Simulink Report: Node_D_2

Christian Müller

2008-03-06

Model - Node_2D_

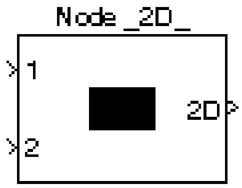


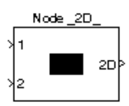
Tabelle 1.1. Node_2D_ Simulation Parameters

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. Node_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**
Node_2D_	<root>		Node_2D_	Node_2D_<1>

Blocks

Tabelle 1.3. Block Type Count

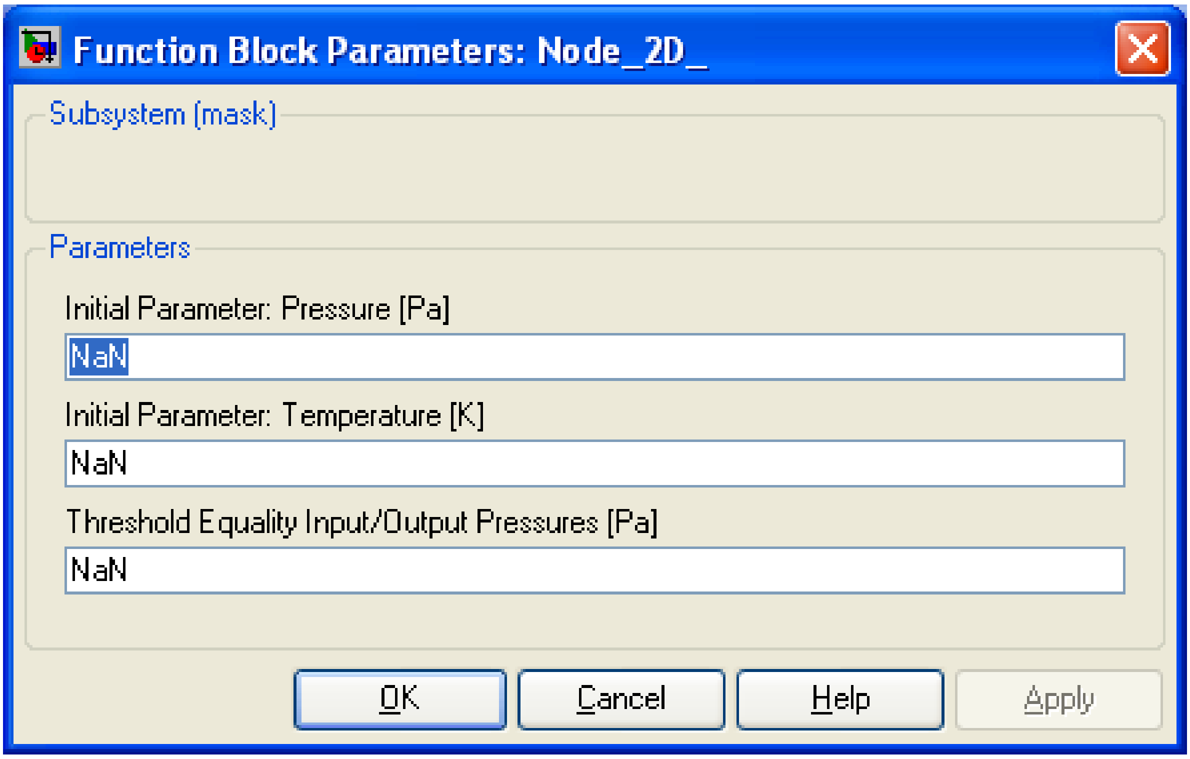
BlockType	**Count**	**Block Names**
Inport	23	In_1, In_2, p_old, rho_old, T_old, x_H2O_gas_old, x_CO2_old, x_H2O_liq_old, p_in_1, rho_in_1, T_in_1, m_dot_air_in_1, x_H2O_gas_in_1, x_CO2_in_1, x_H2O_liq_in_1, p_in_2, rho_in_2, T_in_2, m_dot_air_in_2, x_H2O_gas_in_2, x_CO2_in_2, x_H2O_liq_in_2, Delta_p_threshold
Outport	7	p, rho, T, x_H2O_gas, x_CO2, x_H2O_liq, Out

