Simulink Report: Generalized_Fan_

Christian Müller

2008-03-06

Model - Generalized_Fan_

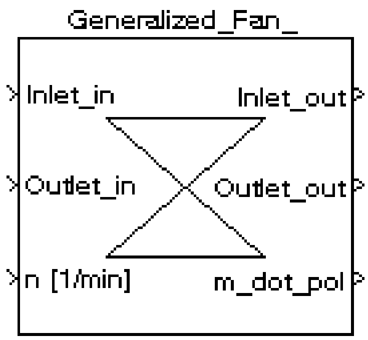


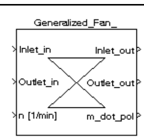
Tabelle 1.1. Generalized_Fan_ Simulation Parameters

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. Generalized_Fan_ 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**
Generalized_ Fan_	<root>		Generalized_Fan_	Generalized_Fan_<1> Generalized_Fan_<2> Generalized_Fan_<3>

Blocks

Tabelle 1.3. Block Type Count

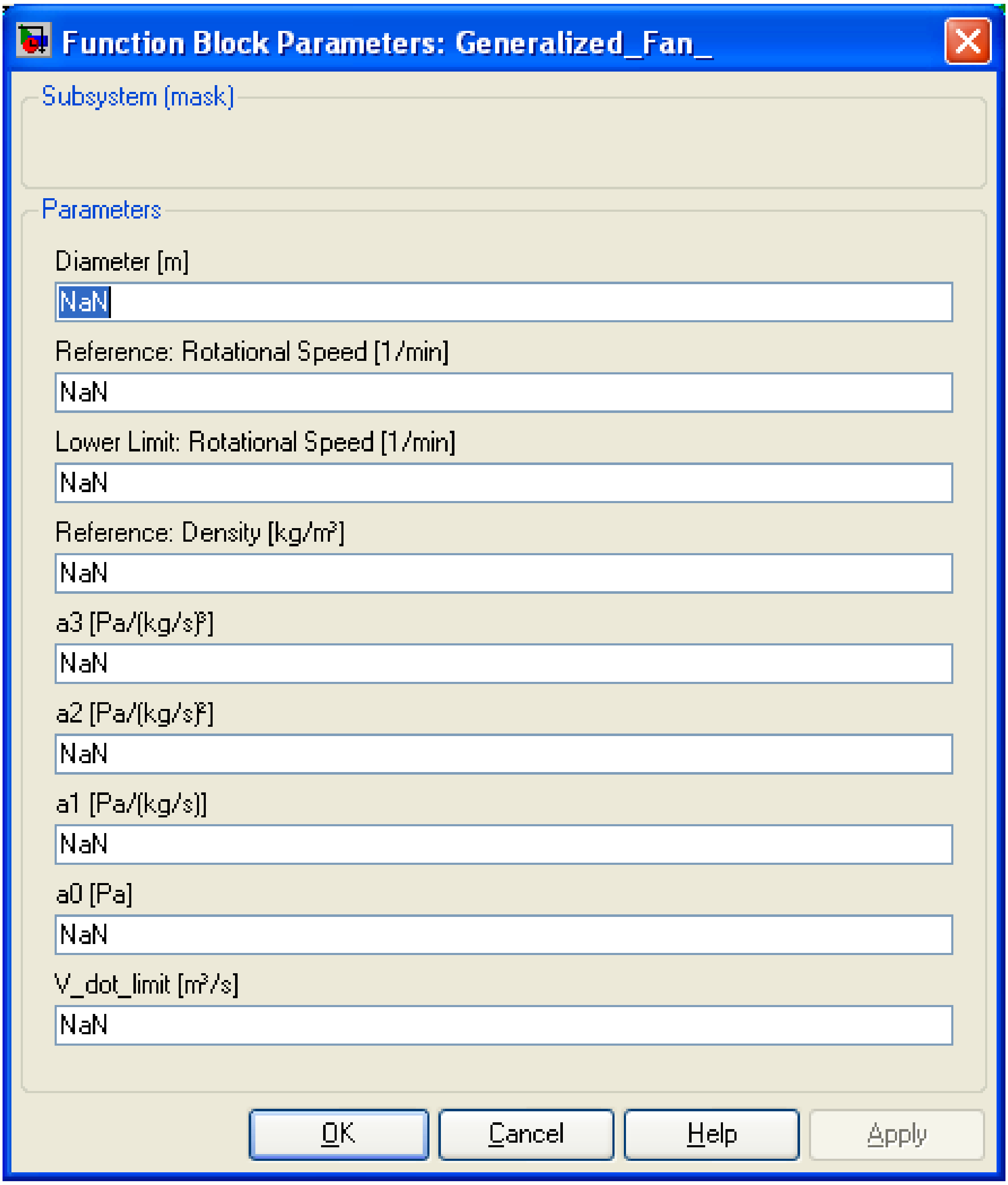
BlockType	**Count**	**Block Names**
Inport	22	Inlet_In, Outlet_In, n, n, n_0, n_min, rho_0, a_0, a_1, a_2, a_3, V_dot_limit, K, p_in, rho_in, T_in, x_H2O_gas_in,

