Modélisation dans Thermoptim de l’humidification d'un gaz par barbotage (sparging saturator)

Le barbotage d'un gaz (sparging saturator) est une forme d'humidification de gaz par lavage qui permet à la fois de le nettoyer d'impuretés du fait qu'il se refroidit et de le saturer d'eau.

Cette opération présente la particularité de mettre en jeu deux flux séparés : le gaz entrant et l'eau, qui échangent de la matière et de l'énergie par l'intermédiaire d'une interface. Elle se comporte donc comme un quadripôle recevant deux fluides en entrée, et dont en sortent deux autres.

Ceci pose une petite difficulté pour en réaliser un modèle dans Thermoptim, étant donné que les seuls composants disponibles sont soit des transfos, soit des nœuds. La solution est de former le quadripôle en associant un mélangeur en entrée et un diviseur en sortie, les deux étant reliés par une transfo-point qui joue un rôle purement passif.

Pour que le modèle soit bien cohérent, on synchronise les calculs effectués par les deux nœuds. Plus précisément le diviseur de sortie prend le contrôle du mélangeur, dont le rôle se limite à effectuer une mise à jour des variables de couplage associées aux flux d'entrée. La structure du modèle est donnée figure 1.

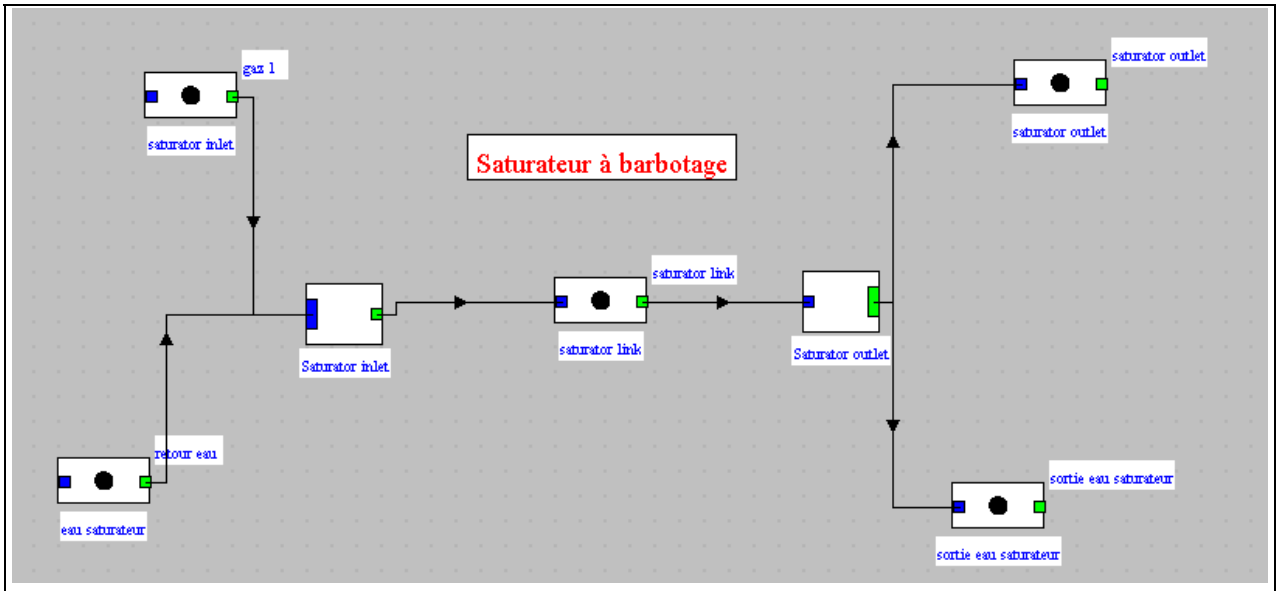

Figure 1 : Schéma du composant sparging saturator

Modèle d’humidification du gaz

Le modèle que nous développons ici détermine la température du gaz en établissant les bilans massiques et énergétiques, et en supposant que les flux sortants sont tous à cette même température. Il considère connue l’humidité absolue en sortie du composant, le seul paramètre étant le pourcentage de l’humidité saturante en sortie. Les débits des deux flux entrants sont imposés par les conditions en amont du composant et non pas recalculés. Si le débit d'eau est insuffisant pour que son échauffement (jusqu'à la température du gaz sortant) permette d'extraire du gaz l'enthalpie requise, un message en avertit l'utilisateur.

