Modélisation dans Thermoptim d'un évapo-concentrateur à simple effet

Nous avons développé une classe externe dans laquelle le **produit à concentrer** est modélisé par un **retard à l'ébullition** proportionnel à la concentration, les autres propriétés thermodynamiques étant celles de l'eau (classe EauSolute). Si nécessaire, cette classe pourrait facilement être modifiée pour prendre en compte un modèle plus précis du produit.

Un **évapoconcentrateur** se comporte comme un diviseur recevant en entrée le produit à concentrer, et d'où sortent deux fluides, des buées (de la vapeur d'eau) et le produit concentré. L'apport de chaleur est assuré par un ou deux fluides, les couplages thermiques étant représentés par des thermocoupleurs. La raison pour laquelle nous avons introduit deux thermocoupleurs au lieu d'un seul qui semblerait suffire à première vue, est que, dans les cycles à effet multiple, on peut récupérer une partie de l'apport nécessaire à un effet aval en condensant les buées provenant d'un effet amont. Le deuxième thermocoupleur permet ainsi d'apporter le complément manquant à partir d'une autre source de chaleur.

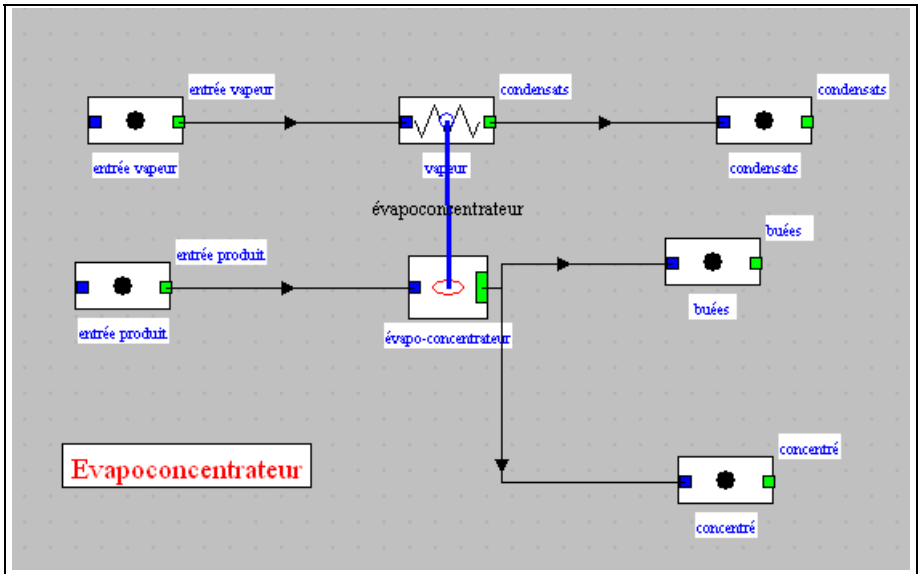

Figure 1 : Schéma du composant

La structure du modèle d'évapoconcentrateur est donnée figure 1.

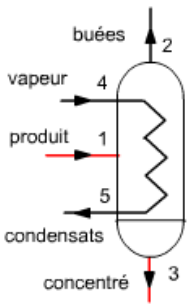

Figure 2

Modèle physique

Dans un cycle évaporatif à simple effet (figure 2), on injecte en 1 le produit à concentrer (soluté + solvant) dans une unité, chauffée par un apport de chaleur quelconque Q (vapeur 4-5). En 3, en bas de l'unité, est extrait le produit concentré, tandis que la vapeur de solvant (buées) sort en 2 et est condensée, son enthalpie étant perdue.

En appelant x la concentration massique en soluté, les équations qui régissent le comportement de cette unité sont les suivantes :

conservation du débit total : $\dot{m}_1 = \dot{m}_2 + \dot{m}_3$ (1)

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

conservation de l'enthalpie : $h_1 \dot{m}_1 + Q = h_3 \dot{m}_3 + h_2 \dot{m}_2$ (3)

Le modèle que l'on peut retenir est le suivant :

1) le seul paramètre modifiable est le pourcentage de pertes thermiques (déterminé par rapport à la charge des thermocoupleurs), les concentrations et les pressions du produit en entrée et en sortie du composant étant fixées par leurs valeurs dans l'écran des points ;
2) on ne considère pour cet exemple qu'un seul thermocoupleur, le second n'étant pas nécessaire.