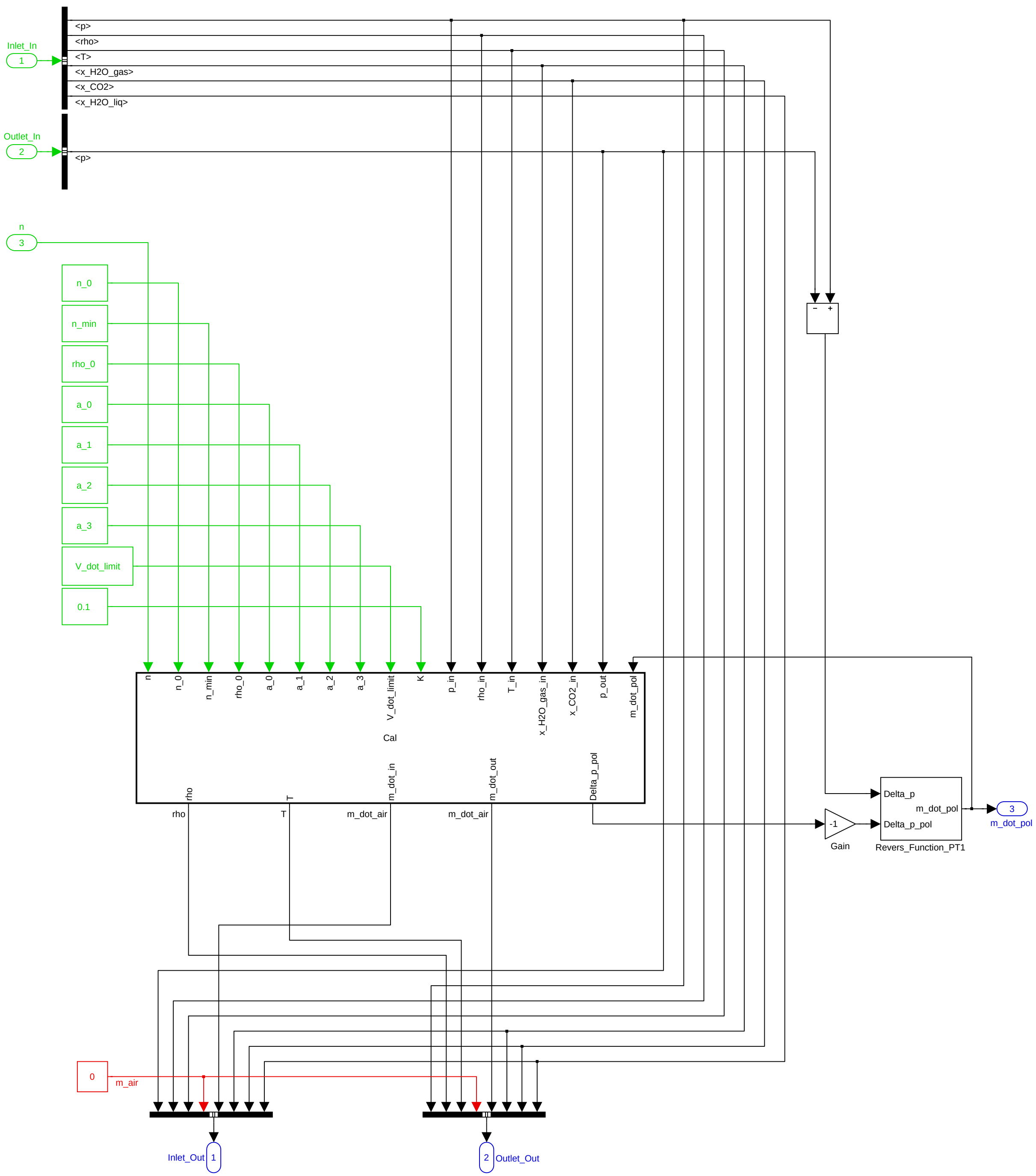
BlockType	Count	Block Names
		x_CO2_in, p_out, m_dot_pol, Delta_p, Delta_p_pol
Constant	12	K, Constant1, Constant2, V_dot_limit, a_0, a_1, a_2, a_3, m_air, n_0, n_min, rho_0
Outport	9	rho, T, m_dot_in, m_dot_out, Delta_p_pol, m_dot_pol, Inlet_Out, Outlet_Out, m_dot_pol
Sum	2	Add, Sum
SubSystem	2	Generalized_Fan_, Revers_Function_PT1
Product	2	Product1, Product2
BusSelector	2	Bus Selector3, Bus Selector4
BusCreator	2	Bus Creator1, Bus Creator2
Terminator	1	Terminator
Stateflow (m)	1	Embedded MATLA Function
S-Function	1	SFunction
Integrator	1	Integrator
Gain	1	Gain
Demux	1	Demux

