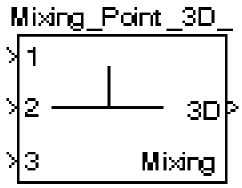
# Simulink Report: Mixing_Point_3D_

Christian Müller

2008-03-06

# Model - Mixing_Point_3D_



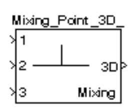

### Tabelle 1.1. Mixing_Point_3D_ Simulation Parameters

| | | |
|---|---|---|
| *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_Point_3D_ 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** |
|---|---|---|---|---|
| Mixing_Point _3D_ | <root> |   | Mixing_Point_3D_ | Mixing_Point_3D_<1> |

# Blocks

### Tabelle 1.3. Block Type Count

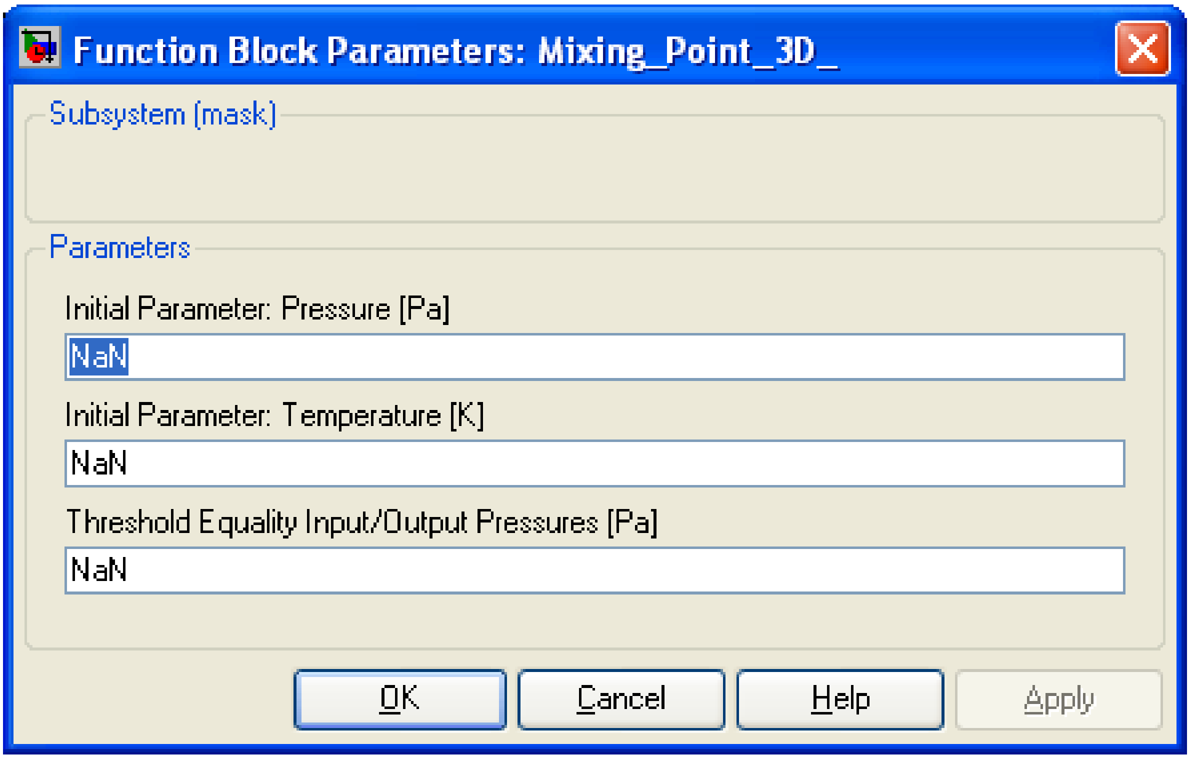
| **BlockType** | **Count** | **Block Names** |
|---|---|---|
| Inport | 31 | In_1, In_2, In_3, p_old, rho_old, T_old, x_H2O_gas_old, x_CO2_old, x_H2O_liq_old, p_in_1, rho_in_1, T_in_1, m_dot_air_in_1, x_H2O_gas_in_1, x_CO2_in_1, x_H2O_liq_in_1, p_in_2, rho_in_2, T_in_2, m_dot_air_in_2, x_H2O_gas_in_2, x_CO2_in_2, x_H2O_liq_in_2, p_in_3, rho_in_3, T_in_3, m_dot_air_in_3, x_H2O_gas_in_3, x_CO2_in_3, x_H2O_liq_in_3, Delta_p_threshold |

