Simulink Report: FR_

Christian Müller

2008-03-06

Model - FR_

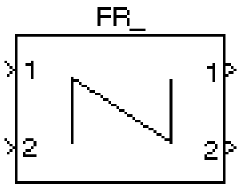


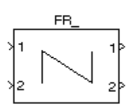
Tabelle 1.1. FR_ Simulation Parameters

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. FR_ 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**
FR_	<root>		FR_	FR_<1> FR_<2>

Blocks

Tabelle 1.3. Block Type Count

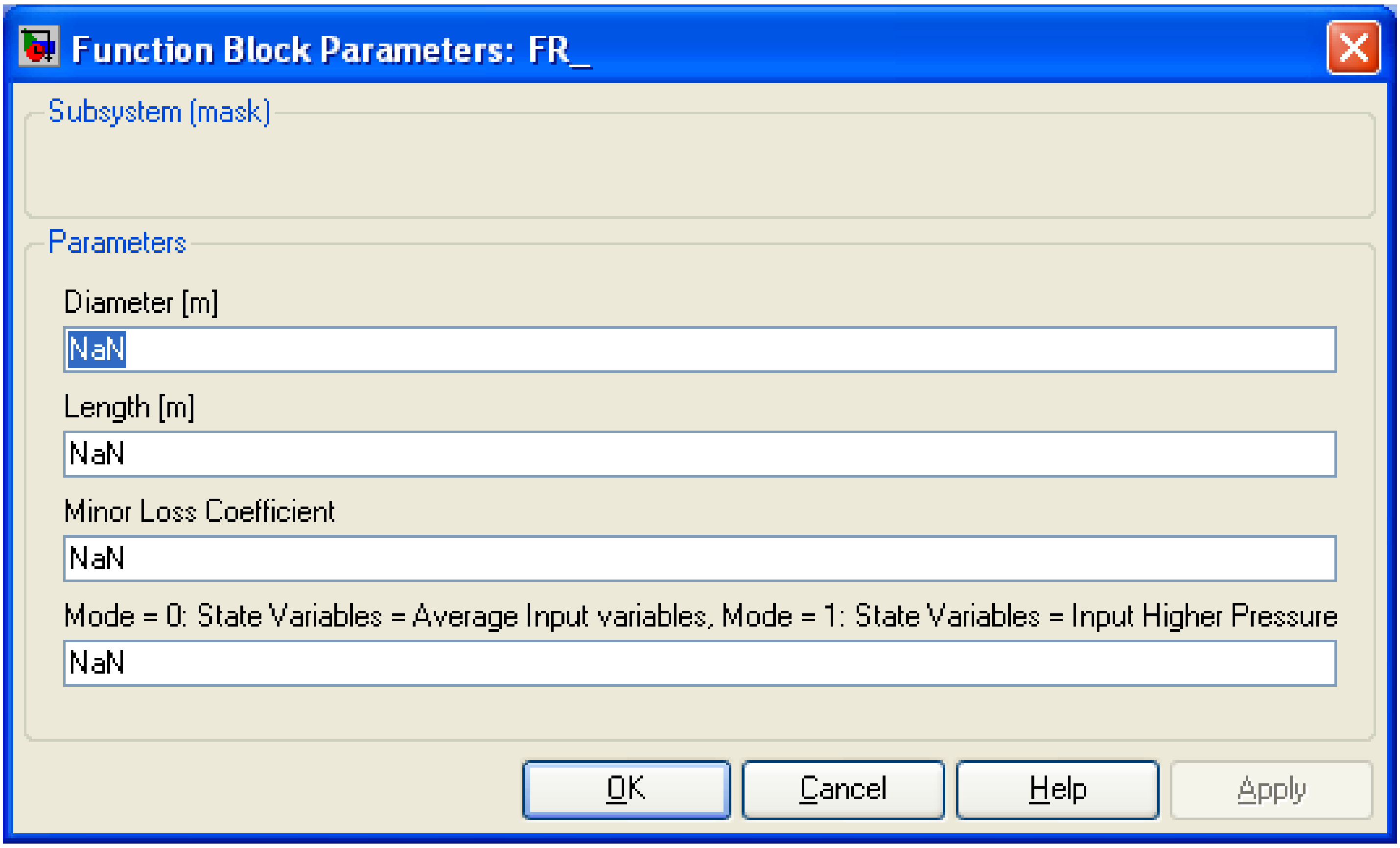
BlockType	**Count**	**Block Names**
Inport	18	In_1, In_2, D, L, zeta, mode, p_in_1, rho_in_1, T_in_1, x_H2O_gas_in_1, x_CO2_in_1, x_H2O_liq_in_1, p_in_2, rho_in_2, T_in_2, x_H2O_gas_in_2, x_CO2_in_2, x_H2O_liq_in_2
Outport	11	rho, T, x_H2O_gas, x_CO2, x_H2O_liq, p_1, m_dot_air_1, p_2, m_dot_air_2, Out_1, Out_2
Constant	5	D, L, m_air, mode, zeta

