CALCULS DES GAZ HUMIDES DANS THERMOPTIM

INTRODUCTION

Depuis l'origine, Thermoptim dispose de fonctions de calcul des propriétés humides des gaz et des points, mais celles-ci ne sont généralement utilisées que pour effectuer des calculs particuliers, découplés des calculs de cycles thermodynamiques standard, comme par exemple du traitement de l'air ou de la climatisation (cf. manuel de référence tome 2). C'est d'ailleurs pour cela que les transfos humides ne disposent pas de pictogramme dans l'éditeur de schémas.

Par commodité de langage nous utiliserons le qualificatif de **standard** lorsque nous ferons référence à l'environnement de Thermoptim hors calculs humides.

L'environnement de calcul des cycles standard de Thermoptim et celui des calculs humides ne sont en effet pas directement compatibles, et ceci pour les deux raisons suivantes :

- suivant l'usage en matière de calculs humides, les valeurs de ces fonctions particulières sont généralement ramenées au gaz sec, dont la composition est invariante, alors que les calculs standard effectués dans Thermoptim le sont par rapport à la composition réelle des gaz. Les valeurs auxquelles elles conduisent sont ainsi appelées spécifiques, pour les distinguer des autres
- de plus, les températures et pressions de référence n'étant pas les mêmes dans les deux environnements, il est nécessaire de faire des conversions lorsque l'on passe de l'un à l'autre.

Pourtant, un certain nombre de cycles thermodynamiques mettent en jeu des variations de l'humidité des gaz, et il était regrettable de ne pouvoir les modéliser facilement dans Thermoptim. C'est pour cela que les fonctions de calcul des propriétés humides des gaz et des points de Thermoptim ont été rendues accessibles depuis les classes externes. Cette note explique comment les utiliser.

Nous supposerons dans ce qui suit que le lecteur est suffisamment familier à la fois de l'utilisation des classes externes et de ce type de calcul, afin de nous concentrer sur la manière de les utiliser. Les définitions, notations et équations sont celles du tome 1 du livre Systèmes Energétiques.

On peut représenter un gaz humide dans Thermoptim de deux manières équivalentes : soit directement comme un corps composé comprenant au moins deux constituants : H2O et un autre gaz, pur ou composé, soit comme un gaz sec dont on connaît l'humidité spécifique. La première manière présente l'avantage que la composition du gaz humide est accessible à tout moment. En revanche, elle suppose que, pour un même gaz sec, on crée un nouveau corps humide pour chaque valeur de l'humidité. La seconde représentation est quant à elle beaucoup plus concise, étant donné qu'elle ne fait appel qu'au gaz invariant et à la valeur de l'humidité spécifique. C'est pourquoi c'est cette dernière qui est utilisée par les fonctions de calcule des gaz humides, alors que c'est la première qui est de règle dans l'environnement de calcul standard des cycles de Thermoptim.

Rappelons que l'on appelle **humidité relative ε** le rapport de la pression partielle de la vapeur d'eau à sa pression de vapeur saturante, et que, par définition, l'indice gs correspondant aux valeurs relatives au gaz sec, **l'humidité absolue ou spécifique w** est égale au rapport de la masse d'eau contenue dans un volume donné de mélange humide à la masse de gaz sec contenue dans ce même volume, soit :

$$w = \frac{y_{H2O}}{y_{gs}} = \frac{M_{H2O}\, x_{H2O}}{M_{gs}\, x_{gs}} = \frac{18\, x_{H2O}}{M_{gs}\,(1 - x_{H2O})}$$

Cette relation permet de calculer w lorsqu'on connaît la composition du gaz humide, comme le montre notamment l'exemple donné ci-dessous.

METHODES DISPONIBLES DANS LES CLASSES EXTERNES

Généralement, les calculs de gaz humides sont effectués (à partir des classes externes) au niveau d'un point, mais une méthode permet de directement modifier l'humidité d'un gaz. L'humidité du gaz est entrée

en indiquant soit son humidité absolue w, soit son humidité relative ε. Les deux méthodes employées sont définies dans la classe ExtProcess, dont dérivent toutes les transfos et tous les nœuds externes.

