ColdBattery component model

The external divider "cold battery" is used to model the cooling and condensation of water in moist gases. This is a simplified model, which has two parameters, the water temperature and the extraction efficiency of water ε, which represents the percentage (by volume) of the condensed water from the incoming water.

Modeling a cooling coil with condensation in Thermoptim

A cooling coil with condensation acts as a divider receiving the humid air coming in and out of which exit two fluids: water and drier air. It is cooled by a fluid, and the thermal coupling can be represented by a thermocoupler. To simplify writing, we talk about air, but this component can cool and condense any moist mixture.

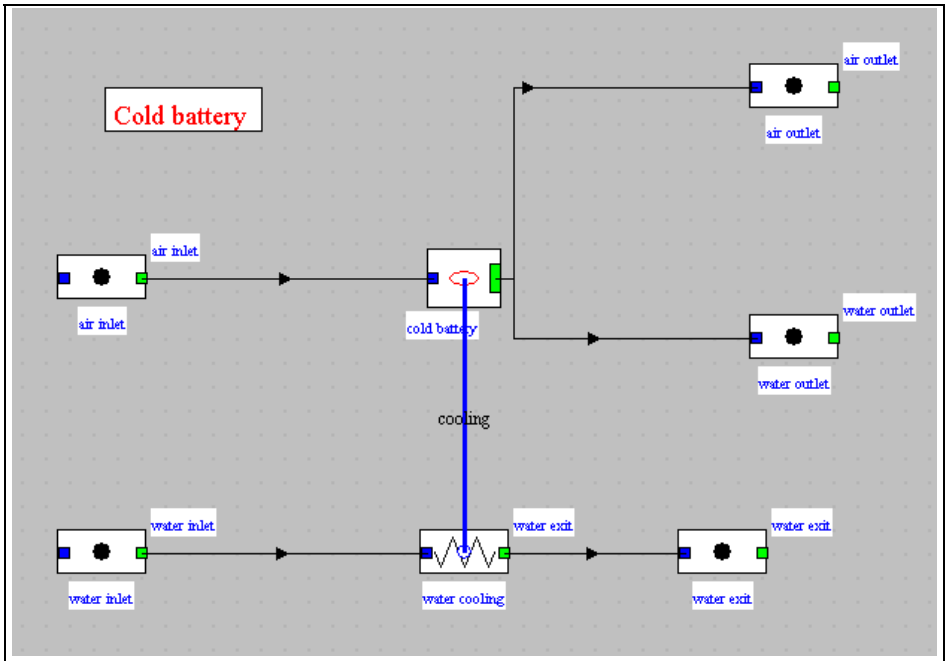

Figure 1: Schéma du composant batterie de refroidissement avec condensation

The model structure is given Figure 1. Here we have chosen in Thermoptim Global settings screen, the optional display in specific units for moist gas. For air, the flow shown is that of dry gas, which is invariant, and enthalpy h is replaced by the specific enthalpy q'. For water, which is not a moist gas, nothing is changed compared to the usual display.

