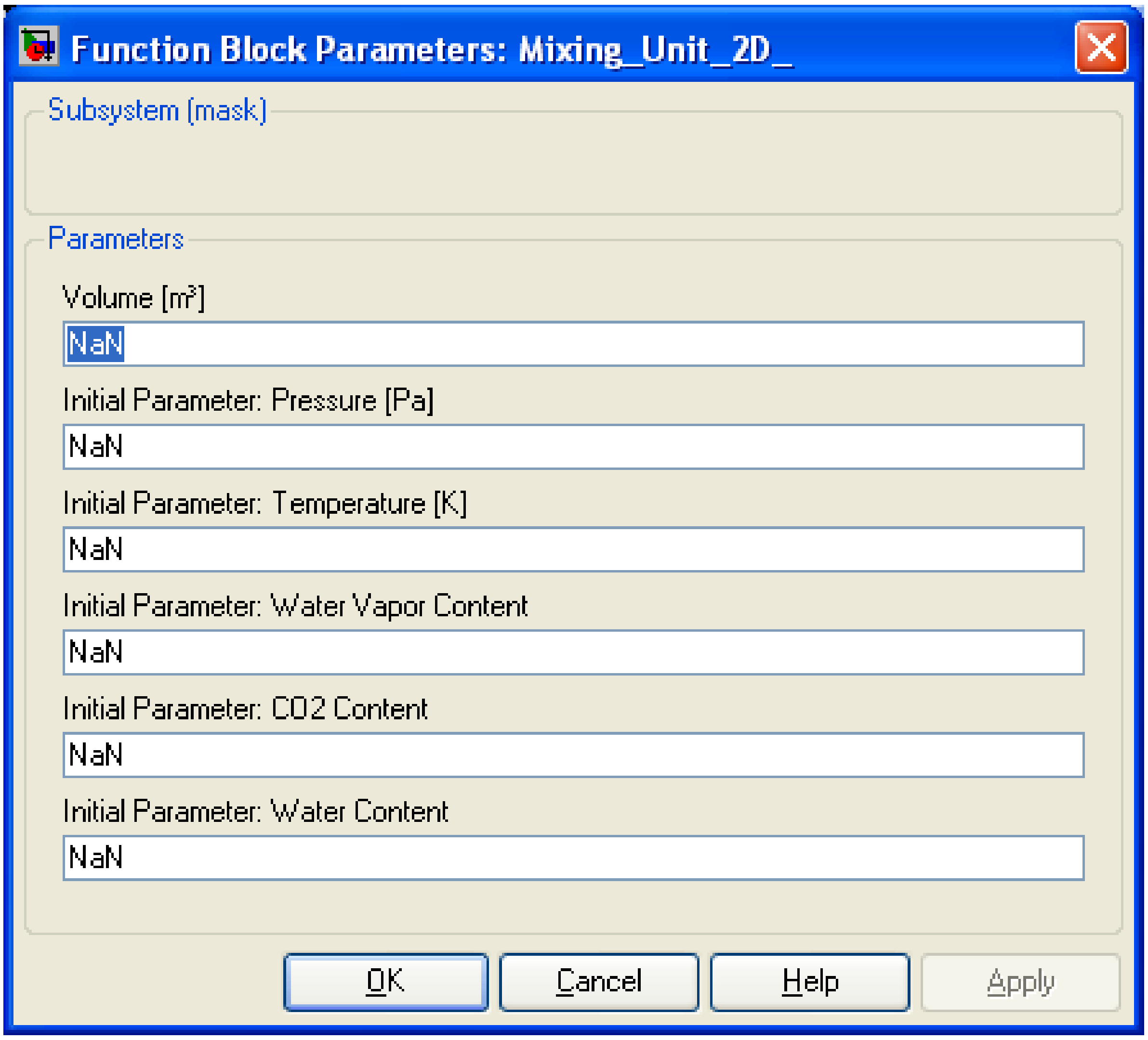
BlockType	**Count**	**Block Names**
Inport	34	In_1, In_2, Q_dot, V, p_init, T_init, x_H2O_gas_init, x_CO2_init, x_H2O_liq_init, T, m_air, m_H2O_gas, m_CO2, m_H2O_liq, T_in_1, m_dot_air_in_1, x_H2O_gas_in_1, x_CO2_in_1, x_H2O_liq_in_1, T_in_2, m_dot_air_in_2, x_H2O_gas_in_2, x_CO2_in_2, x_H2O_liq_in_2, Q_dot, T, m_air, m_H2O_gas, m_CO2, m_H2O_liq, V, Delta_T_old, Delta_m_H2O_gas_old,