La méthode **updatepoint("nomPoint",…)** permet d'effectuer les calculs humides tandis que la méthode **getPointProperties("nomPoint")** récupère, en sus des propriétés standard (P,T,h,s…) les valeurs des propriétés humides d'un point par les grandeurs suivantes : Wpoint pour l'humidité absolue w, Epsipoint pour l'humidité relative ε, Qprimepoint pour l'enthalpie spécifique q', Tprimepoint pour la température adiabatique t' (en °C), Trpoint pour la température de rosée tr (en °C), VPrimepoint pour le volume spécifique vs, Condpoint pour les condensats, et M_secpoint pour la masse molaire du gaz sec.

Recherche de l'humidité d'un point

Lorsque l'état du point a été sauvé, w est connu, et la méthode **getPointproperties("nomPoint")** permet d'y accéder directement.

Lorsque la composition du gaz est déterminée par programmation, il peut être nécessaire de recalculer w, ce qui peut être fait par la formule ci-dessous :

```
//calcul de l'humidité absolue du gaz
double inlet_w=18.01528*fractH2OFuel/fuelM/(1-fractH2OFuel);// w = M_H2O x_H2O / M_gs x_gs
```

Mise à jour des propriétés humides d'un point

La méthode **updatepoint("nomPoint",…)** est une méthode de portée générale de mise à jour des variables d'état d'un point et de recalcul, qui a été généralisée pour permettre les calculs humides. Son code est donné ci-dessous :

```
public void updatepoint(String name, boolean updateT, double T,
    boolean updateP, double P, boolean updateX, double x,
    boolean melHum, String task, double value){
            String[] args=new String[2];
     Vector vPoint=new Vector();
     vPoint.addElement(name);
     vPoint.addElement(Util.aff_b(updateT));
     vPoint.addElement(Util.aff_d(T));
     vPoint.addElement(Util.aff_b(updateP));
     vPoint.addElement(Util.aff_d(P));
     vPoint.addElement(Util.aff_b(updateX));
     vPoint.addElement(Util.aff_d(x));
     vPoint.addElement(Util.aff_b(melHum));
     vPoint.addElement(task);
     vPoint.addElement(Util.aff_d(value));
     proj.updatePoint(vPoint);

            }
```

Comme le montre ce code, elle construit un Vector puis passe le relais à la méthode **updatePoint()** de Projet.

Si le booléen melHum vaut "false", la mise à jour du point concerne T, P ou x, selon que les booléens updateT, updateP et updateX valent "true" ou "false" : il s'agit d'une mise à jour standard sans calcul des propriétés humides.

Si le booléen melHum vaut "true", seuls les calculs humides sont effectués, même si updateT, updateP et updateX valent "true".

Ces calculs sont définis par les deux grandeurs **task** et **value**.

task est une chaîne de caractère précisant le type de calculs à effectuer, et **value** un double fournissant la valeur de la grandeur à modifier.

1) Calculs sans modification de la composition du gaz

Les calculs étant effectués par rapport au gaz sec, la composition du gaz n'est pas modifiée.

Imposer l'humidité spécifique w

Si task ="setW and calculate all", Thermoptim impose w (passé dans value) et calcule toutes les propriétés humides

Lorsque la température d'un point est élevée, des messages avertissant l'utilisateur que la notion d'humidité relative perd son sens peuvent être affichés. Pour contourner cette difficulté, le paramétrage ci-dessous ne calcule que l'enthalpie spécifique

Si task ="setW and calculate q'", Thermoptim impose w (passé dans value) et calcule toutes les propriétés humides sauf t'

Si task ="calcWsat", Thermoptim calcule wsat et toutes les propriétés humides à la saturation sauf t'

Imposer l'humidité relative ε

Si task ="setEpsi", Thermoptim impose ε (passé dans value)

Si task ="setEpsi and calculate", Thermoptim impose ε (passé dans value) et calcule toutes les propriétés humides

2) Modification de la composition du gaz

2.1 en opérant indirectement à partir d'un point

Lorsqu'on désire modifier la composition d'un gaz, la méthode updatePoint() permet de le faire avec les paramétrages suivants :

Si task ="modHum", Thermoptim modifie la composition du gaz pour que son humidité soit égale à Wpoint (il n'y a alors pas besoin de passer de valeur dans value).

