Modeling a single effect evaporator in Thermoptim

We have developed an external class in which the product to concentrate is modeled taking account a boiling point elevation proportional to the concentration, the other thermodynamic properties being those of water (Class EauSolute). If necessary, this class could easily be modified to reflect a more accurate model of the product.

An **evaporator** acts as a divider receiving in input the product to concentrate, and from which exit two fluids: steam (water vapor) and the concentrated product. The heat is provided by one or two fluids, thermal couplings being represented by thermocouplers. The reason we introduced two thermocouplers instead of one that would seem sufficient at first, is that in multiple-effect cycles, we can recover some of the necessary input to a downstream effect by condensing the steam from an upstream effect. The second thermocoupler allows you to provide the additional missing load from some other heat source.

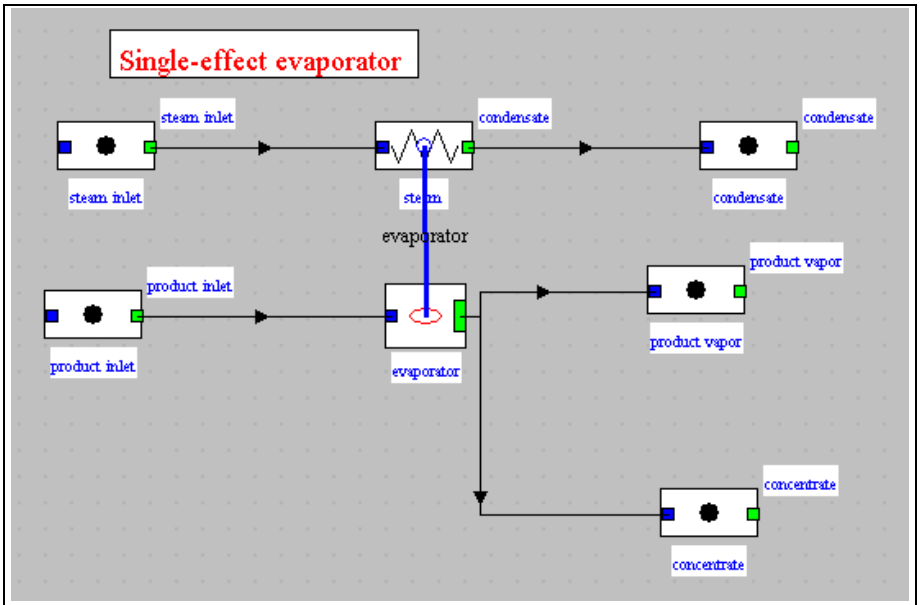

Figure 1: Component diagram

The structure of the evaporator model is given in Figure 1.

Physical Model

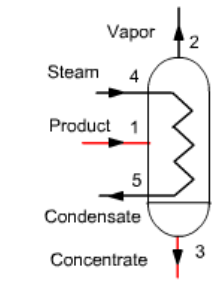

Figure 2

In a conventional single-effect evaporation cycle (Figure 2) a product to concentrate (solute + solvent) is injected into a unit, heated by any heat input Q (steam 4-5). The concentrated product is extracted in 3, at the bottom of the unit, while the solvent vapor exits in 2 and is condensed, its enthalpy being lost.

Calling x the mass concentration of solute, equations governing the behavior of this unit are:

conservation of the total flow: $\dot{m}_1 = \dot{m}_2 + \dot{m}_3$ (1)

conservation of solute: $x_1 \dot{m}_1 = x_3 \dot{m}_3$ (2)

conservation of enthalpy: $h_1 \dot{m}_1 + Q = h_3 \dot{m}_3 + h_2 \dot{m}_2$ (3)

The model that can be retained is:

1) the only parameter modified is the percentage of heat loss (determined by reference to the thermocoupler load), concentrations and pressures of the product in and out of the component being determined by their values in the point screens;
2) we consider for this example only one thermocoupler, the latter being not necessary.