BlockType	Count	Block Names
		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, p, rho_bar, T_bar, x_H2O_gas_bar, x_CO2, x_H2O_liq_bar, Delta_T, Delta_m_H2O_gas, Delta_m_H2O_liq, Out
Constant	7	T_init, V, m_dot_air, p_init, x_CO2_init, x_H2O_gas_init, x_H2O_liq_init
Integrator	5	Integrator1, Integrator2, Integrator3, Integrator4, Integrator5
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
Product	2	Product, Product1
BusSelector	2	Bus Selector6, Bus Selector8
SubSystem	1	Mixing_Unit_2D_
BusCreator	1	Bus Creator2

Data and Functions

Tabelle 1.4. Model Functions

Function Name	Parent Blocks	Calling string
NaN	Mixing_Unit_2D_ Mixing_Unit_2D_ Mixing_Unit_2D_ Mixing_Unit_2D_ Mixing_Unit_2D_ Mixing_Unit_2D_	`NaN` `NaN` `NaN` `NaN` `NaN` `NaN`



Subsystem (mask)

Parameters

Volume [m³]

NaN

Initial Parameter: Pressure [Pa]

NaN

Initial Parameter: Temperature [K]

NaN

Initial Parameter: Water Vapor Content

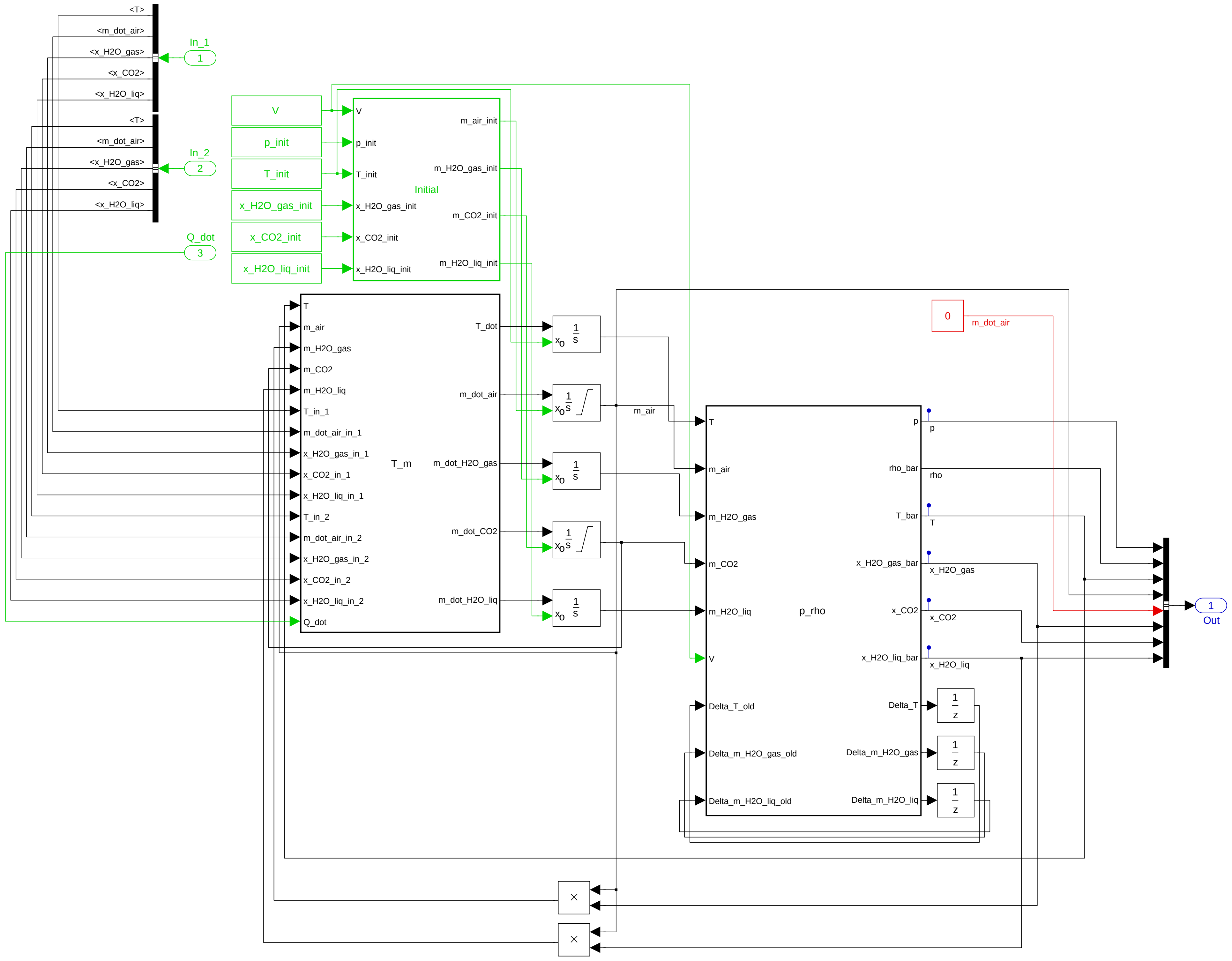
NaN

Initial Parameter: CO2 Content

NaN

Initial Parameter: Water Content

NaN



<T>
<m_dot_air>
<x_H2O_gas>
<x_CO2>
<x_H2O_liq>
In_1
1
In_2
2
Q_dot
3
V
p_init
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
T
m_air
m_H2O_gas
m_CO2
m_H2O_liq
T_in_1
m_dot_air_in_1
x_H2O_gas_in_1
x_CO2_in_1
x_H2O_liq_in_1
T_in_2
m_dot_air_in_2
x_H2O_gas_in_2
x_CO2_in_2
x_H2O_liq_in_2
Q_dot
T_m
T_dot
m_dot_air
m_dot_H2O_gas
m_dot_CO2
m_dot_H2O_liq
1/s
0
m_dot_air
p_rho
p
rho_bar
rho
T_bar
x_H2O_gas_bar
x_H2O_gas
x_CO2