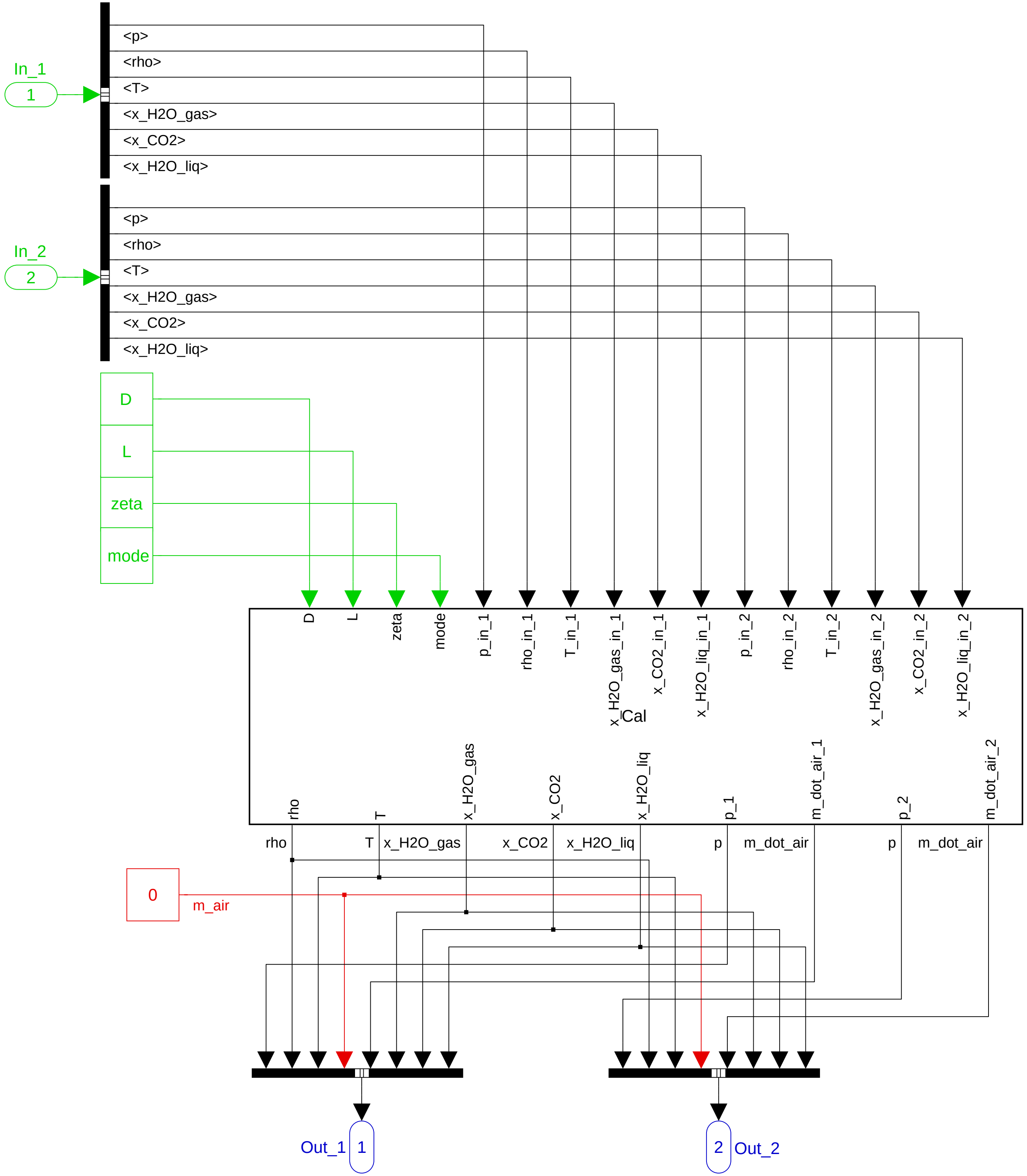
BlockType	Count	Block Names
BusSelector	2	Bus Selector3, Bus Selector4
BusCreator	2	Bus Creator1, Bus Creator2
Terminator	1	Terminator
SubSystem	1	FR_
Stateflow (m)	1	Embedded MATLA Function
S-Function	1	SFunction
Demux	1	Demux