Les fonctions de calcul des propriétés humides des gaz et des points de Thermoptim ont été rendues accessibles depuis les classes externes. Nous vous conseillons de vous référer à la note : "Calculs des gaz humides depuis les classes externes" pour une présentation détaillée des méthodes disponibles.

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

1) le seul paramètre est le pourcentage de l’humidité saturante wsat en sortie de composant, lue à l'écran (*a priori* très proche de 1) ;
2) on commence par calculer l'humidité absolue du gaz entrant, et on détermine le débit-masse de gaz sec à partir de celui du gaz humide ;

3) l'humidité relative ε en sortie est imposée, et les propriétés humides du gaz de sortie sont calculées, ce qui fournit les enthalpies spécifique et totale à extraire du gaz
4) le débit d'eau fourni par le gaz est déterminé et la composition du gaz humide en sortie est modifiée.
5) le bilan enthalpique sur l'eau fournit sa température de sortie, qui doit être inférieure à celle du gaz sortant
6) On itère sur les étapes 3 à 5 jusquà ce que le bilan enthalpique de sortie soit égal à celui des flux entrants, ce qui donne la température de sortie.
7) les valeurs en aval du nœud sont mises à jour

Dans Thermoptim, le composant est comme on l'a dit représenté par un mélangeur externe connecté à un diviseur externe, les calculs étant effectués par ce dernier. Les classes s'appellent SpargingSaturatorInlet et SpargingSaturator. L'écran du composant est donné figure 2.

