Simulink Report: Duct_Heat_Transfer_Wall_

Christian Müller

2008-03-06

Model - Duct_Heat_Transfer_Wall_

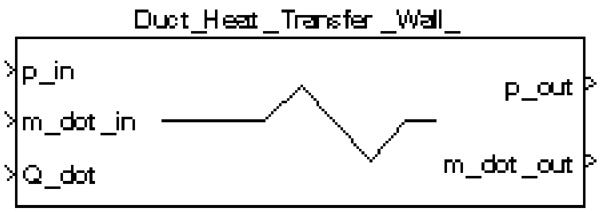


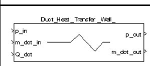
Tabelle 1.1. Duct_Heat_Transfer_Wall_ Simulation Parameters

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. Duct_Heat_Transfer_Wall_ 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**
Duct_Heat_Transfer_Wall_	<root>		Duct_Heat_Transfer_Wall_	Duct_Heat_Transfer_Wall_ <1> Duct_Heat_Transfer_Wall_ <2>

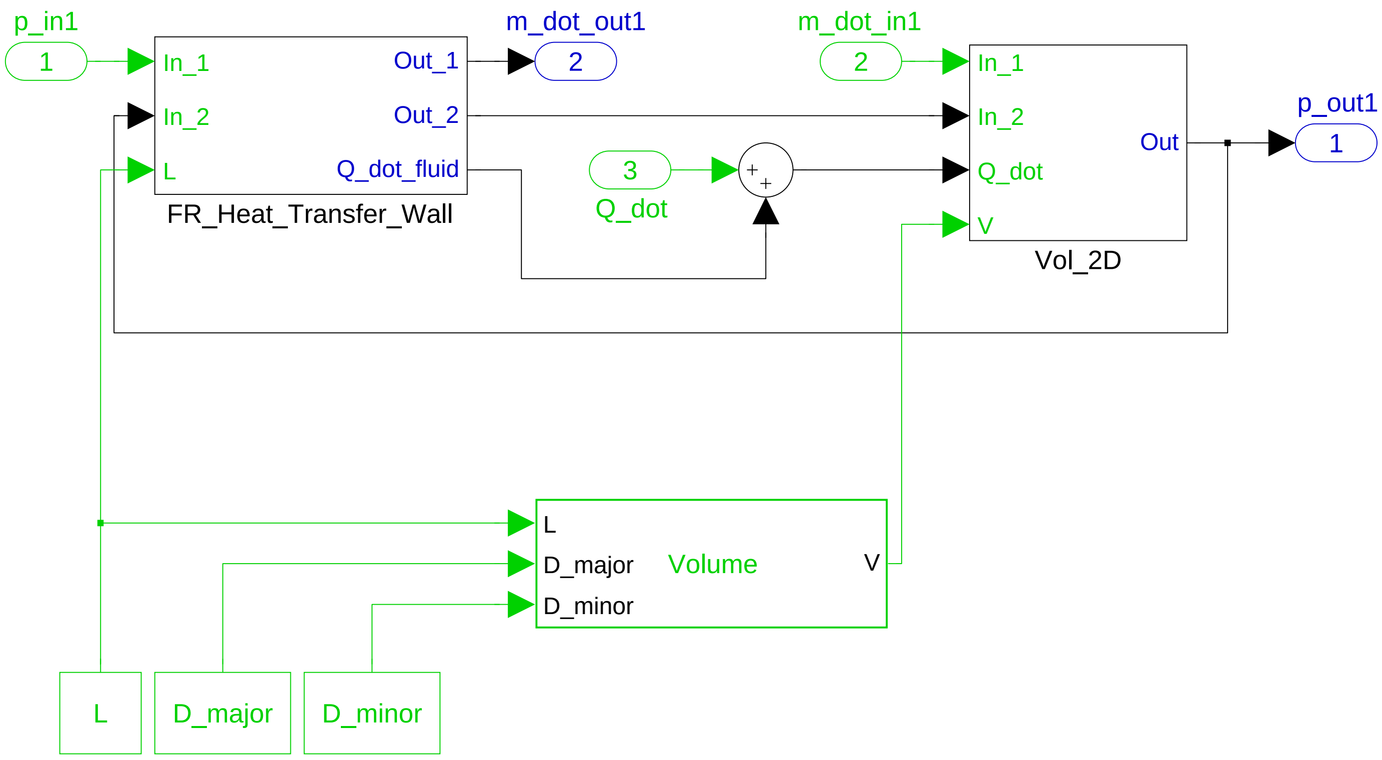
Blocks

Tabelle 1.3. Block Type Count

BlockType	**Count**	**Block Names**
Inport	69	p_in1, m_dot_in1, Q_dot, L, D_major, D_minor, In_1, In_2, L, n_unit, D_major, D_minor, b_wall, b_isolation, rho_wall, rho_isolation, c_wall, c_isolation, lambda_wall, lambda_isolation, L, zeta, mode, p_in_1, rho_in_1,