Data and Functions

Tabelle 1.4. Model Functions

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


Function Block Parameters: Generalized_Fan_
Subsystem (mask)
Parameters
Diameter [m]
NaN
Reference: Rotational Speed [1/min]
NaN
Lower Limit: Rotational Speed [1/min]
NaN
Reference: Density [kg/m³]
NaN
a3 [Pa/(kg/s)³]
NaN
a2 [Pa/(kg/s)²]
NaN
a1 [Pa/(kg/s)]
NaN
a0 [Pa]
NaN
V_dot_limit [m³/s]
NaN
OK
Cancel
Help
Apply



Inlet_In
1
<p>
<rho>
<T>
<x_H2O_gas>
<x_CO2>
<x_H2O_liq>
Outlet_In
2
<p>
n
3
n_0
n_min
rho_0
a_0
a_1
a_2
a_3
V_dot_limit
0.1
n
n_0
n_min
rho_0
a_0
a_1
a_2
a_3
V_dot_limit
K
p_in
rho_in
T_in
x_H2O_gas_in
x_CO2_in
p_out
m_dot_pol
Cal
rho
T
m_dot_in
m_dot_out
Delta_p_pol
rho
T
m_dot_air
m_dot_air
- +
Delta_p
m_dot_pol
Delta_p_pol
Revers_Function_PT1
-1
Gain
3
m_dot_pol
0
m_air
Inlet_Out
1
2
Outlet_Out

```
function [rho,T,m_dot_in,m_dot_out,Delta_p_pol] = Cal(n,n_0,n_min,rho_0,a_0,a_1,a_2,↙
a_3,V_dot_limit,K,p_in,rho_in,T_in,x_H2O_gas_in,x_CO2_in,p_out,m_dot_pol)

% *************************************************************************
% * Definition of a generalized fan
% *
% * Number of inputs :                 3
% *
% * Parameter: Diameter:               D
% *            Reference:              n_0, rho_0
% *            Fan Curve:              a_0, a_1, a_2, a_3
% *
% *
% * Relevant input variables of Generalized_Fan
% *
% * 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 Generalized_Fan
% *
% * 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. Definition specific gas constants.
% * 2. Definition state variables.
% * 3. Calculation mass flow.
% *
% *
% FAN CURVE
% =========
% Delta_p_zero -|-------_______              ->|    |<- volume_margin
%               |             -------_______        |
% p_delta_end  -|------------------------------E____|
%               |                              |    ---__
%               |                              |    |    --_
%             0 |______________________________|____|_______-______
%                0                   V_dot_end      |
%                                              V_dot_slope
%                <------  linear curve  ------>|<-- cubic curve -->
% *
% *
% * 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;

R_avg                      = (R_air+x_H2O_gas_in*R_H2O_gas+x_CO2_in*R_CO2)/↙
(1+x_H2O_gas_in+x_CO2_in);
c_p_avg                    = (c_p_air+x_H2O_gas_in*c_p_H2O_gas+x_CO2_in*c_p_CO2)/↙
(1+x_H2O_gas_in+x_CO2_in);
c_v_avg                    = c_p_avg-R_avg;
gamma_avg                  = c_p_avg/c_v_avg;