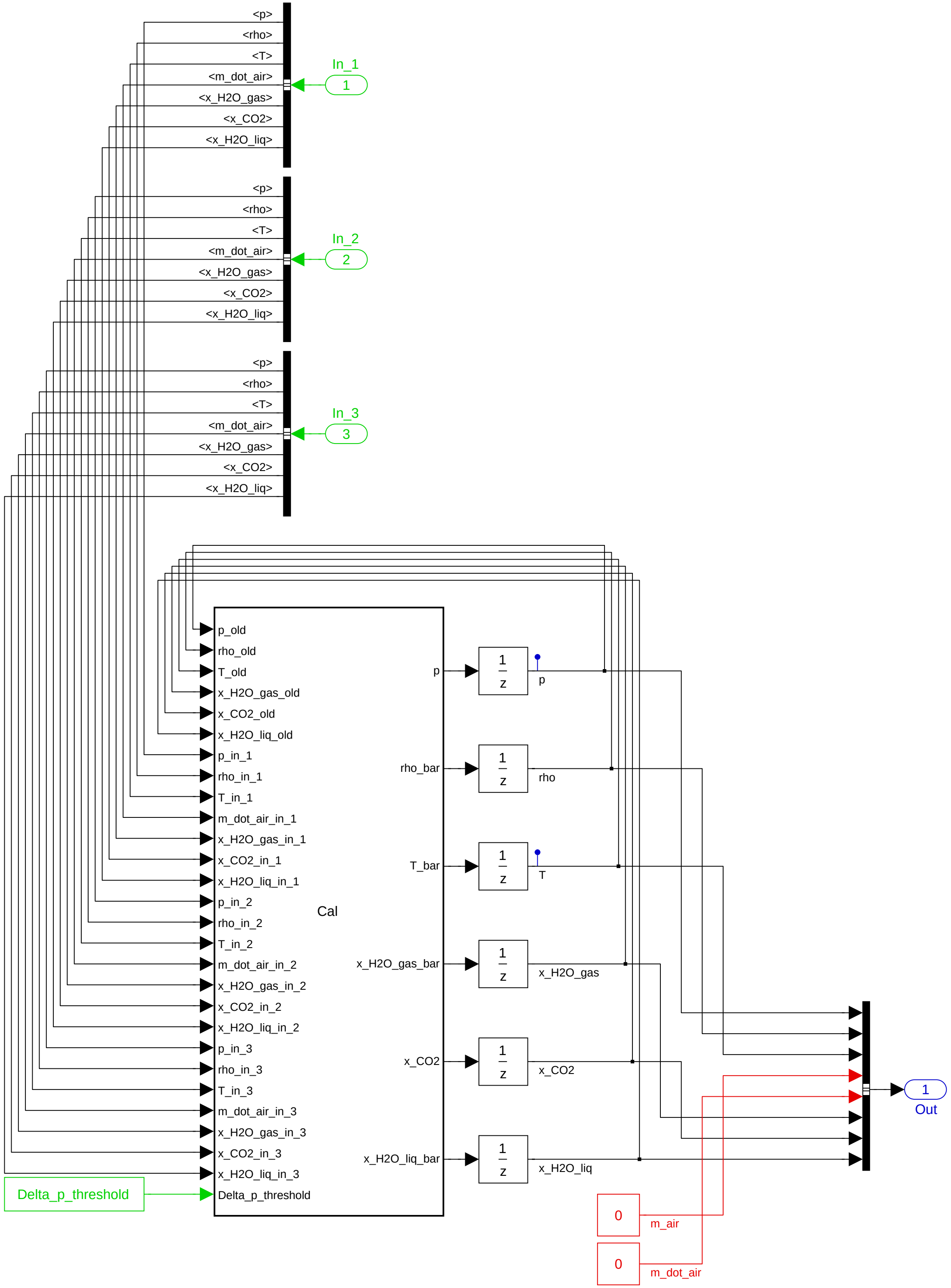
| BlockType | Count | Block Names |
|---|---|---|
| Outport | 7 | p, rho_bar, T_bar, x_H2O_gas_bar, x_CO2, x_H2O_liq_bar, Out |
| UnitDelay | 6 | Unit Delay, Unit Delay1, Unit Delay2, Unit Delay3, Unit Delay4, Unit Delay5 |
| Constant | 3 | Delta_p_threshold, m_air, m_dot_air |
| BusSelector | 3 | Bus Selector1, Bus Selector2, Bus Selector4 |
| Terminator | 1 | Terminator |
| SubSystem | 1 | Mixing_Point_3D_ |
| Stateflow (m) | 1 | Embedded_MATLAB_Function |
| S-Function | 1 | SFunction |
| Demux | 1 | Demux |
| BusCreator | 1 | Bus Creator1 |