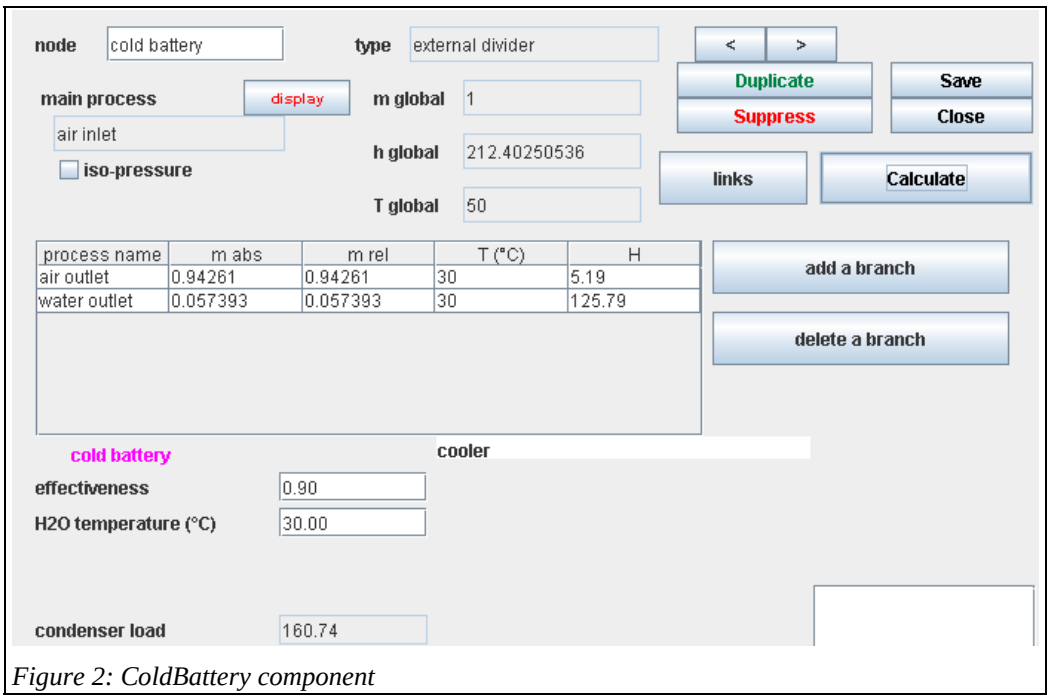

Figure 2: ColdBattery component

In the ColdBattery model, we assume that ε fractH2O is extracted and we recalculate the new composition. Given the assumptions, we perform a simple calculation on the overall compositions without having to determine the properties of moist mixtures put in. There is a more accurate model, called DehumidifyingCoil.

For example, consider a flow of 1 kg/s of moist air at 50 ° C and 1 bar, relative humidity equal to 0.8. We try to condense the excess water by cooling the air stream with water between 4.5 and 10 ° C, water and outlet gas temperatures being assumed to be 30 ° C.

The setting of the component is given in figure 2 assuming an effectiveness of 0.9. The thermal load calculated by the thermocoupler is equal to 161 kilowatts (Figure 3).

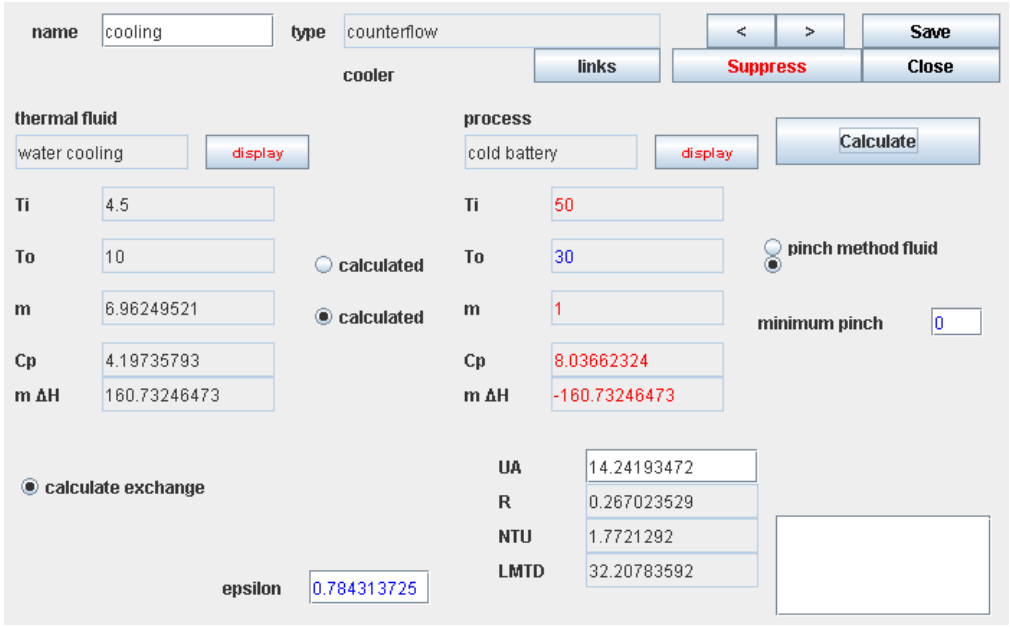

Figure 3: Thermocoupler

The thermocoupler setting is shown in Figure 3. Since the inlet and outlet cooling water temperatures are known, check the calculated flow-rate.

component name	molar fraction	mass fraction
Ar	0.008111861	0.01162456
O2	0.1892768	0.2172553
N2	0.7039293	0.7073498
H2O	0.09868213	0.06377036

Figure 4: Composition of the inlet moist air

The compositions of the air inlet and outlet are given in Figure 4 and 5.

component name	molar fraction	mass fraction
Ar	0.00890253	0.01233236
O2	0.2077257	0.2304835
N2	0.7725417	0.7504188
H2O	0.01083007	0.006765321

Figure : Composition of the outlet moist air

Study of the external class ColdBattery

Consistency tests are performed by the node method checkConsistency() to check that the fluids are well connected: in this case, moist air at the inlet and humid air and water at the outlet.