x_H2O_liq_bar
x_H2O_liq
Delta_T_old
Delta_m_H2O_gas_old
Delta_m_H2O_liq_old
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_init,↙
T_init,x_H2O_gas_init,x_CO2_init,x_H2O_liq_init)

% *************************************************************************
% * Definition of a mixing unit with 2 apertures
% *
% * Number of Inputs:        3
% *
% * Parameter : Volume:      V
% *
% *
% * Relevant input variables of Mixing_Unit_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 Mixing_Unit_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_init*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_init*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,↙
m_H2O_gas,m_CO2,m_H2O_liq,T_in_1,m_dot_air_in_1,x_H2O_gas_in_1,x_CO2_in_1,↙
x_H2O_liq_in_1,T_in_2,m_dot_air_in_2,x_H2O_gas_in_2,x_CO2_in_2,x_H2O_liq_in_2,Q_dot)

% ************************************************************************
% * Definition of a mixing unit with 2 apertures
% *
% * Number of Inputs:        3
% *
% * Parameter : Volume:      V
% *
% *
% * Relevant input variables of Mixing_Unit_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 Mixing_Unit_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. Redefinition of the input variables.
% * 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;
c_v_air          = c_p_air-R_air;
c_v_H2O_gas      = c_p_H2O_gas-R_H2O_gas;
c_v_CO2          = c_p_CO2-R_CO2;
% ************************************************************************

