OPEN SOURCE ENGINEERING

openeering

A SCILAB PROFESSIONAL PARTNER

powered by ENGINSOFT

AN AERAULIC TOOLBOX FOR XCOS

Authors: *M. Venturin*

Keywords: Aeraulic system; Heating, Ventilation, Air Conditioning (HVAC); Scilab; Xcos; Modelica

Abstract: The aim of this paper is to show the possibilities offered by Scilab/Xcos to model and simulate aeraulic and HVAC systems. In particular we develop a new Xcos module combined with the use of Modelica language to show how aeraulic systems can be modeled and studied. In this paper we construct a reduced library composed by few elements: hoods, pipes and ideal junctions. The library can be easily extended to obtain a more complete toolbox. The developed library is tested on a simple aeraulic circuit.

Contacts info@openeering.com

1. Introduction

The study of aeraulic systems plays today an import role for maintaining adequate indoor air quality and thermal comfort. Moreover, studying and optimizing aeraulic systems can reduce energy consumption in buildings due, for example, to heating and cooling or to drying/humidifying control systems.

Progress in airflow analysis made it possible to design building ventilation quantitatively and qualitatively. For this kind of analysis, the main primary thermal-fluid parameters under consideration are velocity, turbulence, temperature and humidity.

Airflow problems in buildings can be treated at various levels depending on the different stages of the design process.

The two main categories of fluid flow analysis are:

- macroscopic air flow in which methods are based on modeling the air flow in building including heating, ventilating and air-conditioning (HVAC) systems as a collection of finite-size control volumes which leads to differential equations with lumped mass parameters;
- microscopic air flow or computational fluid dynamic (CFD) where methods are based on a continuum (spatial and time) approach that provides detailed descriptions of the flow, heat and mass transport processes which leads to partial differential equations.

Here we focus on the first category; the aim of this paper is to develop an aeraulic systems toolbox in Scilab/Xcos for macroscopic air flow analysis. For sake of simplicity, the system under consideration is composed of a limited number of blocks. In particular the following blocks will be presented:

- pipes;
- hoods;
- fans;
- and ideal junctions.

In further studies, the developed library can be straightforward extended with other resistive elements like bends, elbows or cross-sectional flow area changes or it can be extended with the development of new elements like valves, orifices and tanks.

The computation toolbox is developed in the Scilab/Xcos engineering environment. An additional requirement concerns the mathematical modeling approach via Model-based design. The adopted strategy is based on the use of the Modelica language that reflects more closely the visual structure of simulated circuits.

Moreover, the use of Modelica language allows the possibility to develop independent libraries of physical components, which are easy to re-use due to this separation from the simulation package.

The paper is organized as follows. First we present the basic modeling strategy, then the constitutive laws of the network elements and after we present an application of the developed library to simulate an aeraulic circuit.

2. Library modeling

In this section we describe the basic library modeling approach. The Modelica developed package is named “Aeraulics” and it is contained in the file “Aeraulics.mo”.

The implementation of the toolbox is done in Scilab/Xcos through the use Modelica features. The first step is to identify through and across system variables and the use of the “passive sign convention[1]” for all elements. For our problem we choose:

- the volumetric flow rate q [m3/s] as the through variable;
- the pressure p [Pa] as the across variable.

This is implemented in the connector class named “Pin”:

```
connector Pin
          Real p "[Pa] pressure";
          flow Real q "[m^3/s] volumetric flow";
end Pin;
```

Next, since all library elements have two pins, we develop a partial model class named “TwoPin”:

```
partial model TwoPin
          Pin air_p, air_n;
          Real q, p;
equation
          q = air_p.q;
          air_n.q = -q;
          p = air_p.p - air_n.p;
end TwoPin;
```

which is very useful since it simplifies writing all the following constitutive laws.

The previous class implements the basic conservation laws for a two pins element:

```
Flux of the element  = Flux element at the input node
                     = - Flux element at the output node
```

and

```
Pressure drop        = Pressure at the input node
                     - Pressure at the output node
```

[1] In the “user convention” the through variable enters the positive terminal of a component (in Scilab is denoted by the black square).

3. Element constitutive laws and properties

In this section we summarize the constitutive laws and properties that have been used to develop the toolbox.