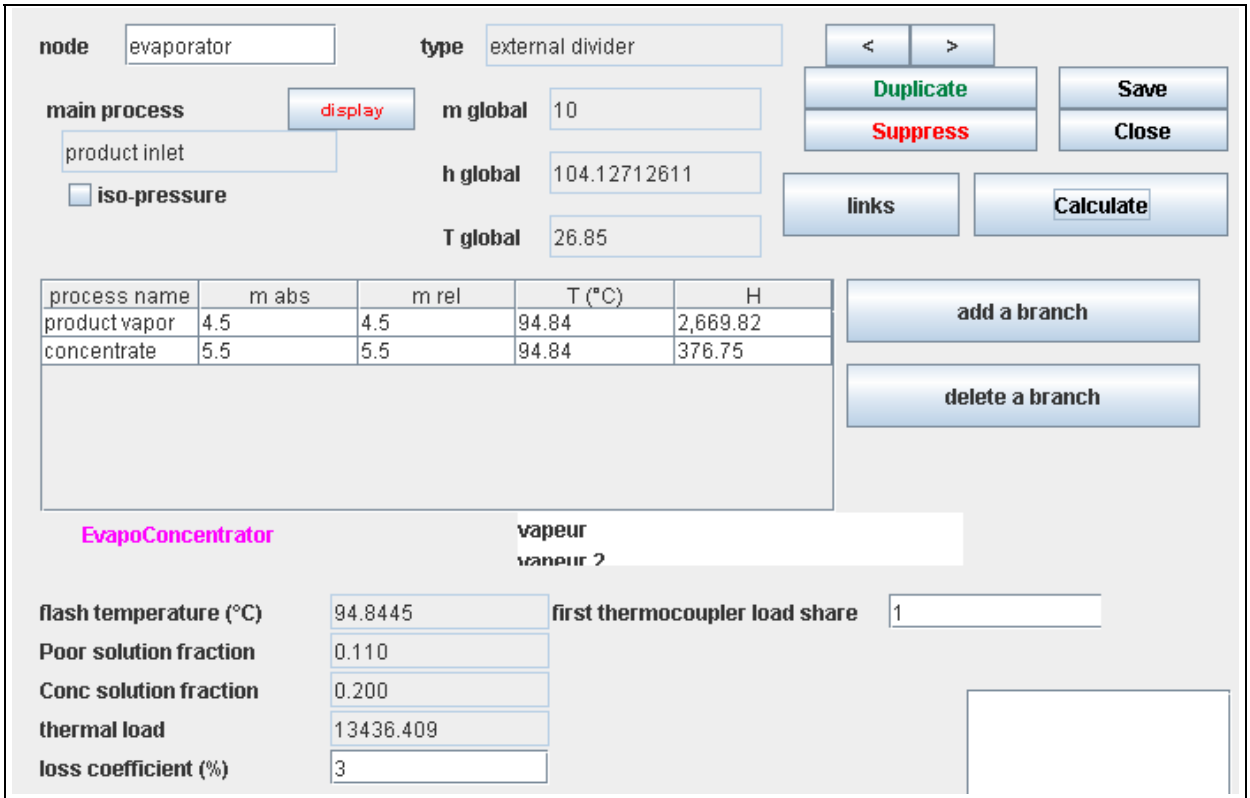

Figure 3: Component screen

The component screen is given in Figure 3, the product being concentrated from 11 to 20%, heat loss being 3% of the thermal load.