Si task ="setGasHum", Thermoptim modifie la composition du gaz pour que son humidité soit égale à w (passé dans value)

2.2 en opérant directement sur le gaz

Il est aussi possible de modifier l'humidité d'un gaz indépendamment de l'état d'un point, en utilisant la méthode **updateGasComp()** de GazIdeal, accessible par **public void updateGasComposition(Vector vComp)** de Corps : si le premier élément de vComp est un Integer d'une valeur négative, un traitement particulier est effectué. L'humidité absolue passée en troisième élément de vComp est imposée au gaz.

```
else{//modifications gaz humides
   String task=(String)vComp.elementAt(1);
   String value=(String)vComp.elementAt(2);
   if(task.equals("setGasHum")){//sets the gas humidity
      double w=Util.lit_d(value);
      setGasHum(w);
   }
}
```

L'exemple ci-dessous, issu de la classe BiomassCombustion, montre comment modifier la composition d'un gaz sec pour qu'elle corresponde au gaz humide dont la fraction molaire en eau est fractH2Ofuel :

```
//mise en forme du Vector
```

```
Vector vComp=new Vector();
 vComp.addElement(new Integer(-1));
 vComp.addElement("setGasHum");
 vComp.addElement(Util.aff_d(inlet_w));
//modification de la composition du gaz
 NewFuelSubstance.updateGasComposition(vComp);
```

EXEMPLE D'UTILISATION : MODELE DE SATURATEUR

Dans une turbine à gaz à air humide, on augmente la puissance de la machine et l'efficacité du cycle en humidifiant dans un saturateur l'air comprimé avant entrée dans la chambre de combustion. Le cycle est assez complexe pour optimiser la récupération de chaleur, mais le modèle de saturateur se présente de manière relativement simple (figure 1) : de l'eau à une température de l'ordre de 280 °C entre dans le saturateur, où elle est mise en contact avec un flux d'air chaud (200 °C)et relativement sec sortant du compresseur. Une partie de l'eau est vaporisée et sert à augmenter l'humidité de l'air, qui sort proche de l'état saturé. Le reliquat d'eau est recyclé. On suppose ici que le saturateur est adiabatique et que l'eau et l'air sortent à la même température.