% **********************************************************************

% * 2. Definition state variables
rho                        = rho_in*((p_out/p_in)^(1/gamma_avg));
rho_avg                    = (rho_in+rho)/2;
T                          = T_in*((p_out/p_in)^((gamma_avg-1)/gamma_avg));
% **********************************************************************

% * 3. Calculation mass flow
Delta_p                    = 0;
Delta_p_end                = 0;
Delta_p_pol                = 0;
V_dot                      = 0;
V_dot_end                  = 0;
m_dot_in                   = 0;
m_dot_out                  = 0;
d_Delta_p_d_V_dot          = 0;

a_1                        = abs(a_1);
a_2                        = -abs(a_2);
a_3                        = -abs(a_3);
A_3                        = (rho_avg/rho_0)*a_3*(n/n_0)^(-1);
A_2                        = (rho_avg/rho_0)*a_2*(n/n_0)^0;
A_1                        = (rho_avg/rho_0)*a_1*(n/n_0)^1;
A_0                        = (rho_avg/rho_0)*a_0*(n/n_0)^2;
V_dot_limit                = abs(V_dot_limit);

if n>0
   Delta_p                 = abs(p_out-p_in);
   n                       = max(n,n_min);

   if A_3<0
      V_dot_end_1          = (-2*A_2+sqrt((2*A_2)^2-12*A_3*A_1))/(6*A_3);
      V_dot_end_2          = (-2*A_2-sqrt((2*A_2)^2-12*A_3*A_1))/(6*A_3);
      V_dot_end            = max(V_dot_end_1,V_dot_end_2);
      Delta_p_end          = A_3*V_dot_end^3+A_2*V_dot_end^2+A_1*V_dot_end+A_0;

      if Delta_p_end < 0
         Delta_p_end       = 0;
      end
```

```
      if Delta_p <= Delta_p_end
         A_max             = max(abs(A_3),abs(A_2));
         A_max             = max(A_max,abs(A_1));
         A_max             = max(A_max,abs(A_0));
         Z_A_3             = A_3/A_max;
         Z_A_2             = A_2/A_max;
         Z_A_1             = A_1/A_max;
         Z_A_0             = A_0/A_max;
         Z_Delta_p         = Delta_p/A_max
         K_1               = 3*Z_A_3*Z_A_1-(Z_A_2^2);
         K_2               = 27*(Z_A_3^2)*(Z_A_0-Z_Delta_p)-9*Z_A_3*Z_A_2*Z_A_1+2*↙
(Z_A_2^3);
         K_3               = 0.5*((-4*K_2+4*sqrt((K_2^2)+4*(K_1^3)))^(1/3));
         K_4               = 0.5*((-4*K_2-4*sqrt((K_2^2)+4*(K_1^3)))^(1/3));
         V_dot             = (K_3+K_4-Z_A_2)/(3*Z_A_3);
      end

      if Delta_p > Delta_p_end
         Delta_V_dot       = 0.1*V_dot_end;
         d_Delta_p_d_V_dot = (3*A_3*(V_dot_end+Delta_V_dot)^2+2*A_2*↙
(V_dot_end+Delta_V_dot)+A_1);
         d_Delta_p_d_V_dot = max(abs(d_Delta_p_d_V_dot),0.1);
         Delta_p_zero      = Delta_p_end+d_Delta_p_d_V_dot*V_dot_end;