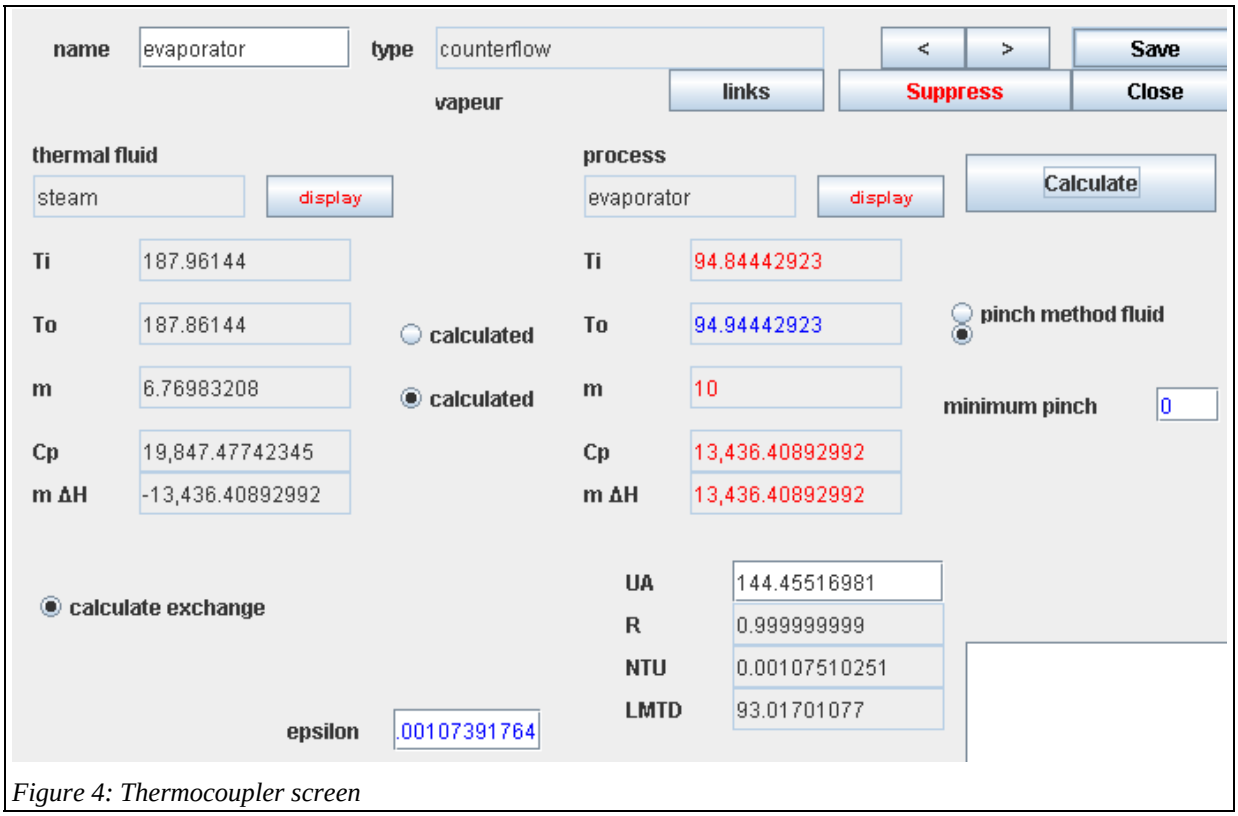

Figure 4: Thermocoupler screen

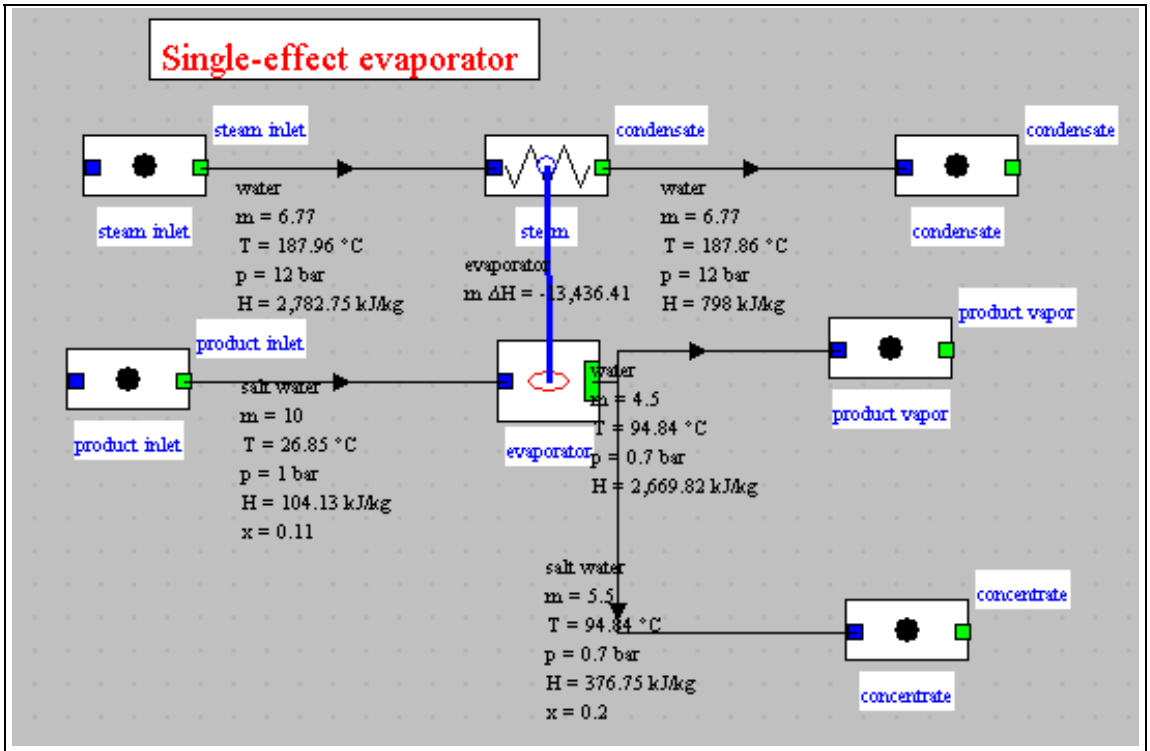

Figure 5: Component synoptic view

Study of the external class EvapoConcentrator

Consistency tests on the construction of the external divider are made by the method checkConsistency() to check the fluids are well connected: in this case, a product in input and the product and water in output. The tests implemented here are very basic and could be improved. 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 EvapoConcentrator shows how the model has been implemented. Two steps are enough to make the calculations:

1) we first determine the exit temperature of fluids (concentrated product and steam) and their mass flow rates and then computes the thermal power to provide Qgen

```
Tgen=produit.getSatTemperature(Pconc,Xconc);
Tgen_value.setText(Util.aff_d(Tgen-273.15,4));
produit.CalcPropCorps(Tgen,Pconc,Xconc) ;
getSubstProperties(nomCorps);
Hconc=Hsubst;

mconc=mpoor*Xpoor/Xconc;
mbuees=mpoor-mconc;

Tbuees=Tgen;
//ce calcul n'est pas vraiment nécessaire avec l'hypothèse que le produit a la même enthalpie que l'eau