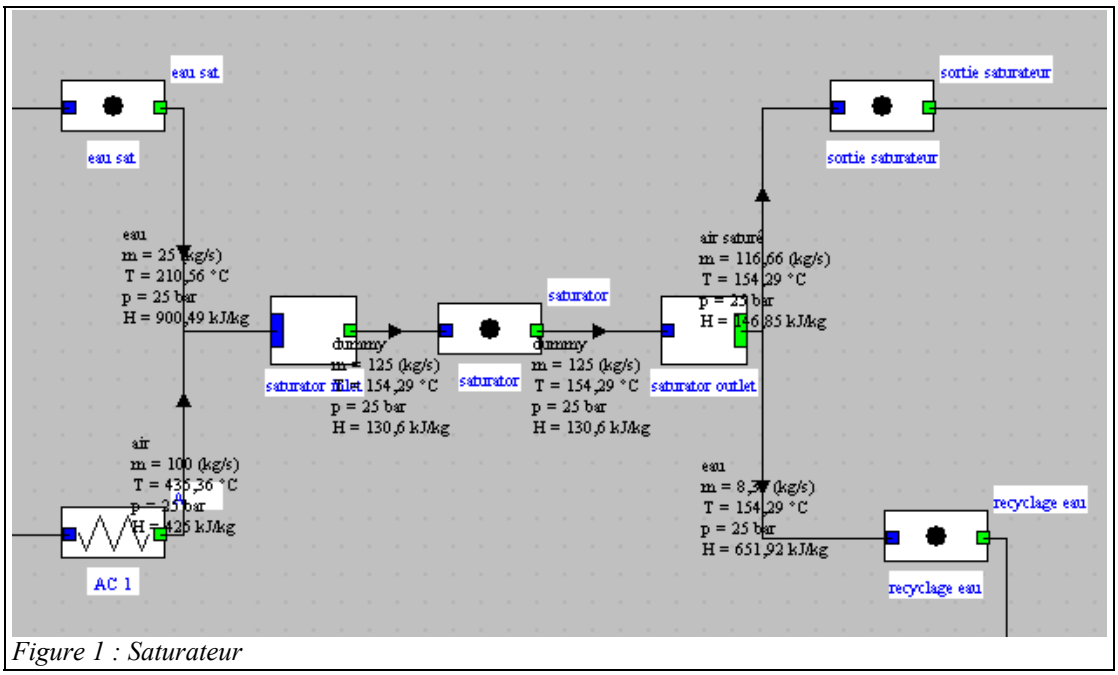

Figure 1 : Saturateur

Le code de la classe seprésente comme suit :

1) on commence par chercher la composition du gaz entrant, et on met à jour la composition du gaz en sortie, précaution au cas où les deux gaz secs seraient différents

```
//Propriétés humides du gaz entrant
args[0]="process";//type of the element (see method getProperties(String[] args))
args[1]=wq.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();
outletGasSubstance.updateGasComposition(vComp);//on met à jour la composition du gaz en sortie
//cette ligne de code permet d'utiliser le saturateur avec n'importe quel gaz
```

2) on calcule l'humidité absolue en entrée par la formule de définition, pour éviter, compte tenu de la température élevée du gaz, d'avoir à estimer les conditions de saturation

```
//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
double inlet_w=fractH2O*18.01528/M_secpoint/(1-fractH2O);//(M_H2O)(x_H2O)/(M_gs)/(x_gs)
getPointProperties(amont);
double inletT=Tpoint;
```

3) on détermine le débit de gaz sec et l'enthalpie spécifique du gaz en entrée, en utilisant les deux méthodes updatepoint() et getPointProperties()

```
//estimation du débit d'air sec
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

updatepoint(wq.gasPoint, false, 0,  //T
    false, 0, false, 0,//P,x
    true, "setW and calculate q'", inlet_w);
getPointProperties(wq.gasPoint);
qPrimeAmont=QPrimepoint;//enthalpie spécifique de l'air entrant dans le saturateur
```

4) A ce stade, les conditions humides du gaz amont sont parfaitement calculées. Il faut maintenant déterminer la température de sortie du saturateur Ts, en résolvant simultanément :

- le bilan hydrique (le débit d'eau consommé est égal au produit du débit de gaz sec par la variation d'humidité du gaz)
- le bilan enthalpique (la somme des débits denthalpie entrants (en unités spécifiques pour le gaz humide) est égal à la somme des débits d'enthapie sortants.

Etant donné que Ts est inconnue, on fait une recherche de solution par dichotomie, en utilisant la fonction générique Util.dicho_T(), qui fait appel à f_dicho(). Le code est donné ci-dessous :