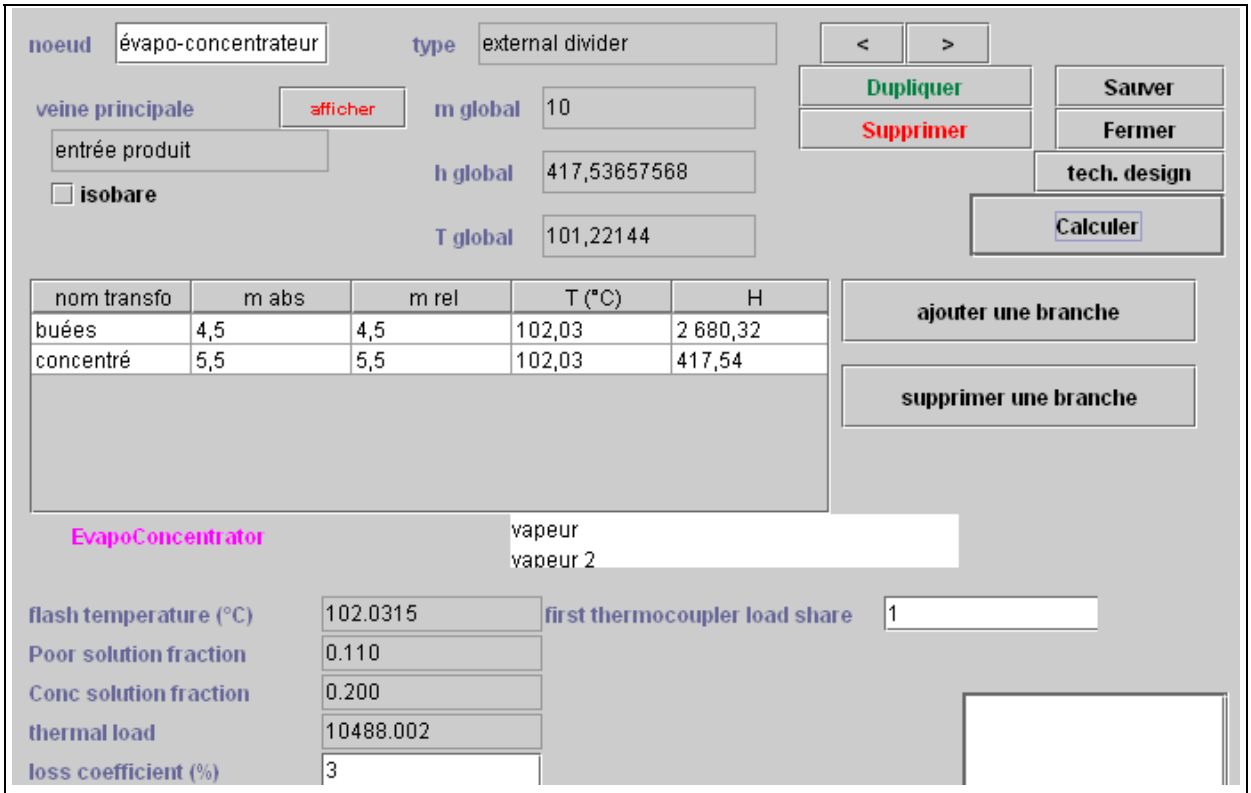

Figure 3 : Ecran du composant

L'écran du composant est donné figure 3, le produit étant concentré de 11 à 20 %, avec 3 % de pertes thermiques.