3.1. Air properties

Air properties are common to almost all elements and, hence, are treated as constants in the library. Here we list the air properties that are used in the constitutive laws of aeraulic elements:

- air density $\rho = 1.205$ [kg/m^3];
- air kinematic viscosity $\nu = 12.68 \times 10^{-6}$ [m^2/s];
- minimum volumetric flow $q_{\min} = 10^{-3}$ [m^3/s].

The minimum volumetric flow is used as a numerical trick for better convergence of models. In particular, it useful to linearize the constitutive laws when values are close to zero.

The library constants are defined as follows:

```
package Constants "Air fluid constants"
          constant Real rho = 1.205 "[kg/m^3] air density";
          constant Real nu = 1.268e-005 "[m^2/s] air kinematic viscosity";
          constant Real qmin = 1e-3 "[m^3/s] minimum volumetric flow";
end Constants;
```

3.2. Generic resistive elements

The mathematical model of a generic resistive element is the basis of all the other elements (like bend, hood, elbow, pipe, …) since it describes a generic aeraulic resistance. The pressure drop caused by the resistance is computed by loss coefficients that are generally provided in catalogs or manuals. See for example I.E. Idelchik and M. O. Steinberg, Handbook of Hydraulic resistance, Jaico Publishing House, 2011.

The pressure drop equation reads as follows:

$$\Delta p = K \frac{\rho}{2A^2} q|q|$$

where

- Δp is the pressure drop [Pa];
- q is the volumetric flow rate [m^3/s];
- K is the loss coefficient that depends on the resistive element [-];
- A is the cross section area of the elements [m^2];
- ρ is the air fluid density [kg/m^3].

Some limitations are used to obtain this simplified formula. For a more accurate modeling, this equation can be improved considering the regime of the fluid that depends on the local Reynolds number and temperature effects on fluid density. Moreover, K is not generally constant but depends on local Reynolds number. We recall that the purpose of this tutorial is not to give a detailed mathematical formulation of the considered problem but to show the possibilities offered by Scilab to the aspect of the mathematical modeling of the physical system. For more real-case applications the interested reader can contact the Openeering team.

As an example in the following figure we plot the pressure drop Δp as a function of the volumetric flow rate q of a resistive element with $K = 1$, $A = 0.1256$ m^2 (resistive pipe with a circular diameter of 40 cm) and density $\rho = 1.205$ kg/ m^3.

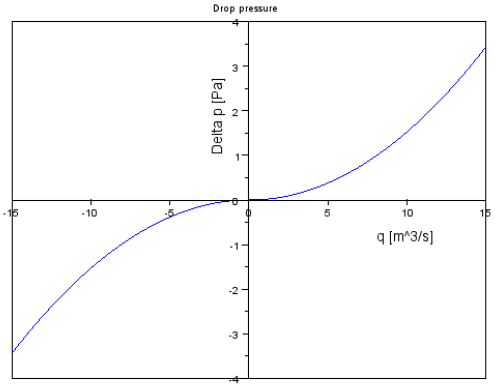


Figure 1: Example of pressure drop in a generic element.

3.2.1. Resistive pipe

A resistive pipe is modeled as a generic resistive element where the loss coefficient K is given as

$$K = f\frac{L}{D_H}$$

where

- L is the length of the pipe [m];
- D_H is the hydraulic diameter [m];
- $f = \frac{K_S}{Re}$ is the Darcy friction factor [-];
- K_S is the shape cross section factor which is equal to 64 for circular cross section pipe [-];
- $Re = \frac{q \cdot D_H}{A \cdot \nu}$ is the element Reynolds number [-];
- q is the volumetric flow rate [m^3/s];
- A is the cross section area of the elements [m^2];
- ν is the kinematic viscosity [m^2/s].

Re is limited by the relation $Re \geq Re_{min}$ where Re_{min} is computed from the constant q_{min}, this limit guarantees a more robust behavior of models.

The default values are in our case:

- shape cross section factor $K_S = 64$ [-];
- hydraulic diameter $D_H = 0.4$ [m];
- pipe cross section area $A = 0.1257$ [m^2] which corresponds to a diameter of 40 [cm];
- pipe length $L = 1$ [m].

As an example in the following figure we plot the pressure drop Δp as a function of the volumetric flow rate q of a resistive pipe long 10 m, cross section area $A = 0.1257$ m^2 (circular cross section area with a diameter of 40 cm) and kinematic viscosity $\nu = 12.68 \times 10^{-6}$ m^2/s.