- on fait passer l'humidité d'entrée en argument, à la place de la pression, connue par ailleurs
- on commence par calculer l'enthalpie de l'eau en sortie heau(Ts)
- on modifie la température du gaz en sortie, puis son humidité, à partir de la valeur lue à l'écran, et on récupère les valeurs de son humidité absolue et de son enthalpie spécifique
- on calcule le débit d'eau restant (il faudrait pour bien faire effectuer un test pour vérifier qu'il reste positif)
- on écrit que l'enthalpie perdue par l'eau se retrouve dans l'air, et on calcule le résidu diff
- la température Ts est déterminée lorsque diff=0.

```
//       double T=Util.dicho_T(this, 0, inlet_w, "saturator", 373.15, 450., 0.01);
      if (fonc.equals("saturator")){
          double diff;
          double w_dicho=P;
          //enthalpie de l'eau en sortie
          updatepoint(waterPoint, true, T,  //T
              false, 0, false, 0,//P,x
              false, "", 0);
          getPointProperties(waterPoint);//état de l'eau en sortie
          double hEau=Hpoint;

          //enthalpie spécifique et humidité spécifique de l'air en sortie
          updatepoint(gasPoint, true, T,  //T
              false, 0, false, 0,//P,x
              false, "", 0);
          updatepoint(gasPoint, false, 0,  //T
              false, 0, false, 0,//P,x
              true, "setEpsi and calculate", Util.lit_d(outletEpsi_value.getText()));
          getPointProperties(gasPoint);//propriétés humides
          waterFlow=wq.waterFlow -(Wpoint-w_dicho)*flow_as;//débit d'eau restant
          //on écrit que l'enthalpie perdue par l'eau se retrouve dans l'air
          diff=wq.waterFlow*wq.waterH-hEau*waterFlow+flow_as*(qPrimeAmont-QPrimepoint);
          return diff;
      }
```

5) Ts étant déterminé, on modifie la composition du gaz humide en sortie

```
//modification de la composition du gaz
outletT=T;
updatepoint(gasPoint, false, 0,  //T
        false, 0, false, 0,//P,x
        true, "modHum", 0);
```

Si l'on se reporte aux valeurs qui apparaissent sur le synoptique de la figure 1, le bilan apparent du saturateur est le suivant :

	kW
débit enthalpique total d'entrée	65 012
débit enthalpique total de sortie	22 570
écart apparent	42 443
écart par kg/s d'eau consommée	2 547

Tout se passe comme si 42,4 MW thermiques avaient disparu, mais, comme le montre la dernière ligne du tableau, cette valeur correspond exactement à l'enthalpie de vaporisation de l'eau contenue dans le gaz humide, qui n'est pas comptabilisée dans la valeur affichée par Thermoptim compte tenu des conventions adoptées pour les gaz idéaux (enthalpie nulle à 25 °C et 1 bar).

Bien évidemment, si l'on refroidit suffisamment les gaz d'échappement pour que l'eau qu'ils contiennent se condense, cette enthalpie apparaîtra de nouveau (avec d'ailleurs celle provenant de l'eau formée dans la chambre de combustion).