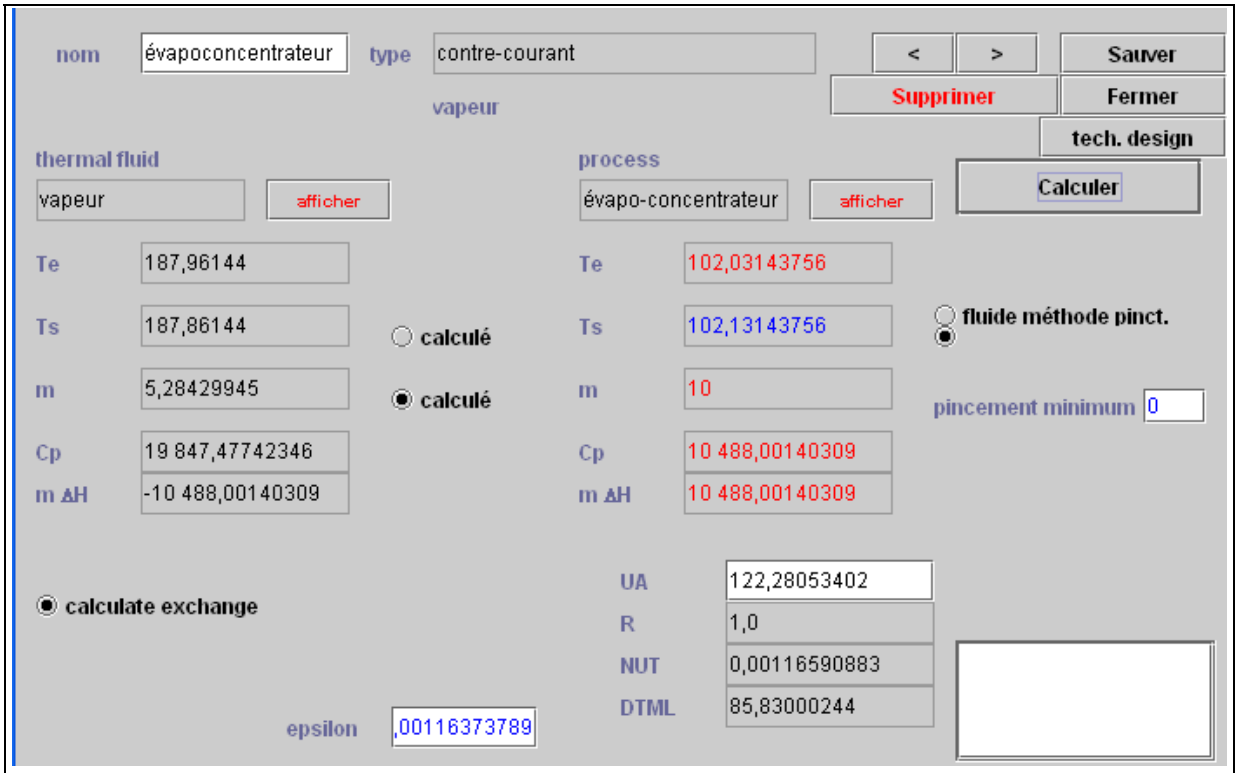

Figure 4 : Ecran du thermocoupleur

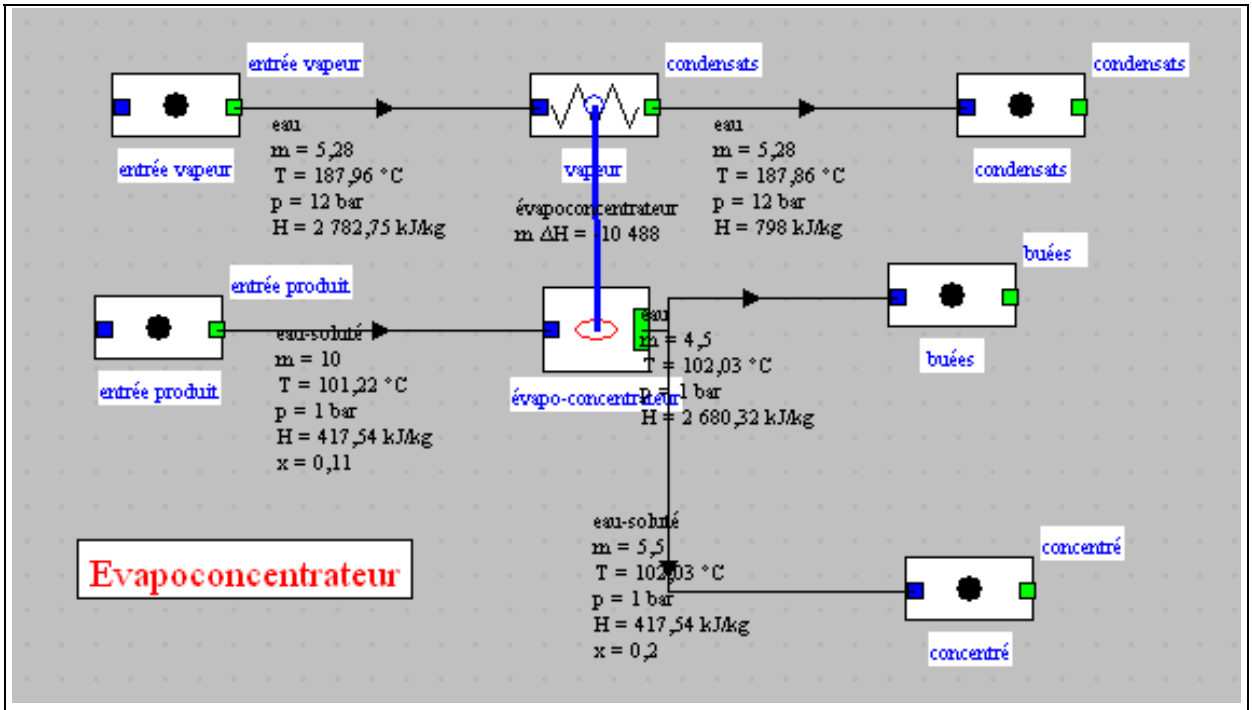

Figure 5 : Synoptique du composant

Etude de la classe externe EvapoConcentrator

Des tests de cohérence sur la construction du diviseur externe sont effectués par la méthode checkConsistency() pour vérifier que les fluides connectés sont les bons : dans ce cas, un produit en entrée et le produit et de l'eau et en sortie. Les tests ici implémentés sont très sommaires et pourraient être améliorés. On se reportera au tome 3 du manuel de référence pour les explications sur ce point, valables pour tous les nœuds externes.

L'étude de la classe externe EvapoConcentrator permet de voir comment le modèle a été implémenté. Deux étapes suffisent pour effectuer les calculs :

1) on commence par déterminer la température de sortie des fluides (produit concentré et buées) ainsi que leurs débits-masses, puis on calcule la puissance thermique Qgen à fournir

```
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
Object obj=Corps.createSubstance("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) le nœud est ensuite mis à jour en utilisant les méthodes génériques décrites dans le manuel de référence

```
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");
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));
```