BlockType	**Count**	**Block Names**
		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, T_wall, T_ambient, In_1, In_2, Q_dot, V, 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
Outport	32	V, rho, T, x_H2O_gas, x_CO2, x_H2O_liq, p_1, m_dot_air_1, p_2, m_dot_air_2, Q_dot_fluid, T_dot_wall, Out_1, Out_2, Q_dot_fluid, 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, x_H2O_gas, x_CO2, x_H2O_liq, Out, p_out1, m_dot_out1
Constant	24	D_major, D_minor, Constant2, D_major, D_minor, T_ambient, b_isolation, b_wall, c_isolation, c_wall, lambda_isolation, lambda_wall, mode, n_unit, rho_isolation, rho_wall, zeta, L, T_init, m_dot_air, p_init, x_CO2_init, x_H2O_gas_init, x_H2O_liq_init
Integrator	6	Integrator, Integrator1, Integrator2, Integrator3, Integrator4, Integrator5
Terminator	5	Terminator , Terminator , Terminator , Terminator , Terminator
Stateflow (m)	5	Embedded MATLAB Function, Embedded MATLA Function, Embedded MATLAB Function, Embedded_MATLAB_Function_1, Embedded_MATLAB_Function_2
S-Function	5	SFunction , SFunction , SFunction , SFunction , SFunction
Demux	5	Demux , Demux , Demux , Demux , Demux
BusSelector	4	Bus Selector3, Bus Selector4, Bus Selector2, Bus Selector5
SubSystem	3	Duct_Heat_Transfer_Wall_, FR, Vol_2D
BusCreator	3	Bus Creator1, Bus Creator2, Bus Creator1
Sum	1	Sum


Function Block Parameters: Duct_Heat_Transfer_Wall_
Subsystem (mask)
Parameters
Number of identical Units
1
Length Major Axis [m]
NaN
Length Minor Axis [m]
NaN
Length [m]
NaN
Thickness Wall [m]
NaN
Thickness Isolation [m]
NaN
Density Wall [kg/m³]
NaN
Density Isolation [kg/m³]
NaN
Specific Heat Capacity Wall [J/kg/K]
NaN
Specific Heat Capacity Isolation [J/kg/K]
NaN
Thermal Conductivity Wall [J/s/m/K]
NaN
Thermal Conductivity Isolation [J/s/m/K]
NaN
Ambient Temperature [K]
NaN
Initial Parameter: Temperature Wall [K]
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
Minor Loss Coefficient
NaN
Mode = 0: State Variables = Average Input variables, Mode = 1: State Variables = Input Variables Higher Pressure
NaN
OK
Cancel
Help
Apply