Annexe: méthodes de Projet accessibles depuis les classes externes

La méthode updatePoint de Projet permet de modifier par programmation l'état d'un point et de le recalculer, y compris ses propriétés humides.

```
public void updatePoint(Vector properties){
        String nomPoint=(String)properties.elementAt(0);
        PointCorps point=getPoint(nomPoint);
        if(point!=null){
          String test=(String)properties.elementAt(1);
          boolean updateT=Util.lit_b(test);
          String value=(String)properties.elementAt(2);
          double T=Util.lit_d(value);
          test=(String)properties.elementAt(3);
          boolean updateP=Util.lit_b(test);
          value=(String)properties.elementAt(4);
          double P=Util.lit_d(value);
          test=(String)properties.elementAt(5);
          boolean updateX=Util.lit_b(test);
          value=(String)properties.elementAt(6);
          double x=Util.lit_d(value);

          //pour mélanges humides
          if(properties.size()>7){
             test=(String)properties.elementAt(7);
             boolean melHum=Util.lit_b(test);

             if(!melHum){//calculs à effectuer dans le cas général
                if(updateT)point.setT(T);
                if(updateP)point.setP(P);
                if(updateX)point.setX(x);
                point.CalculeUnPoint();
             }
             else{//calculs humides

                String task=(String)properties.elementAt(8);
                value=(String)properties.elementAt(9);

                if(task.equals("setW and calculate all")){//sets w and calculates moist properties
                   double w=Util.lit_d(value);
                   point.setW(w);
                   point.calcHum();
                }
                if(task.equals("setW and calculate q'")){//sets w and calculates moist properties except t'
                   double w=Util.lit_d(value);
                   point.setW(w);
                   point.calcQprime();
                }
               if(task.equals("setEpsi")){//sets epsilon
                   double epsi=Util.lit_d(value);
                   point.setEpsi(epsi);
                   point.impHumRel();
                }
                if(task.equals("setEpsi and calculate")){//sets epsilon and calculates moist properties
                   double epsi=Util.lit_d(value);
                   point.setEpsi(epsi);
                   point.impHumRel();
                   point.calcQprime();
                }
                if(task.equals("calcWsat")){//calculates saturation properties and moist properties except t'
                   T=Util.lit_d(value);
```

```
            double wsat=point.wsat(T);
            point.setW(wsat);
            point.calcQprime();
          }
          if(task.equals("modHum")){//modifies the gas composition
            point.modGasHum(false);
          }
        }
      }
      else{//calculs à effectuer dans le cas général
        if(updateT)point.setT(T);
        if(updateP)point.setP(P);
        if(updateX)point.setX(x);
        point.CalculeUnPoint();
      }
    }
```

La méthode getProperties() permet ensuite de récupérer les valeurs spécifiques, sachant que la méthode getPointProperties("nomPoint") d'ExtProcess permet de charger directement ces valeurs dans les double suivants : Wpoint pour l'humidité absolue w, Epsipoint pour l'humidité relative ε, Qprimepoint pour l'enthalpie spécifique q', Tprimepoint pour la température adiabatique t' (en °C), Trpoint pour la température de rosée tr (en °C), VPrimepoint pour le volume spécifique vs, Condpoint pour les condensats, et M_secpoint pour la masse molaire du gaz sec

```
public Vector getProperties(String[] args){ (partiel)
else if(type.equals("point")){
      PointCorps pt=getPoint(nomType);
      if(pt!=null){
        vProp.addElement(pt.lecorps);//Substance
        vProp.addElement(pt.lecorps.getNom());//Substance name
        vProp.addElement(new Double(pt.getT()));//Temperature
        vProp.addElement(new Double(pt.getP()));//Pressure
        vProp.addElement(new Double(pt.getXx()));//Quality
        vProp.addElement(new Double(pt.getV()));//Volume
        vProp.addElement(new Double(pt.getU()));//Internal energy
        vProp.addElement(new Double(pt.getH()));//Enthalpy
        vProp.addElement(new Double(pt.getS()));//Entropy
        String setTsat="set_Tsat="+Util.aff_b(pt.JCheckSetTsat.isSelected());
        vProp.addElement(setTsat);//setTsat
        vProp.addElement(new Double(pt.dTsat_value.getValue()));//DTsat
        String setpsat="set_psat="+Util.aff_b(pt.JCheckSetPsat.isSelected());
        vProp.addElement(setpsat);//setpsat

        //wet gas values
        vProp.addElement(new Double(pt.w_value.getValue()));//specific humidity
        vProp.addElement(new Double(pt.epsi_value.getValue()));//relative humidity
        vProp.addElement(new Double(pt.qprime_value.getValue()));//specific enthalpy
        vProp.addElement(new Double(pt.tprime_value.getValue()));//adiabatic temperature
        vProp.addElement(new Double(pt.tr_value.getValue()));//dew point temperature
        vProp.addElement(new Double(pt.v_spec_value.getValue()));//specific volume
        vProp.addElement(new Double(pt.cond_value.getValue()));//condensates
        vProp.addElement(new Double(pt.lecorps.M_sec));//Dry gas molar mass
      }
```