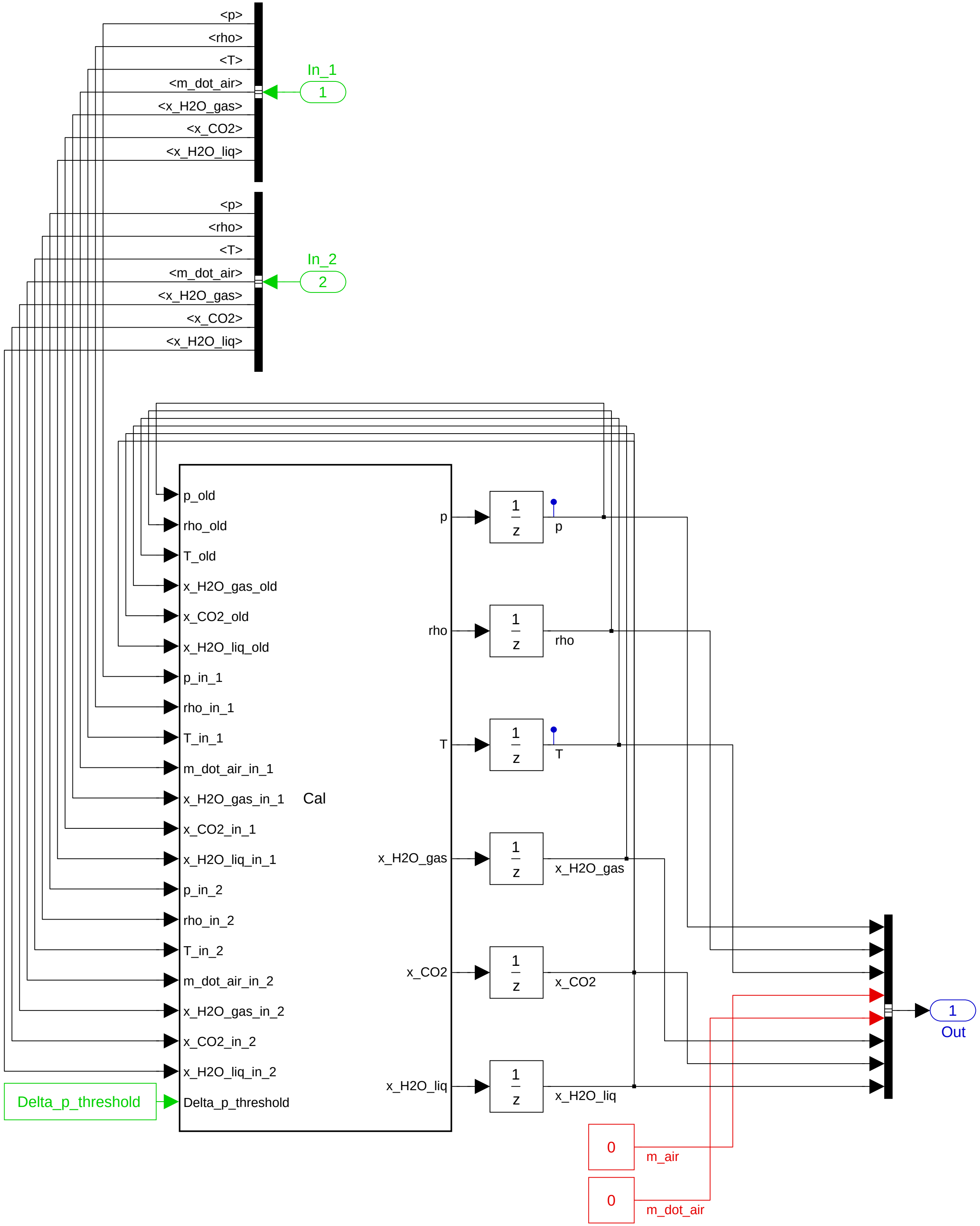
BlockType	Count	Block Names
UnitDelay	6	Unit Delay, Unit Delay1, Unit Delay2, Unit Delay3, Unit Delay4, Unit Delay5
Constant	3	Delta_p_threshold, m_air, m_dot_air
BusSelector	2	Bus Selector1, Bus Selector4
Terminator	1	Terminator
SubSystem	1	Node_2D_
Stateflow (m)	1	Embedded_MATLAB_Function
S-Function	1	SFunction
Demux	1	Demux
BusCreator	1	Bus Creator1

Data and Functions

Tabelle 1.4. Model Functions

Function Name	Parent Blocks	Calling string
NaN	Node_2D_ Node_2D_ Node_2D_	`NaN` `NaN` `NaN`


Function Block Parameters: Node_2D_
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_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
Delta_p_threshold
Delta_p_threshold
Cal
p
rho
T
x_H2O_gas
x_CO2
x_H2O_liq
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,T,x_H2O_gas,x_CO2,x_H2O_liq] = 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,Delta_p_threshold)

% ***********************************************************************
% * Definition of a node with 2 apertures
% *
% * Number of Inputs:        2
% *
% * Parameter:               Threshold Equality Input/Output Pressures
% *
% *
% * Relevant input variables of Node_2D
% *
% * 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 Node_2D
% *
% * 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
% *
% *
% * 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 and K_2.
% *                        K_1 and K_2 are time dependant.
% *
% *                        m_dot_1 = K_1*(p_in_1-p)
% *                        m_dot_2 = K_2*(p_in_2-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 = 0)
% *
% *                               m_dot_1+m_dot_2 = 0
% *                               => K_1*(p_in_1-p)+K_2*(p_in_2-p) = 0
% *                               => p = (K_1*p_in_1+K_2*p_in_2)/(K_1+K_2)
% *
% *                               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_2
% *
% *                               For stability reasons p_in_1 or p_in_2 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 : 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. 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;
% ************************************************************************

% * 4. Consistency check
check            = 1;

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

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

if abs(m_dot_1)+abs(m_dot_2) == 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 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 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
% **************************************************************************
% **************************************************************************
```