Data and Functions

Tabelle 1.4. Model Functions

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


Function Block Parameters: FR_
Subsystem (mask)
Parameters
Diameter [m]
NaN
Length [m]
NaN
Minor Loss Coefficient
NaN
Mode = 0: State Variables = Average Input variables, Mode = 1: State Variables = Input Higher Pressure
NaN
OK
Cancel
Help
Apply



In_1
1
<p>
<rho>
<T>
<x_H2O_gas>
<x_CO2>
<x_H2O_liq>
In_2
2
<p>
<rho>
<T>
<x_H2O_gas>
<x_CO2>
<x_H2O_liq>
D
L
zeta
mode
D
L
zeta
mode
p_in_1
rho_in_1
T_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
x_H2O_gas_in_2
x_CO2_in_2
x_H2O_liq_in_2
Cal
rho
T
x_H2O_gas
x_CO2
x_H2O_liq
p_1
m_dot_air_1
p_2
m_dot_air_2
rho
T
x_H2O_gas
x_CO2
x_H2O_liq
p
m_dot_air
p
m_dot_air
0
m_air
Out_1
1
2
Out_2

```
function [rho,T,x_H2O_gas,x_CO2,x_H2O_liq,p_1,m_dot_air_1,p_2,m_dot_air_2]=Cal(D,L,↙
zeta,mode,p_in_1,rho_in_1,T_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,x_H2O_gas_in_2,x_CO2_in_2,x_H2O_liq_in_2)

% ***************************************************************************
% * Definition of a flow resistance
% *
% * Number of inputs :               2
% *
% * Parameter: Diameter:             D
% *            Length:               L
% *            Minor loss coefficient: zeta
% *
% *
% * Relevant input variables of FR
% *
% * Pressure:                        p_in
% * Density:                         rho_in
% * Temperature:                     T_in
% * Content water vapor:             x_H2O_gas_in
% * Content CO2:                     x_CO2_in
% * Content water:                   x_H2O_liq_in
% *
% *
% * Relevant output variables of FR
% *
% * Temperature:                     T
% * Mass flow dry air:               m_dot_air
% * Content water vapor:             x_H2O_gas
% * Content CO2:                     x_CO2
% * Content water:                   x_H2O_liq
% *
% **************************************************************************
% * Embedded Matlab Function Cal:
% *
% * Calculations:
% * 1. Calculation parameter.
% * 2. Calculation of the state variables.
% * 3. Calculation average viscosity.
% * 4. Calculation flow velocity.
% *    4.1 Initial value of the flow velocity.
% * 5. Calculation mass flow.
% *
% *
% * Assumptions:
% * 2. Mode = 0: state variables = average input variables
% *    Mode = 1: state variables = input variables higher pressure
% * 3. The gas mixture inside the volume consists of water vapor (H2O_gas),
% *    CO2 and water (H2O_liq, state: fog).
% * 4. The intial value of the flow velocity is estimated by the law of
% *    Hagen-Poiseuille. Knowing the initial value of the velocity, the
% *    initial value of the Reynolds number can be calculated. The iterative
% *    while loop has a break condition of |v_(i-1)-v_(i)| < 0.001*v_(i-1).
% *    The input pressures are cross linked. p_in_1 is transfered to the output
% *    p_1 and the other way around.
% * 5. The mass flow is calculated with a incompressible mass flow equation.
```

```
% *    This equation is only applicable for systems with low flow velocities.
% *
% *
% * Last modification : 15.03.2008
% * Author : Christian Müller(HAW)
% *
