## *Modèle du composant ColdBattery*

Le diviseur externe "cold battery" permet de modéliser le refroidissement et la condensation de l'eau contenue dans des gaz humides. Il s'agit d'un modèle simplifié, qui comporte deux paramètres, la température de l'eau, et l'efficacité d'extraction de l'eau ε, qui représente le pourcentage (en volume) de l'eau condensée par rapport à l'eau entrante.

### Modélisation d'un batterie de refroidissement avec condensation dans Thermoptim

Une batterie de refroidissement avec condensation se comporte comme un diviseur recevant de l'air humide en entrée, et dont sortent deux fluides : de l'eau et de l'air plus sec. Le refroidissement est assuré par un fluide, le couplage thermique pouvant être représenté par un thermocoupleur. Pour simplifier l'écriture, nous parlons d'air, mais ce composant peut refroidir et condenser n'importe quel mélange humide.

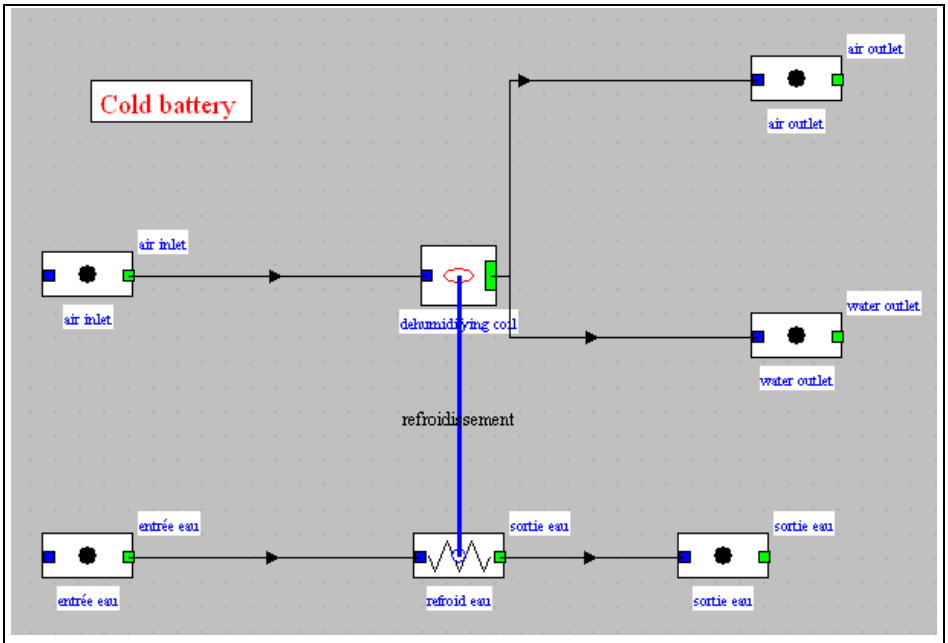


*Figure 1 : Schéma du composant batterie de refroidissement avec condensation*

La structure du modèle est donnée figure 1.

Dans le modèle ColdBattery, on suppose que ε fractH2O est extrait et on recalcule la nouvelle composition. Compte tenu des hypothèses faites, on effectue un simple calcul global portant sur les compositions, sans avoir à déterminer les propriétés des mélanges humides mis en jeu. Il existe un modèle plus précis, appelé DehumidifyingCoil.

A titre d'exemple, considérons un débit d'1 kg/s d'air humide à 50 °C et 1 bar, d'humidité relative égale à 0,8. On cherche à condenser l'eau en excès en refroidissant ce flux d'air par de l'eau entre 25 et 20 °C, la température de l'eau et du gaz en sortie étant supposée égale à 30 °C.

Le paramétrage du composant est donné figure 2 en supposant une efficacité de 0,9. La puissance thermique à évacuer par le thermocoupleur est calculée égale à 161 kW (figure 3).