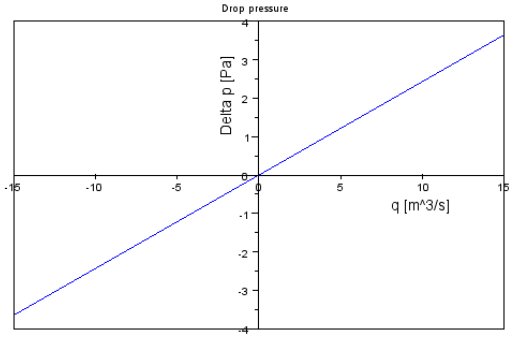


Figure 2: Example of pressure drop in a resistive pipe.

The Modelica implementation is done in the class “Pipe”:

```
model Pipe "Pipe"
          extends TwoPin;
          parameter Real Ks = 64 "[-] shape cross section area";
          parameter Real DH = 0.4 "[m] cross section hydraulic diameter";
          parameter Real A = 0.1257 "[m^2] cross section area";
          parameter Real L = 1 "[m] pipe length";

          Real REmin "[-] minimum local Reynolds number";
          Real REloc "[-] local Reynolds number";
          Real REeff "[-] used Reynolds number";
          Real f "[-] Darcy friction factor";
          Real K "[-] loss coefficient";
protected
          constant Real rho = Aeraulics.Constants.rho;
          constant Real nu = Aeraulics.Constants.nu;
          constant Real qmin = Aeraulics.Constants.qmin;
equation
          REmin = (qmin * DH) / (A * nu);
          REloc = (q * DH) / (A * nu);
          REeff = if REloc < REmin then REmin else REloc;
          f = Ks / REeff;
          K = (f * L) / DH;
          p = (K * rho) / (2.0 * A * A) * q * abs(q);
end Pipe;
```

3.2.2. Hood

Hood element is a particular case of resistance and hence it has the same formula for the pressure drop, which is equal to:

$$\Delta p = K \frac{\rho}{2A^2} q|q|.$$

For the system described above the value of K is considered to be constant.

In our implementation we use a linearized version of the formula when the value is $q \leq q_{\min}$

$$\Delta p = K_{\min} \frac{\rho}{2A^2} q.$$

The default values used in our case are:

- loss coefficient $K = 0.65$ [-];
- hood pipe cross section area $A = 0.1257$ [m^2] which corresponds to a diameter of 40 [cm].

The Modelica implementation is done in the class “Hood”:

```
model Hood "Hood"
          extends TwoPin;
          parameter Real K = 0.65 "[-] loss coefficient";
          parameter Real A = 0.1257 "[m^2] cross section area";

          Real Kmin;
protected
          constant Real rho = Aeraulics.Constants.rho;
          constant Real qmin = Aeraulics.Constants.qmin;
equation
          Kmin = K * qmin;
          p = if abs(q) < qmin then (Kmin * rho) / (2.0 * A * A) * q else (K *
rho) / (2.0 * A * A) * q * abs(q);
end Hood;
```

3.2.3. Ground

When composing a scheme, it is always necessary to have a reference element. In this case, our reference element is the “ground element” that fixes the pressure at a given node. The equation for the ground element is

$$p = 0.$$

The Modelica implementation is done in the class “Ground”:

```
model Ground "Ground, fix pressure"
          Pin air_p;
equation
          air_p.p = 0.0;
end Ground;
```

3.2.4. Fan element

Typically fan performance curves are provided by tabular value. Here we use a simplify version based on an ideal flux generator in parallel with a resistive element. Moreover, in order to guarantee that at the initial time everything is in equilibrium, we use a linear time variation for the flux generator.

For more details, see the Modelica implementation in the class “Fan”:

```
model Fan "Fan"
          extends TwoPin;
          parameter Real qmax = 300.0 / 3600 "[m^3/s] max volumetric flow rate";
          parameter Real K = 10 "[-] loss coefficient";
          parameter Real timemax = 1 "[s] maximum time";

          Real q1;
```

```
          Real q2;
          Real Klin;
protected
          constant Real qmin = Aeraulics.Constants.qmin;
equation
          q = q1 + q2;
          q1 = if time < timemax then qmax / timemax * time else qmax;
          Klin = K * qmin;
          p = if abs(q2) < qmin then Klin * q2 else K * q2 * abs(q2);
end Fan;
```