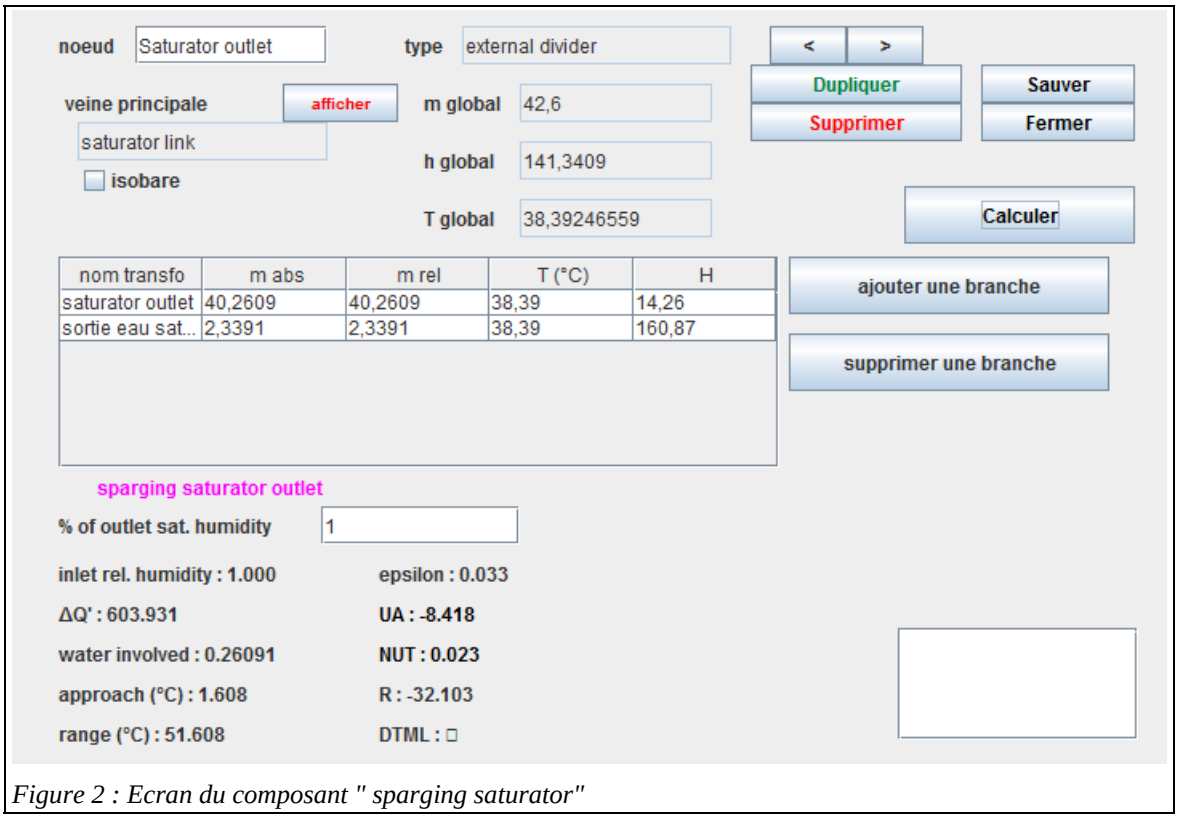

Figure 2 : Ecran du composant " sparging saturator"