         V_dot_virtuell    = (Delta_p-Delta_p_end)/d_Delta_p_d_V_dot;
         V_dot             = V_dot_end-V_dot_virtuell;
      end

      if V_dot > V_dot_limit
         V_dot             = V_dot_limit+K*log(1+(V_dot-V_dot_limit));
      end

      if V_dot < -V_dot_limit
         V_dot             = -V_dot_limit-K*log(1+abs(V_dot+V_dot_limit));
      end

      m_dot                = V_dot*rho_avg;
      m_dot_in             = -m_dot/(1+x_H2O_gas_in+x_CO2_in);
      m_dot_out            =  m_dot/(1+x_H2O_gas_in+x_CO2_in);
      V_dot_pol            = m_dot_pol/rho_avg;
      Delta_p_pol          = A_3*V_dot_pol^3+A_2*V_dot_pol^2+A_1*V_dot_pol+A_0;

      if V_dot_pol<V_dot_end
         Delta_p_pol       = Delta_p_end+d_Delta_p_d_V_dot*(V_dot_end-V_dot_pol);
      end
   end

   if A_3==0
      if A_2 == 0
         V_dot_end         = 0;
         Delta_p_end       = 0;
         V_dot             = 0;
      end

      if A_2 < 0
         V_dot_end         = -A_1/(2*A_2);
```

```
         Delta_p_end       = A_2*V_dot_end^2+A_1*V_dot_end+A_0;

         if Delta_p_end < 0
            Delta_p_end    = 0;
         end
      end

      if Delta_p <= Delta_p_end
         V_dot_1           = (-A_1+sqrt(A_1^2-4*A_2*(A_0-Delta_p)))/(2*A_2);
         V_dot_2           = (-A_1-sqrt(A_1^2-4*A_2*(A_0-Delta_p)))/(2*A_2);
         V_dot             = max(V_dot_1,V_dot_2);
      end

      if Delta_p > Delta_p_end
         Delta_V_dot       = 0.1*V_dot_end;
         d_Delta_p_d_V_dot = (2*A_2*(V_dot_end+Delta_V_dot)+A_1);
         d_Delta_p_d_V_dot = max(abs(d_Delta_p_d_V_dot),0.1);
         Delta_p_zero      = Delta_p_end+d_Delta_p_d_V_dot*V_dot_end;
         V_dot_virtuell    = (Delta_p-Delta_p_end)/d_Delta_p_d_V_dot;
         V_dot             = V_dot_end-V_dot_virtuell;
      end

      if V_dot > V_dot_limit
         V_dot             =  V_dot_limit+K*log(1+(V_dot-V_dot_limit));
      end

      if V_dot < -V_dot_limit
         V_dot             = -V_dot_limit-K*log(1+abs(V_dot+V_dot_limit));
      end

      m_dot                = V_dot*rho_avg;
      m_dot_in             = -m_dot/(1+x_H2O_gas_in+x_CO2_in);
      m_dot_out            =  m_dot/(1+x_H2O_gas_in+x_CO2_in);
      V_dot_pol            = m_dot_pol/rho_avg;
      Delta_p_pol          = A_2*V_dot_pol^2+A_1*V_dot_pol+A_0;

      if V_dot_pol < V_dot_end
         Delta_p_pol       = Delta_p_end+d_Delta_p_d_V_dot*(V_dot_end-V_dot_pol);
      end
   end
end
% ************************************************************************
```