3.2.5. Pressure and Flux sensors

If we want to exchange data from Modelica to Scilab/Xcos two other blocks are necessary. The first block is used to convert the across variable “pressure” to a Scilab/Xcos variable. The equation of this element is

$$q = 0.$$

The second block is used to convert the through variable “volumetric flow” to a Scilab/Xcos variable. The equation of this element is

$$\Delta p = 0.$$

The Modelica implementation is done in the classes “FluxSensor”:

```
model FluxSensor
          extends TwoPin;
equation
          p = 0.0;
end FluxSensor;
```

and “PressureSensor”:

```
model PressureSensor
          extends TwoPin;
equation
          q = 0.;
end PressureSensor;
```

3.2.6. Junctions

In our simplified model, all junctions are considered as ideal. This means that there is not loss of pressure in the junctions. Hence it is not necessary to develop new elements with particular configurations since Xcos generate the conservation of law for each connected nodes.

4. Example

The developed library is tested in the following example. The problem under consideration consists of three hoods with a unique fan. The object is to study the pressure and flow distribution into the circuit.

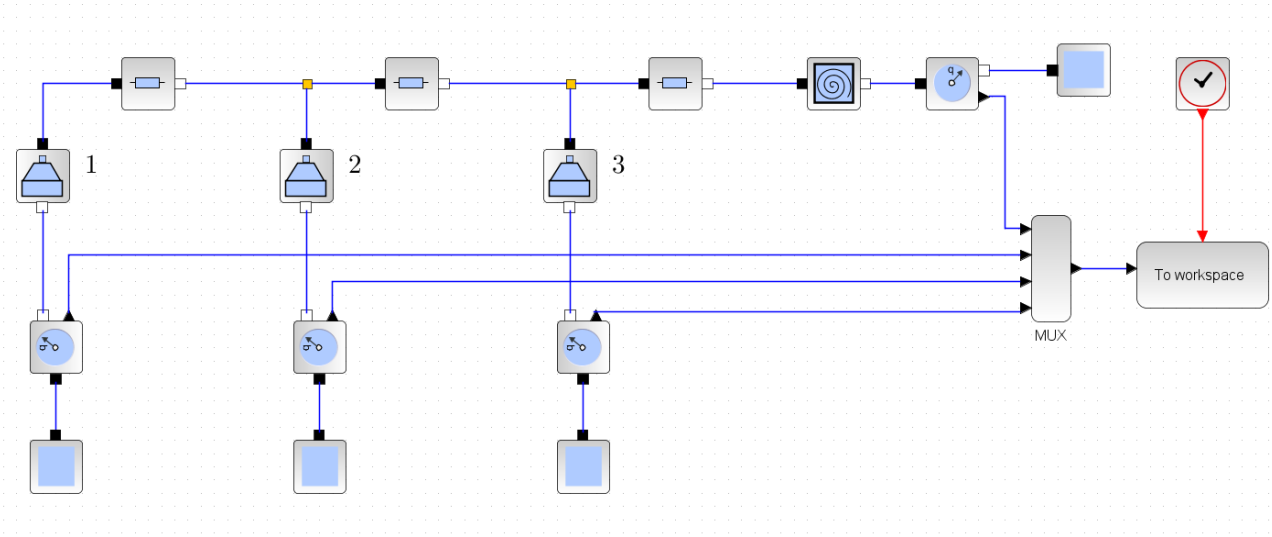


Figure 3: The aeraulic problem in the Xcos scheme

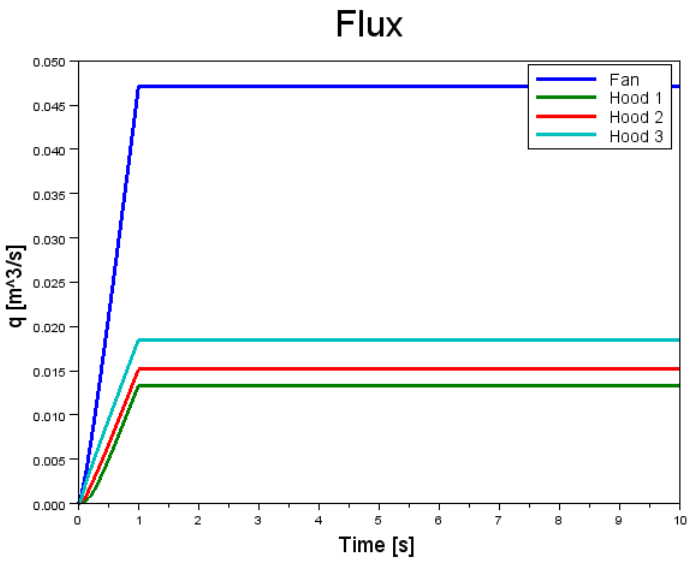


Figure 4: Numerical results

As results we plot time evolution of the fluxes through the hoods and the fan.