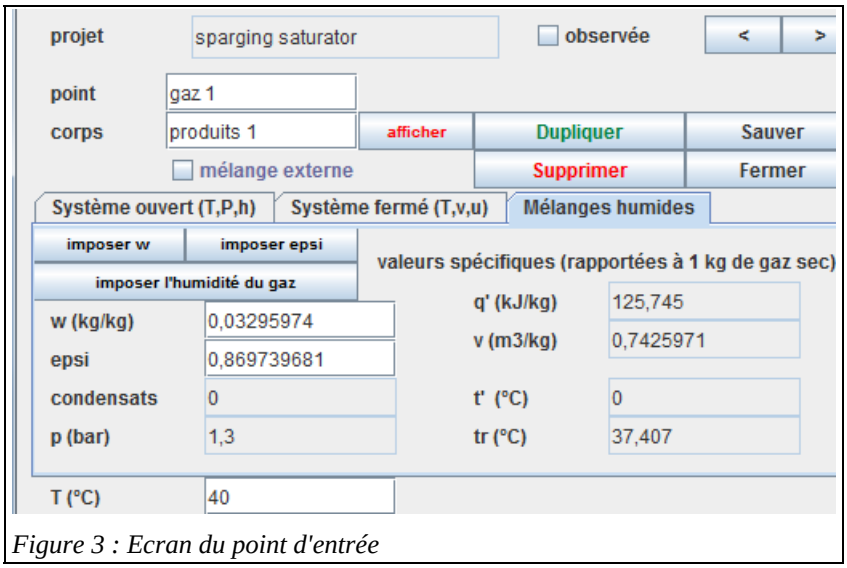

Figure 3 : Ecran du point d'entrée

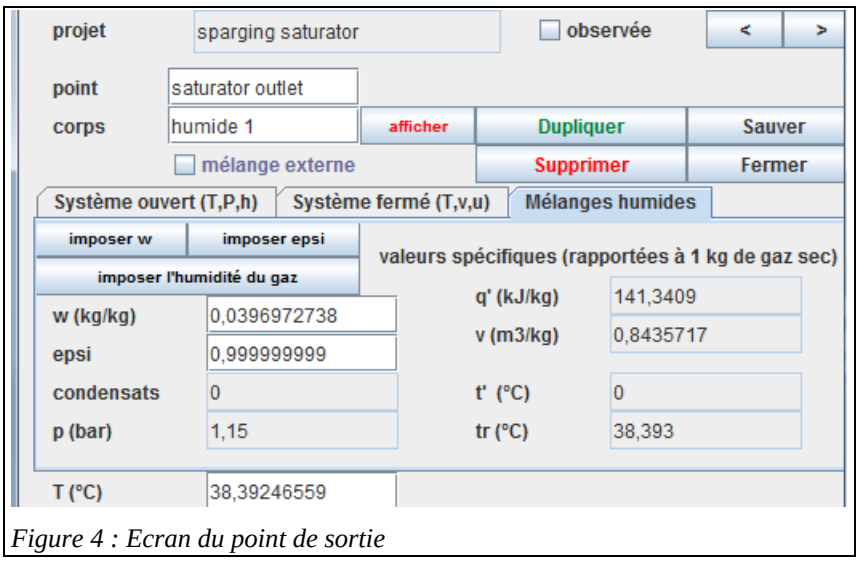

Figure 4 : Ecran du point de sortie

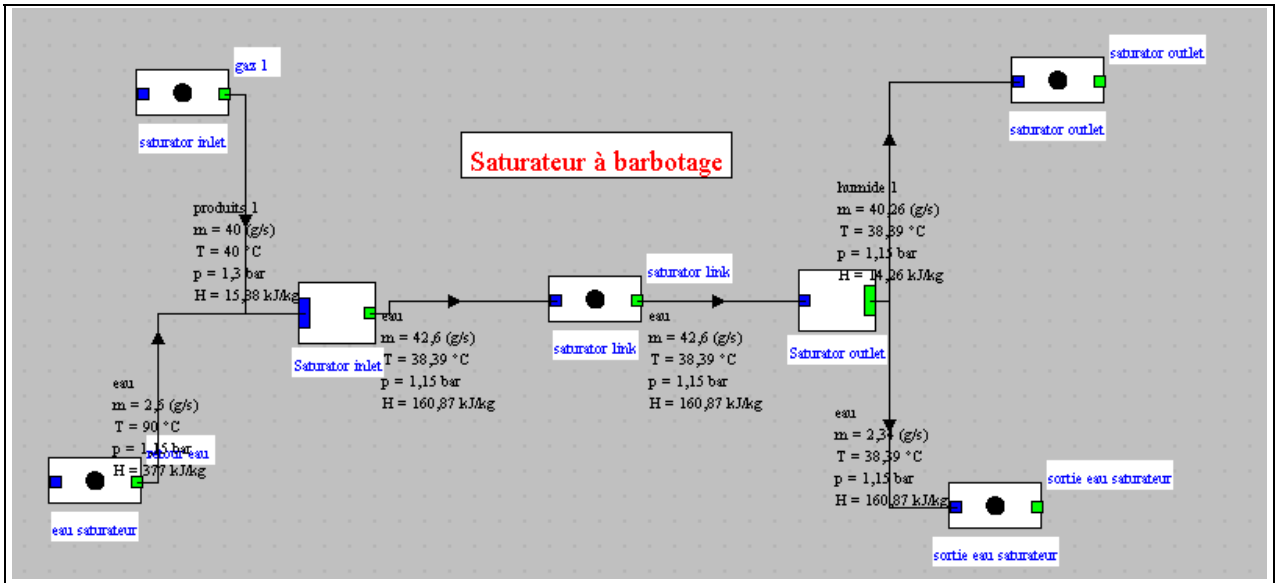

Figure 5 : Synoptique du composant " sparging saturator "

Etude de la classe externe SpargingSaturator

Pour assurer la cohérence du modèle (éviter que l'on connecte le mélangeur d'entrée à un diviseur de sortie inadéquat), chacun des deux nœuds essaie d'instancier l'autre en recherchant sa classe parmi les composants externes du projet, et vérifie que tous deux sont bien connectés à la même transfo de liaison. Si l'opération échoue, un message avertit l'utilisateur que la construction est incorrecte. Cette vérification est effectuée par les méthodes setupOutlet() et setupInlet().

De surcroît, des tests de cohérence de chaque nœud sont effectués par la méthode checkConsistency() pour vérifier que les fluides connectés sont les bons : dans ce cas, un gaz humide et de l'eau en entrée et 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 SpargingSaturator permet de voir comment le modèle a été implémenté. Six étapes suffisent pour effectuer les calculs :

1) on commence par calculer l'humidité absolue du gaz entrant, et on détermine le débit-masse de gaz sec à partir de celui du gaz humide puis on initialise l'enthalpie spécifique amont :

```
//Propriétés humides du gaz entrant
args[0]="process";//type of the element (see method getProperties(String[] args))
args[1]=ss.gasProcess;//name of the process (see method getProperties(String[] args))
Vector vProp=proj.getProperties(args);
String amont=(String)vProp.elementAt(2);//gets the downstream point name
getPointProperties(amont);//direct parsing of point property vector
getSubstProperties(nomCorps);
double M=Msubst;
Vector vComp=lecorps.getGasComposition();
drySynGasSubstance.updateGasComposition(vComp);//on met à jour la composition du gaz traité

//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
double fractH2O=Util.molarComp(vComp,"H2O");//fraction molaire de H2O
inlet_w=fractH2O*18.01528/M_secpoint/(1-fractH2O);//(M_H2O)(x_H2O)/(M_gs)/(x_gs)
hamont=ss.hamont/(1+inlet_w);//passage en unités spécifiques
getPointProperties(amont);
double inletT=Tpoint;

//Calcul de l'enthalpie spécifique du gaz sortant
updatepoint(amont, false, 0,  //T
        false, 0, false, 0,//P,x
        true, "setW and calculate q'", inlet_w);
getPointProperties(amont);

hamont=QPrimepoint;//enthalpie spécifique du gaz sortant

//imposition de w et calcul des propriétés humides
Double f=(Double)vProp.elementAt(3);
double flow=f.doubleValue();//débit massique de gaz humide
flow_as=flow/(1+inlet_w);//débit massique de gaz sec

inletEnthalpy=hamont*flow_as+ss.waterH*ss.waterFlow;
JLabel1.setText("inlet rel. humidity : "+Util.aff_d(1,3));
```