Please refer to Volume 3 of the reference manual for explanations on this point, valid for all external nodes.

The study of the external class ColdBattery allows one to understand how the model has been implemented. As can be seen, three steps are enough to make the calculations:

1) we first calculate the composition of the outlet gas, given the amount of water removed:

```
//recalcul des fractions molaires pour tenir compte de l'extraction de H2O
//recalculation of molar fractions to take into account H2O extraction
for(int i=0;i<inletWetGasFractmol.length;i++){//fractions molaire de sortie / outlet molar fractions
    double x_out=inletWetGasFractmol[i]/(1-sigma*fractH2O);
    Util.updateMolarComp(dryGasComp,inletWetGasComp[i], x_out);
}
double xH2O_out=(1-sigma)*fractH2O/(1-sigma*fractH2O);//fraction molaire de sortie en H2O /H2O outlet molar fraction
Util.updateMolarComp(dryGasComp,"H2O", xH2O_out);

dryGasSubstance.updateGasComposition(dryGasComp);
```

2) we then calculate the thermocoupler heat load, equal to the difference between the total enthalpy at the inlet (computed in checkConsistency ()), and the sum of enthalpies outflows, calculated in calculateProcess ())

```
dryGasSubstance.CalcPropCorps(273.15,1, 1); //calcul de l'état de référence / reference state calculation
getSubstProperties(dryGasSubstanceName);
double h0subst=Hsubst;

dryGasSubstance.CalcPropCorps(TH2O,20, 0);
getSubstProperties(dryGasSubstanceName);

outlet_w=xH2O_out*18.01528/M_secpoint/(1-xH2O_out);//(M_H2O)(x_H2O)/(M_gs)/(x_gs)
double L0_eau=2501.593;

//Enthalpie du gaz humide en sortie / moist gas outlet enthalpy
double Qprime_outlet=((Hsubst-h0subst)+outlet_w*L0_eau/(1+outlet_w))*(wetGasFlow-H2Oflow);

Object obj=Corps.createSubstance(resInt_fr.getString("eau"));//instanciation du corps, encapsulé dans un Object
                                                              //substance instanciation, wrapped in an Object
Corps lH2O=(Corps)obj;//transtyping the Object
lH2O.CalcPropCorps(TH2O,20, 0);
getSubstProperties(resInt_fr.getString("eau"));

//Enthalpie de l'eau en sortie / water outlet enthalpy
double Heau=Hsubst*H2Oflow;

//charge du condenseur / condenser load
double condLoad=Qprime_inlet-Heau-Qprime_outlet;
```

3) the node is updated using the generic methods described in the reference manual

```
    //affichages à l'écran / screen display
    Compr_value.setText(Util.aff_d(0,2));
    Q_value.setText(Util.aff_d(0,2));
    condenserLoad_value.setText(Util.aff_d(condLoad,2));
    sigma_value.setText(Util.aff_d(sigma,2));

    //mise à jour du thermocoupleur, comme pinchFluid, DTmin=10
    //thermocoupler update, as pinchFluid, DTmin=10
    updateThermoCoupler("cooler", Tinlet, TH2O, condLoad, wetGasFlow,true,10);

    //mise à jour du noeud en utilisant les méthodes génériques
    //node update using generic methods
    vTransfo= new Vector[nBranches+1];
    vPoints= new Vector[nBranches+1];
    setupVector(H2OProcess, H2OPoint, 0, H2Oflow, TH2O, Pamont, 1);
    setupVector(dryGasProcess, dryGasPoint, 1, wetGasFlow-H2Oflow, TH2O, Pamont, 0);
    setupVector(wetGasProcess, wetGasPoint, 2, wetGasFlow, Tinlet, Pamont, Xinlet);
    updateDivider(vTransfo,vPoints,Tinlet,Qprime_inlet);

    de.updateProcess(setEnergyTypes(wetGasProcess,0,0,0));
}
```