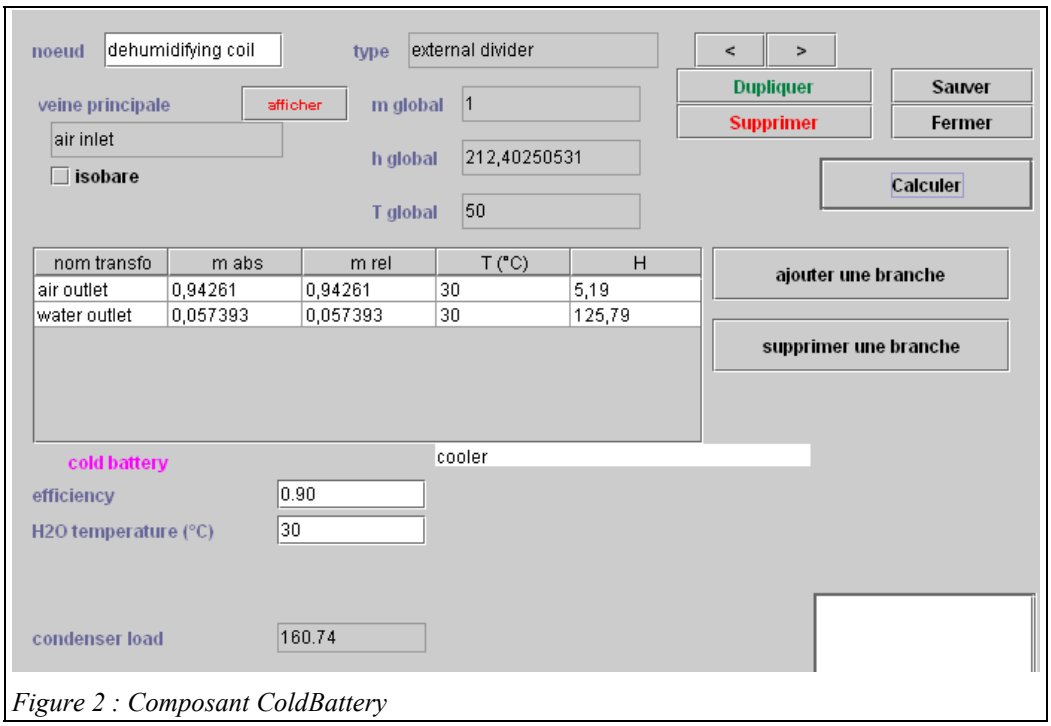


*Figure 2 : Composant ColdBattery*

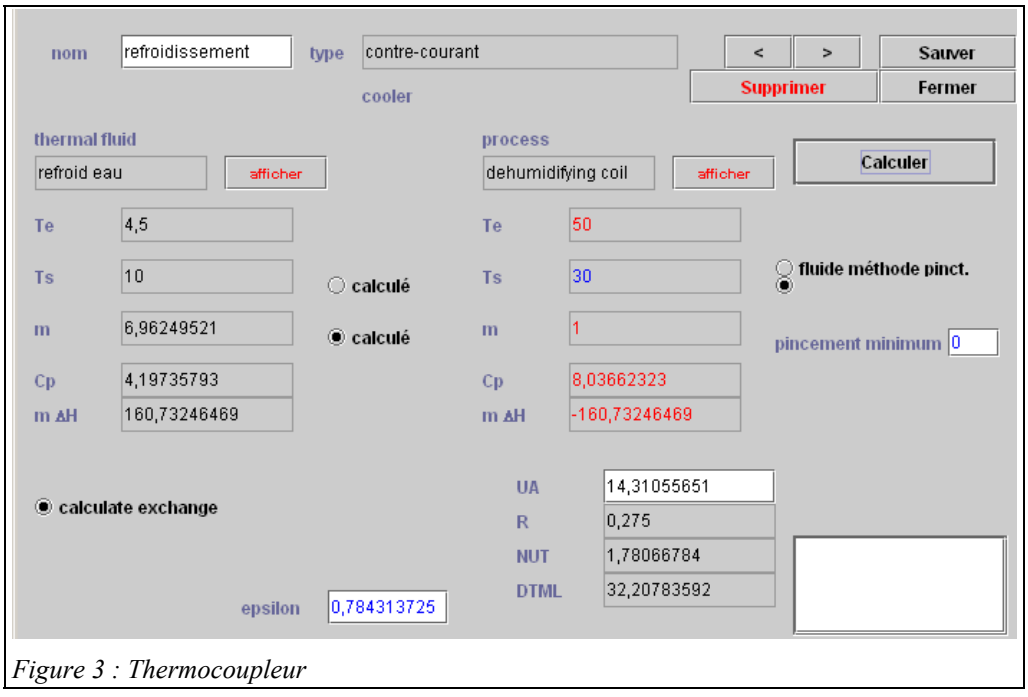


*Figure 3 : Thermocoupleur*

Le paramétrage du thermocoupleur est montré figure 3. Etant donné que les températures d'entrée et de sortie de l'eau de refroidissement sont connues, il faut choisir l'option qui permet de calculer le débit.