% * 2. Redefinition of the input variables
if m_dot_air_in_1 < 0
   T_in_1         = T;
   m_dot          = m_dot_air_in_1*(1+x_H2O_gas_in_1+x_CO2_in_1);
   m_dot_air_in_1 = m_dot*(1/(1+(m_H2O_gas/m_air)+(m_CO2/m_air)));
   x_H2O_gas_in_1 = (m_H2O_gas/m_air);
   x_CO2_in_1     = (m_CO2/m_air);
   x_H2O_liq_in_1 = (m_H2O_liq/m_air);
end

if m_dot_air_in_2 < 0
   T_in_2         = T;
   m_dot          = m_dot_air_in_2*(1+x_H2O_gas_in_2+x_CO2_in_2);
   m_dot_air_in_2 = m_dot*(1/(1+(m_H2O_gas/m_air)+(m_CO2/m_air)));
   x_H2O_gas_in_2 = (m_H2O_gas/m_air);
   x_CO2_in_2     = (m_CO2/m_air);
   x_H2O_liq_in_2 = (m_H2O_liq/m_air);
end
% ************************************************************************

% * 3. Modification of the mass inside the system
m_dot_air        = m_dot_air_in_1+m_dot_air_in_2;
m_dot_H2O_gas    = m_dot_air_in_1*x_H2O_gas_in_1+m_dot_air_in_2*x_H2O_gas_in_2;
m_dot_CO2        = m_dot_air_in_1*x_CO2_in_1+m_dot_air_in_2*x_CO2_in_2;
m_dot_H2O_liq    = m_dot_air_in_1*x_H2O_liq_in_1+m_dot_air_in_2*x_H2O_liq_in_2;
% ************************************************************************

% * 4. Modification of the temperature inside the system
T_dot            = (Q_dot+m_dot_air_in_1*(c_p_air*T_in_1-c_v_air*T)+m_dot_air_in_2*↙
(c_p_air*T_in_2-c_v_air*T)+m_dot_air_in_1*x_H2O_gas_in_1*(c_p_H2O_gas*T_in_1-↙
c_v_H2O_gas*T)+m_dot_air_in_2*x_H2O_gas_in_2*(c_p_H2O_gas*T_in_2-c_v_H2O_gas*T)↙
+m_dot_air_in_1*x_CO2_in_1*(c_p_CO2*T_in_1-c_v_CO2*T)+m_dot_air_in_2*x_CO2_in_2*↙
(c_p_CO2*T_in_2-c_v_CO2*T))/(c_v_air*m_air+c_v_H2O_gas*m_H2O_gas+c_v_CO2*m_CO2);
% ************************************************************************
```

```
function [p,rho_bar,T_bar,x_H2O_gas_bar,x_CO2,x_H2O_liq_bar,Delta_T,Delta_m_H2O_gas,↙
Delta_m_H2O_liq] = p_rho(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 mixing unit with 2 aperture
% *
% * Number of Inputs:        3
% *
% * Parameter : Volume:      V
% *
% *
% * Relevant input variables of Mixing_Unit_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 Mixing_Unit_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 density.
% * 4. Calculation of the pressure.
% * 5. Calculation of the water vapor content, CO2 content and
% *    water content.
% * 6. Condensation/Evaporation
% *
% *
% * Assumptions:
% * 1. The gas mixture inside the volume consists of water vapor (H2O_gas),
% *    CO2 and water (H2O_liq, state: fog).
% * 5. 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 density
rho                            = (m_air+m_H2O_gas+m_CO2)/V;
% ************************************************************************

% * 4. Calculation of the pressure
p                              = T*(m_air*R_air+m_H2O_gas*R_H2O_gas+m_CO2*R_CO2)/V;
% ************************************************************************


% * 5. 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;
% ************************************************************************

% * 6. 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/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/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/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*(x_H2O_gas_bar-x_H2O_gas);
Delta_m_H2O_liq               = Delta_m_H2O_liq_old+m_air*(x_H2O_liq_bar-x_H2O_liq);
% *********************************************************************
```