# Data and Functions

### Tabelle 1.4. Model Functions

| Function Name | Parent Blocks | Calling string |
|---|---|---|
| NaN | Mixing_Point_3D_<br>Mixing_Point_3D_<br>Mixing_Point_3D_ | `NaN`<br>`NaN`<br>`NaN` |


Function Block Parameters: Mixing_Point_3D_
Subsystem (mask)
Parameters
Initial Parameter: Pressure [Pa]
NaN
Initial Parameter: Temperature [K]
NaN
Threshold Equality Input/Output Pressures [Pa]
NaN
OK
Cancel
Help
Apply



<p>
<rho>
<T>
<m_dot_air>
<x_H2O_gas>
<x_CO2>
<x_H2O_liq>
In_1
1
<p>
<rho>
<T>
<m_dot_air>
<x_H2O_gas>
<x_CO2>
<x_H2O_liq>
In_2
2
<p>
<rho>
<T>
<m_dot_air>
<x_H2O_gas>
<x_CO2>
<x_H2O_liq>
In_3
3
p_old
rho_old
T_old
x_H2O_gas_old
x_CO2_old
x_H2O_liq_old
p_in_1
rho_in_1
T_in_1
m_dot_air_in_1
x_H2O_gas_in_1
x_CO2_in_1
x_H2O_liq_in_1
p_in_2
rho_in_2
T_in_2
m_dot_air_in_2
x_H2O_gas_in_2
x_CO2_in_2
x_H2O_liq_in_2
p_in_3
rho_in_3
T_in_3
m_dot_air_in_3
x_H2O_gas_in_3
x_CO2_in_3
x_H2O_liq_in_3
Delta_p_threshold
Delta_p_threshold
Cal
p
rho_bar
T_bar
x_H2O_gas_bar
x_CO2
x_H2O_liq_bar
1/z
p
rho
T
x_H2O_gas
x_CO2
x_H2O_liq
0
m_air
0
m_dot_air
1
Out

```
function [p,rho_bar,T_bar,x_H2O_gas_bar,x_CO2,x_H2O_liq_bar] = Cal(p_old,rho_old,T_old,↙
x_H2O_gas_old,x_CO2_old,x_H2O_liq_old,p_in_1,rho_in_1,T_in_1,m_dot_air_in_1,↙
x_H2O_gas_in_1,x_CO2_in_1,x_H2O_liq_in_1,p_in_2,rho_in_2,T_in_2,m_dot_air_in_2,↙
x_H2O_gas_in_2,x_CO2_in_2,x_H2O_liq_in_2,p_in_3,rho_in_3,T_in_3,m_dot_air_in_3,↙
x_H2O_gas_in_3,x_CO2_in_3,x_H2O_liq_in_3,Delta_p_threshold)

% **************************************************************************
% * Definition of a mixing point with 3 apertures
% *
% * Number of Inputs:        3
% *
% * Parameter:               Threshold Equality Input/Output Pressures
% *
% *
% * Relevant input variables of Mixing_Point_3D
% *
% * Pressure:                p
% * Density:                 rho
% * Temperature:             T
% * Mass flow dry air:       m_dot_air
% * Content water vapor:     x_H2O_gas
% * Content CO2:             x_CO2
% * Content water:           x_H2O_liq
% *
% *
% * Relevant output variables of Mixing_Point_3D
% *
% * Pressure:                p
% * Density:                 rho
% * Temperature:             T
% * Content water vapor:     x_H2O_gas
% * Content CO2:             x_CO2
% * Content water:           x_H2O_liq
% *
% *************************************************************************
% * Embedded MATLAB Function Cal:
% *
% * Calculations:
% * 1. Definition specific gas constants.
% * 2. Definition state variables.
% * 3. Redefinition of the input variables.
% * 4. Consistency check.
% * 5. Operation mode: Determination of the pressure
% *                    5.1. Calculation pressure
% *                    5.2. Calculation temperature, water vapor content,
% *                         CO2-content, 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).
% * 4. Consistency Check:
% *    check = 1: Operation mode
% *    check = 0: Standby mode
% *    Conditions for the standby mode:
```

```
% *    a) All mass flows are incoming mass flows (sign: plus)
% *    b) All mass flows are outgoing mass flows (sign: minus)
% *    c) All mass flows are equal zero
% * 5. Operation mode:
% *    5.1. Calculation pressure
% *
% *         Linearization: For each time step the following equations will
% *                        be resolved for the constants K_1, K_2 and K_3.
% *                        K_1, K_2 and K_3 are time dependant.
% *
% *                        m_dot_1 = K_1*(p_in_1-p)
% *                        m_dot_2 = K_2*(p_in_2-p)