//this calculation is not mandatory with the assumption that the product enthalpy is the same as that of water
Object obj=Corps.createSubstance(resIntTh_fr.getString("eau"));
Corps eau=(Corps)obj;
eau.CalcPropCorps(Tbuees,Pconc,1);
Hbuees=eau.getState()[5];

double loss=Util.lit_d(loss_value.getText());

Qgen=(mconc*Hconc+mbuees*Hbuees-mpoor*Hpoor)*(1+loss/100.);
```

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

```
vTransfo= new Vector[nBranches+1];
vPoints= new Vector[nBranches+1];
setupPoorSolution(mpoor,Tpoor,Ppoor,Xpoor);
setupConcSolution(mconc,Tgen,Pconc,Xconc);
setupBuees(mbuees,Tbuees,Pconc,1);
updateDivider(vTransfo,vPoints,Tpoor,Hpoor);
double share1=Util.lit_d(thermoCoupler1_value.getText());
updateThermoCoupler("vapeur", Tgen, Tgen, Qgen*share1, mpoor);
updateThermoCoupler("vapeur 2", Tgen, Tgen, Qgen*(1-share1), mpoor);
getUA("vapeur");//Imprime la valeur de UA du thermocoupleur / prints the thermocoupler UA value
getUA("vapeur 2");
Qgen_value.setText(Util.aff_d(Qgen,3));
Xpoor_value.setText(Util.aff_d(Xpoor,3));
Xconc_value.setText(Util.aff_d(Xconc,3));
de.updateProcess(setEnergyTypes(poorSolutionProcess,0,Qgen,0));
```