% ************************************************************************

% * 1. Calculation parameter
A                   = (pi*(D/2)^2);
% ************************************************************************

% * 2. Calculation of the state variables.
rho                 = 0;
T                   = 0;
x_H2O_gas           = 0;
x_CO2               = 0;
x_H2O_liq           = 0;

if mode == 0
   rho              = (rho_in_1+rho_in_2)/2;
   T                = (T_in_1+T_in_2)/2;
   x_H2O_gas        = (x_H2O_gas_in_1+x_H2O_gas_in_2)/2;
   x_CO2            = (x_CO2_in_1+x_CO2_in_2)/2;
   x_H2O_liq        = (x_H2O_liq_in_1+x_H2O_liq_in_2)/2;
end

if mode == 1

   if p_in_1 >= p_in_2
      rho           = rho_in_1;
      T             = T_in_1;
      x_H2O_gas     = x_H2O_gas_in_1;
      x_CO2         = x_CO2_in_1;
      x_H2O_liq     = x_H2O_liq_in_1;
   else
      rho           = rho_in_2;
      T             = T_in_2;
      x_H2O_gas     = x_H2O_gas_in_2;
      x_CO2         = x_CO2_in_2;
      x_H2O_liq     = x_H2O_liq_in_2;
   end
end
% ************************************************************************

% * 3. Calculation average viscosity.
eta_air             = 0.0000182;
eta_H2O_gas         = 0.00000877;
eta_CO2             = 0.0000149;
eta                 = (eta_air+x_H2O_gas*eta_H2O_gas+x_CO2*eta_CO2)/↙
(1+x_H2O_gas+x_CO2);
eta_kin             = eta/rho;
% ************************************************************************

% * 4. Calculation flow velocity
% * 4.1 Initial value of the flow velocity.
```

```
p_1                 = p_in_2;
p_2                 = p_in_1;
Delta_p             = p_in_1-p_in_2;
V_dot               = ((pi*(D/2)^4)*abs(Delta_p))/(8*eta*L);
v                   = V_dot/A;
Re                  = (v*D)/eta_kin;
% **********************************************************************

% **********************************************************************
if Re < 1
   Re               = 1;
end

check               = 1;

while check > 0

      if Re < 1187.384382
         lambda     = 64/Re;
         zeta_total = lambda*(L/D)+zeta;
         Z_v        = sqrt(abs(Delta_p)*(2/rho)*(1/zeta_total));

         if abs(v-Z_v) > (0.001*v)
            check   = 1;
         else
            check   = 0;
         end

         v          = Z_v;
         Re         = (v*D)/eta_kin;

         if Re < 1
            Re      = 1;
         end
      else
         lambda     = 0.3164*((Re)^(-0.25));
         zeta_total = lambda*(L/D)+zeta;
         Z_v        = sqrt(abs(Delta_p)*(2/rho)*(1/zeta_total));

         if abs(v-Z_v) > (0.001*v)
            check   = 1;
         else
            check   = 0;
         end

         v          = Z_v;
         Re         = (v*D)/eta_kin;
      end
end
% **********************************************************************

% * 5. Calculation mass flow
m_dot               = A*rho*v;

if Delta_p >= 0
   m_dot_air_1      = -m_dot/(1+x_H2O_gas+x_CO2);
```

```
   m_dot_air_2      =  m_dot/(1+x_H2O_gas+x_CO2);
else
   m_dot_air_1      =  m_dot/(1+x_H2O_gas+x_CO2);
   m_dot_air_2      = -m_dot/(1+x_H2O_gas+x_CO2);
end
% ***********************************************************************
```