% *                        m_dot_3 = K_3*(p_in_3-p)
% *
% *         Calculation pressure: Assuming that the constants are fixed within
% *                               one time step, a estimation for the pressure
% *                               can be made. the sum of the incoming and the
% *                               outgoing mass flows has to be zero.
% *                               (m_dot_1+m_dot_2+m_dot_3 = 0)
% *
% *                               m_dot_1+m_dot_2+m_dot_3 = 0
% *                               => K_1*(p_in_1-p)+K_2*(p_in_2-p)+K_3*(p_in_3-p) = 0
% *                               => p = (K_1*p_in_1+K_2*p_in_2+K_3*p_in_3)/↙
(K_1+K_2+K_3)
% *
% *                               In the case that one mass flow is constant.
% *                               e.g. m_dot_1 = const. The pressure has to be
% *                               calculated by the following equation.
% *
% *                               p = (m_dot_1+K_2*p_in_2+K_3*p_in_3)/(K_2+K_3)
% *
% *                               For stability reasons p_in_1, p_in_2 or p_in_3 and
% *                               p will be assumed as equal, then e.g.
% *                               (p_in_1-p) < Delta_p_threshold
% *
% *    5.2. Calculation temperature, water vapor content, CO2-content, water content
% *
% *         The state variables of incoming mass flows contibute to the
% *         calculation of the temperature, the water vapor content,
% *         the CO2-content and the water content of the node. The values
% *         will be calculated by a weighted average. The state variables
% *         of the outgoing mass flows will be determined by the node.
% *
% *
% * Last modification : 18.01.2008
% * 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_p_H2O_liq                  = 4173;
c_p_H2O_ice                  = 2050;
r_0                          = 2500000;
r_ice                        = 333000;
% ************************************************************************

% * 2. Definition state variables
p                            = p_old;
rho                          = rho_old;
T                            = T_old;
x_H2O_gas                    = x_H2O_gas_old;
x_CO2                        = x_CO2_old;
x_H2O_liq                    = x_H2O_liq_old;
% ************************************************************************

% * 3. Calculation incoming and outgoing mass flows
m_dot_1                      =↙
m_dot_air_in_1+m_dot_air_in_1*x_H2O_gas_in_1+m_dot_air_in_1*x_CO2_in_1;
m_dot_2                      =↙
m_dot_air_in_2+m_dot_air_in_2*x_H2O_gas_in_2+m_dot_air_in_2*x_CO2_in_2;
m_dot_3                      =↙
m_dot_air_in_3+m_dot_air_in_3*x_H2O_gas_in_3+m_dot_air_in_3*x_CO2_in_3;
% ************************************************************************

% * 4. Consistency check
check                        = 1;

if m_dot_1 > 0
   if m_dot_2 > 0
      if m_dot_3 > 0
         check               = 0;
      end
   end
end

if m_dot_1 < 0
   if m_dot_2 < 0
      if m_dot_3 < 0
         check               = 0;
      end
   end
end

if abs(m_dot_1)+abs(m_dot_2)+abs(m_dot_3) == 0
   check                     = 0;
end
% ************************************************************************

% * 5. Operation mode:
% * 5.1. Calculation pressure
if check == 1
   Numerator                 = 0;
   Denominator               = 0;

   if p_in_1 == 0
      Numerator              = Numerator+m_dot_1;
```

```
   else
      if abs(p_in_1-p) < Delta_p_threshold
         K_1                    = 0;
      else