| nom du composant | fraction molaire | fraction massique |
|---|---|---|
| Ar | 0,008111861 | 0,01162456 |
| O2 | 0,1892768 | 0,2172553 |
| N2 | 0,7039293 | 0,7073498 |
| H2O | 0,09868213 | 0,06377036 |

*Figure 4 : Composition de l'air humide en entrée*

Les compositions de l'air en entrée et en sortie sont données figure 4 et 5.

| nom du composant | fraction molaire | fraction massique |
|---|---|---|
| Ar | 0,00890253 | 0,01233236 |
| O2 | 0,2077257 | 0,2304835 |
| N2 | 0,7725417 | 0,7504188 |
| H2O | 0,01083007 | 0,006765321 |

*Figure 5 : Composition de l'air humide en sortie*

## *Etude de la classe externe ColdBattery*

Des tests de cohérence du nœud sont effectués par la méthode checkConsistency() pour vérifier que les fluides connectés sont les bons : dans ce cas, de l'air humide en entrée et de l'air humide et de l'eau en sortie.

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 ColdBattery permet de voir comment le modèle a été implémenté. Comme on peut le voir, six étapes suffisent pour effectuer les calculs :

1) on commence par calculer la composition du gaz sortant, compte tenu de la quantité d'eau enlevée :

```
//recalcul des fractions molaires pour tenir compte de l'extraction de H2O
for(int i=0;i<inletWetGasFractmol.length;i++){//mise à jour des fractions molaire de sortie
    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
Util.updateMolarComp(dryGasComp,"H2O", xH2O_out);

dryGasSubstance.updateGasComposition(dryGasComp);
```

2) on calcule ensuite la charge thermique du thermocoupleur, égale à la différence entre l'enthalpie totale en entrée (calculée dans checkConsistency()), et la somme des enthalpies des flux sortants, calculées dans calculateProcess())

```
lecorps.CalcPropCorps(273.15,Ppoint, 1); //calcul de l'état de référence
getSubstProperties(nomCorps);
double hOsubst=Hsubst;

lecorps.CalcPropCorps(Tpoint,Ppoint, 1); //recalcul de l'état de sortie (fournit Hsubst)
getSubstProperties(nomCorps);
inputWetGasComp=lecorps.getGasComposition();//composition du gaz à traiter
fractH2O=Util.molarComp(inputWetGasComp,"H2O");//fraction molaire de H2O

//on calcule w inlet par la formule de définition, pour éviter, compte tenu de la
//température élevée du gaz, d'avoir à estimer les conditions de saturation
//cf. p. 71-72 tome 1
inlet_w=fractH2O*18.01528/M_secpoint/(1-fractH2O);//(M_H2O)(x_H2O)/(M_gs)/(x_gs)
double LO_eau=2501.593;

//Enthalpie totale d'entrée
Qprime_inlet=((Hinlet-hOsubst)+inlet_w*LO_eau/(1+inlet_w))*wetGasFlow;
```

```
double xH2O_out=(1-sigma)*fractH2O/(1-sigma*fractH2O);//fraction molaire de sortie en H2O
Util.updateMolarComp(dryGasComp,"H2O", xH2O_out);

dryGasSubstance.updateGasComposition(dryGasComp);

double wetSynGasMolFlow=wetGasFlow/wetGasM;
double H2OMassFlowTotal=wetSynGasMolFlow*fractH2O*18.01528;
double H2Oflow=H2OMassFlowTotal*sigma;

dryGasSubstance.CalcPropCorps(273.15,1, 1); //calcul de l'état de référence
getSubstProperties(dryGasSubstanceName);
double hOsubst=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 LO_eau=2501.593;

//Enthalpie du gaz humide en sortie
double Qprime_outlet=((Hsubst-hOsubst)+outlet_w*LO_eau/(1+outlet_w))*(wetGasFlow-H2Oflow);

Object obj=Corps.createSubstance("eau");//instanciation du corps, encapsulé dans un Object
Corps lH2O=(Corps)obj;//transtypage de l'Object
lH2O.CalcPropCorps(TH2O,20, 0);
getSubstProperties("eau");
//Enthalpie de l'eau en sortie
double Heau=Hsubst*H2Oflow;

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

3) le nœud est mis à jour en utilisant les méthodes génériques décrites dans le manuel de référence

```
//affichages à l'écran
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 des thermocoupleurs, comme pinchFluid, DTmin=10
updateThermoCoupler("cooler", Tinlet, TH2O, condLoad, wetGasFlow,true,10);

//mise à jour du noeud en utilisant les méthodes génériques
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));
```