2) on calcule ensuite par itération les propriétés du gaz sortant, en lui imposant l'humidité et en égalisant l'enthalpie totale en sortie avec celle en entrée

```
//recherche de la solution entre 12 °C et la plus haute des températures d'entrée
double Tmax=inletT;
if(ss.Tpoint>Tmax)Tmax=ss.Tpoint;
double  T=Util.dicho_T(this, 0, Ppoint, "qprime2", 285, Tmax, 0.01);
```

```
if (fonc.equals("qprime2")){//point final hors zone saturée / final point outside saturated zone
    String[] args=new String[2];

    //Propriétés humides du gaz sortant
    args[0]="process";//type of the element (see method getProperties(String[] args))
    args[1]=gasProcess;//name of the process (see method getProperties(String[] args))
    Vector vProp=proj.getProperties(args);
    String aval=(String)vProp.elementAt(1);//gets the upstream point name
    getPointProperties(aval);
    outletT=Tpoint;
    double Psat=eau.getSatPressure(T, 0);
    double wsat=18.01528/M_secpoint*Psat/(Ppoint-Psat)*epsi;//calcul de w final = wsat*epsi
    if(debug)System.out.println("Psat out : "+Psat);
    if(debug)System.out.println("wsat out : "+wsat);
    double deltaW=wsat-inlet_w;
    eau.CalcPropCorps(T, Ppoint, 0);
    double[]state=eau.getState();
    getSubstProperties("eau");
    Heau=state[5];
    remainingWater=ss.waterFlow-flow_as*deltaW;
    if(debug)System.out.println("remainingWater : "+remainingWater+"  Heau : "+Heau);

    updatepoint(aval, true, T,  //T
        false, 0, false, 0,//P,x
        false, "", 0);
    updatepoint(aval, false, 0,  //T
        false, 0, false, 0,//P,x
        true, "setW and calculate q'", wsat);

    getPointProperties(aval);//propriétés humides / moist properties
    double diff=inletEnthalpy-QPrimepoint*flow_as-remainingWater*Heau;
    if(debug)System.out.println("T : "+T+"  diff : "+diff);
    DeltaQprime=flow_as*(QPrimepoint-hamont);//enthalpie totale acquise par le gaz
    outletFlow=flow_as*(1+Wpoint);

    return diff;
}
```

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

```
//mise à jour du noeud en utilisant les méthodes génériques
vTransfo= new Vector[nBranches+1];
vPoints= new Vector[nBranches+1];
setupVector(gasProcess, aval, 0, outletFlow, T, gasP, 0);
setupVector(waterProcess, waterPoint, 1, remainingWater, T, gasP, 0);
setupVector(mainProcess, ss.linkPoint, 2, remainingWater+outletFlow, T, gasP, 0);
updateDivider(vTransfo,vPoints,T,QPrimepoint);
```

4) on estime enfin les caractéristiques globales de l'échangeur

```
if (mCpc>mCpf){
    epsilon=Math.abs((Tfs-Tfe)/(Tfmin-Tce));
    if(epsilon>1)epsilon=Math.abs((Tfs-Tfe)/(Tfmax-Tce));
    R=mCpf/mCpc;
    NUT=NUT_epsi(epsilon,R) ;
    UA=mCpf*NUT;
}
else{
    epsilon=Math.abs((Tcs-Tce)/(Tfmin-Tce));
    if(epsilon>1)epsilon=Math.abs((Tcs-Tce)/(Tfmax-Tce));
    R=mCpc/mCpf;
    NUT=NUT_epsi(epsilon,R) ;
    UA=mCpc*NUT;
}
```