         K_1                    = abs(m_dot_1)/abs(p_in_1-p);
      end

      Numerator                 = Numerator+K_1*p_in_1;
      Denominator               = Denominator+K_1;
   end

   if p_in_2 == 0
      Numerator                 = Numerator+m_dot_2;
   else
      if abs(p_in_2-p) < Delta_p_threshold
         K_2                    = 0;
      else
         K_2                    = abs(m_dot_2)/abs(p_in_2-p);
      end

      Numerator                 = Numerator+K_2*p_in_2;
      Denominator               = Denominator+K_2;
   end

   if p_in_3 == 0
      Numerator                 = Numerator+m_dot_3;
   else
      if abs(p_in_3-p) < Delta_p_threshold
         K_3                    = 0;
      else
         K_3                    = abs(m_dot_3)/abs(p_in_3-p);
      end

      Numerator                 = Numerator+K_3*p_in_3;
      Denominator               = Denominator+K_3;
   end

   if Denominator > 0
      p                         = Numerator/Denominator;
   end
end
% ************************************************************************

% * 5.2. Calculation temperature, water vapor content, CO2-content,
% *      water content
if check == 1
   Denominator                  = 0;
   T                            = 0;
   x_H2O_gas                    = 0;
   x_CO2                        = 0;
   x_H2O_liq                    = 0;

   if m_dot_1 > 0
      T                         = T+m_dot_1*T_in_1;
      x_H2O_gas                 = x_H2O_gas+m_dot_1*x_H2O_gas_in_1;
      x_CO2                     = x_CO2+m_dot_1*x_CO2_in_1;
```

```
      x_H2O_liq                = x_H2O_liq+m_dot_1*x_H2O_liq_in_1;
      Denominator              = Denominator+m_dot_1;
   end

   if m_dot_2 > 0
      T                        = T+m_dot_2*T_in_2;
      x_H2O_gas                = x_H2O_gas+m_dot_2*x_H2O_gas_in_2;
      x_CO2                    = x_CO2+m_dot_2*x_CO2_in_2;
      x_H2O_liq                = x_H2O_liq+m_dot_2*x_H2O_liq_in_2;
      Denominator              = Denominator+m_dot_2;
   end

   if m_dot_3 > 0
      T                        = T+m_dot_3*T_in_3;
      x_H2O_gas                = x_H2O_gas+m_dot_3*x_H2O_gas_in_3;
      x_CO2                    = x_CO2+m_dot_3*x_CO2_in_3;
      x_H2O_liq                = x_H2O_liq+m_dot_3*x_H2O_liq_in_3;
      Denominator              = Denominator+m_dot_3;
   end

   if abs(Denominator) > 0
      T                        = T/Denominator;
      x_H2O_gas                = x_H2O_gas/Denominator;
      x_CO2                    = x_CO2/Denominator;
      x_H2O_liq                = x_H2O_liq/Denominator;
      R_avg                    = (R_air+x_H2O_gas*R_H2O_gas+x_CO2*R_CO2)/↙
(1+x_H2O_gas+x_CO2);
      rho                      = p/(R_avg*T);
   else
      T                        = T_old;
      rho                      = rho_old;
      x_H2O_gas                = x_H2O_gas_old;
      x_CO2                    = x_CO2_old;
      x_H2O_liq                = x_H2O_liq_old;
   end
end
% *************************************************************************
% *************************************************************************

% * 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;
rho_air                        = rho/(1+x_H2O_gas+x_CO2);
x_H2O_gas_sat                  = rho_H2O_gas_sat/rho_air;
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/rho_air;

         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/rho_air;
         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
% *************************************************************************
```