5. Conclusion

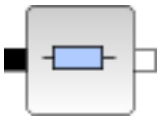
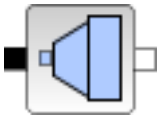
Here, we have developed a Scilab/Xcos toolbox for aeraulic simulation. The toolbox will be extended in the future by providing other elements.

6. References

[1] I.E. Idelchik and M. O. Steinberg, Handbook of Hydraulic resistance, Jaico Publishing House, 2011.

Appendix A – Library description

In the following table we report all the developed elements of the aeraulic library.

Element	Required parameters	Xcos element
Pipe	K_s : Shape cross section factor [m] (default 64) D_H : Hydraulic diameter [m] (default 0.4) A : Cross section area [m^2] (default 0.1257) L : Pipe length [m]; (default 1.0)	
Hood	K : Loss coefficient [-] (default 0.65) A : Cross section area [m^2] (default 0.1257)	
Fan	K : Loss coefficient of parallel resistance [-] (default 10.0) q_{max} : Maximum flux [m^3/s] (default $300/3600$) t_{max} : Time at which the max is reached [s] (default 1.0)	
Ground	—	
Pressure sensor	—	
Flux sensor	—	

Appendix B – Software installation and testing

The “Aeraulic system toolbox” is available to registered users only. For downloading the library visit:

http://www.openeering.com/registered_users_area

Registration is free and automatic.

To load the library into the Xcos environment type

```
--> exec loader.sce
```

from the main directory. The new set of palette is loaded in your Xcos system.

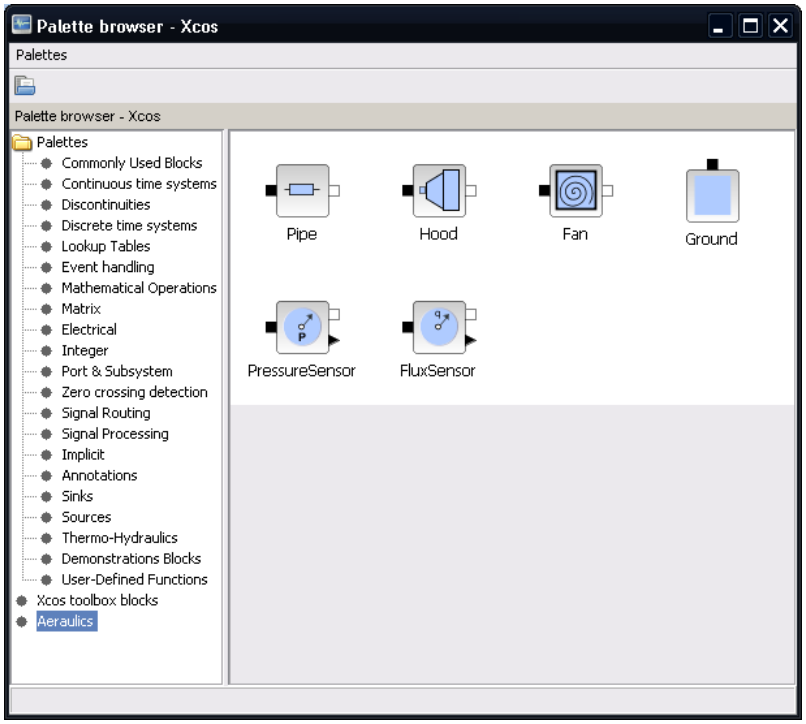


Figure 5: The Aeraulic library in Xcos

Open the Xcos model file "`airSystemTest.xcos`" and run it. Remember that to run an Xcos model with Modelica blocks, a C compiler should be installed in your system. Check if a compiler is available in your system with the Scilab command "`haveacompiler()`".

Then plot the data using the function "`airSystemPlotData.sce`".