p_in1
1
In_1
In_2
L
Out_1
Out_2
Q_dot_fluid
FR_Heat_Transfer_Wall
m_dot_out1
2
3
Q_dot
+
+
m_dot_in1
2
In_1
In_2
Q_dot
V
Out
Vol_2D
p_out1
1
L
D_major
D_minor
Volume
V
L
D_major
D_minor

```
function [rho,T,x_H2O_gas,x_CO2,x_H2O_liq,p_1,m_dot_air_1,p_2,m_dot_air_2,Q_dot_fluid,↙
T_dot_wall]=Cal(n_unit,D_major,D_minor,b_wall,b_isolation,rho_wall,rho_isolation,↙
c_wall,c_isolation,lambda_wall,lambda_isolation,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,T_wall,T_ambient)

% ***************************************************************************
% * Definition of a flow resistance with heat transfer via wall
% *
% * Number of inputs :                2
% *
% * Parameter: Diameter:              D
% *            Length:                L
% *            Minor loss coefficient: zeta
% *
% *
% * Relevant input variables of Duct_Heat_Transfer_Wall
% *
% * 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 Duct_Heat_Transfer_Wall
% *
% * 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.
% * 6. Calculation heat transfer.
% *
% *
% * 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.
% * 6. a) Heat transfer fluid/wall (convection)
% *    b) Heat transfer wall/isolation (conduction)
% *    c) Heat transfer isolation/enviroment (conduction)
% *
% *
% * Last modification : 15.03.2008
% * Author : Christian Müller(HAW)
% *
% ***********************************************************************

% * 1. Calculation parameter
D_major              = abs(D_major);
D_minor              = abs(D_minor);

if D_major == D_minor
   A                 = (pi/4)*D_major*D_minor;
   U_inner_duct      = pi*D_major;
   U_outer_duct      = pi*(D_major+b_wall);
   U_outer_isolation = pi*(D_major+b_wall+b_isolation);
   D                 = U_inner_duct/pi;
else
   A                 = (pi/4)*D_major*D_minor;
   K                 = abs(D_major-D_minor)/(D_major+D_minor);
   U_inner_duct      = (D_major+D_minor)*pi*(1+(3*K^2)/(10+sqrt(4-3*K^2)));
   K                 = abs((D_major+b_wall)-(D_minor+b_wall))/((D_major+b_wall)+↙
(D_minor+b_wall));
   U_outer_duct      = ((D_major+b_wall)+(D_minor+b_wall))*pi*(1+(3*K^2)/(10+sqrt(4-↙
3*K^2)));
   K                 = abs((D_major+b_wall+b_isolation)-(D_minor+b_wall+b_isolation))/↙
((D_major+b_wall+b_isolation)+(D_minor+b_wall+b_isolation));
   U_outer_isolation = ((D_major+b_wall+b_isolation)+(D_minor+b_wall+b_isolation))*pi*↙
(1+(3*K^2)/(10+sqrt(4-3*K^2)));
   D                 = U_inner_duct/pi;
end
% ***********************************************************************

% * 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                = n_unit*A*rho*v;

if Delta_p>=0
   m_dot_air_1       = -m_dot*(1-((x_H2O_gas+x_CO2)/(1+x_H2O_gas+x_CO2)));
   m_dot_air_2       =  m_dot*(1-((x_H2O_gas+x_CO2)/(1+x_H2O_gas+x_CO2)));
else
   m_dot_air_1       =  m_dot*(1-((x_H2O_gas+x_CO2)/(1+x_H2O_gas+x_CO2)));
   m_dot_air_2       = -m_dot*(1-((x_H2O_gas+x_CO2)/(1+x_H2O_gas+x_CO2)));
end
% ************************************************************************

% * 6. Calculation heat transfer
c_p_air              = 1005;
c_p_H2O_gas          = 1870;
c_p_CO2              = 830;
c_p                  = (c_p_air+x_H2O_gas*c_p_H2O_gas+x_CO2*c_p_CO2)/↙
(1+x_H2O_gas+x_CO2);

% * Thermal conductivity dry air
lambda_air           = 0.00452+T*7.28282E-5;
% ************************************************************************

Pr                   = eta*c_p/lambda_air;
if Re < 1187.384382
   Nu                = ((3.66^3)+(1.66^3)*Re*Pr*(U_inner_duct/(pi*L)))^(1/3);
else
   Nu                = 0.012*(Re^(0.87)-280)*Pr^(0.4)*(1+(U_inner_duct/(pi*L))^(2/3));
end

% * Convection heat transfer coefficient dry air
alpha_air            = lambda_air*Nu/(U_inner_duct/pi);
% ************************************************************************
```

```
Q_dot_fluid_wall     = alpha_air*(U_inner_duct*L)*(T-T_wall);
Q_dot_wall_ambient   = 2*pi*L*(T_ambient-T_wall)/((log(U_outer_isolation/U_outer_duct)↙
/lambda_isolation)+(log(U_outer_duct/U_inner_duct)/lambda_wall));
Q_dot_wall           = Q_dot_fluid_wall+Q_dot_wall_ambient;
V_wall               = L*U_inner_duct*b_wall;
V_isolation          = L*U_outer_duct*b_isolation;
T_dot_wall           = Q_dot_wall/↙
(V_wall*rho_wall*c_wall+V_isolation*rho_isolation*c_isolation);
Q_dot_fluid          = -Q_dot_fluid_wall ;
% **************************************************************************
```