Modèle de combustion de biomasse

Cette note présente une documentation succincte de la classe externe BiomassCombustion. Il s'agit d'un modèle simplifié qui permet de simuler différents types de combustions de biomasse, et dans lequel il est possible de faire varier avec une assez grande souplesse la composition et l'humidité du combustible ainsi que les conditions de la combustion. La classe BiomassCombustion peut ainsi être utilisée aussi bien pour simuler une chaudière qu'un gazéifieur à tirage vers le bas.

Sur le plan thermodynamique, les principaux paramètres qui influencent la combustion de la biomasse sont les suivants :

- en premier lieu, bien évidemment, la composition du combustible
- en second lieu, son humidité, qui d'une part détermine l'enthalpie nécessaire au séchage, d'autre part joue sur la composition des gaz, et enfin influence la dissociation du CO_2
- enfin, la température de figeage et le taux de dissociation du CO_2

Afin de séparer autant que possible l'influence de ces deux premiers paramètres, la composition sera celle du combustible sec, et l'humidité sera prise en compte par un apport d'eau. Etant donné que l'humidité globale n'est pas nécessairement celle qui régit l'équilibre thermodynamique, une partie de la vapeur d'eau pouvant, pour des raisons de cinétique ou de géométrie, ne pas réagir, nous introduirons un paramètre complémentaire, égal à la fraction de l'eau intervenant dans la combustion. Physiquement, cela signifie qu'une fraction de l'eau n'est pas dissociée : elle doit être vaporisée mais joue sur la composition des gaz comme si elle était inerte.

Définition du combustible et déroulement de la combustion

La définition du combustible dans Thermoptim sera faite de la manière suivante (figure 1) :

- la composition du gaz sec, hors espèces non prises en compte dans le noyau de Thermoptim) est estimée, par exemple dans un tableur, et entrée sous forme d'un gaz composé (transfo-point "dry fuel")
- les espèces non considérées par le noyau de Thermoptim sont prises en compte séparément : elles sont entrées dans l'écran du composant, et font l'objet d'une précombustion
- le gaz sec est mélangé à de l'eau (vapeur condensable « eau ») pour former le combustible humide (transfo-point "humidity")
- la combustion finale est réalisée grâce aux fonctions du noyau de Thermoptim, qui sont émulées depuis la classe externe (mélangeur externe "biomass combustion"), en tenant compte de la précombustion et de l'enthalpie qui aurait été nécessaire pour vaporiser l'eau

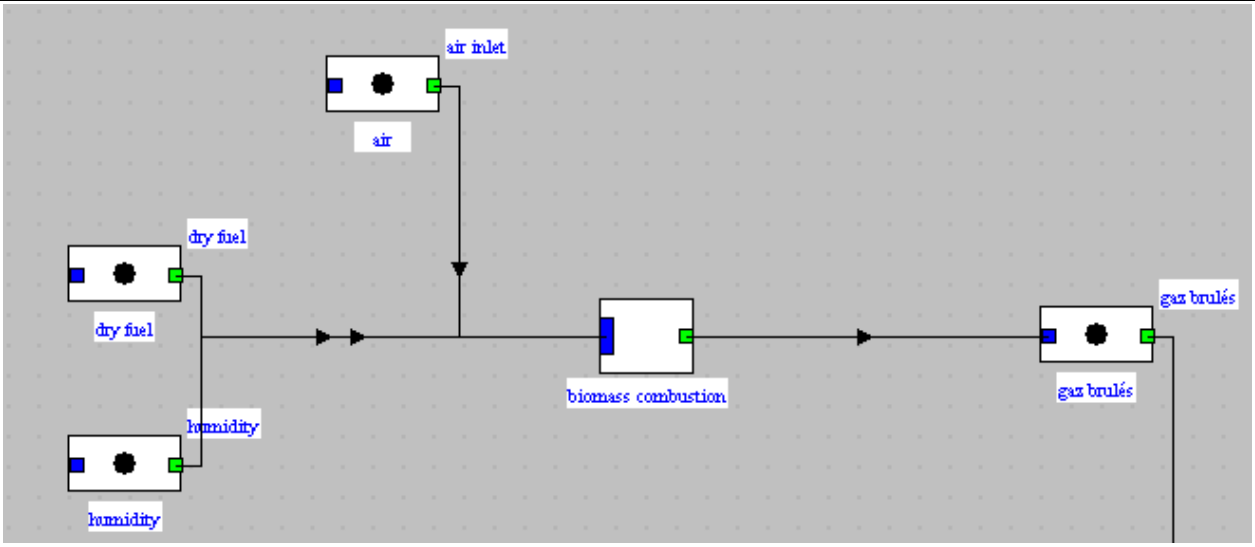

Figure 1 : Schéma du dispositif de combustion

Le comburant (généralement de l'air) est quant à lui défini dans la transfo-point "air inlet". Les débits mis en jeu sont eux aussi spécifiés dans chacune des trois transfos-points en entrée du mélangeur externe. Leurs rapports respectifs permettent ainsi de jouer sur l'humidité du combustible et sur l'excès d'air.

Attention : le mélangeur externe "biomass combustion" doit être **connecté à des transfos-points** et non pas à d'autres types de transfos, faute de quoi l'état des points n'est pas correctement mis à jour. Par ailleurs, dans le cas où le combustible et le comburant sont à des températures différentes, la température choisie pour la transfo-point apportant l'eau doit être choisie sur la base d'un raisonnement physique adéquat pour refléter la répartition de l'humidité entre ces deux flux entrants.

Exemple de combustion en défaut d'air (gazéifieur)

La gazéification de la biomasse correspond à l'oxydation partielle d'une ressource organique, principalement composée de cellulose ($C_6H_{10}O_5$), pour produire un gaz de synthèse. La biomasse étant généralement très humide, l'oxydation s'effectue en quatre étapes successives :

- séchage du combustible ;
- pyrolyse ou carbonisation (en défaut d'oxygène), produisant des goudrons et du carbone
- combustion du carbone et de l'oxygène, réaction exothermique assurant entre autres le séchage du combustible
- réduction du CO_2, de H_2 et de l'eau par le carbone et les goudrons

En réalisant une combustion en défaut d'air, on peut modéliser un gazéifieur (figure 2) à tirage vers le bas en utilisant la classe BiomassCombustion. Nous avons supposé ici que 50 % de l'humidité du combustible seulement intervient dans la combustion, le reste ne participant pas à la combustion.

La figure 2 montre l'écran du composant permettant de modéliser la combustion de la biomasse. Dans le cas présenté, on fait l'hypothèse que le combustible comprend 0,634 % d'ammoniac et 10,5 % de carbone rapportés à la masse sèche, que la température de figeage est égale à 900 °C, que le taux de dissociation du CO2 vaut 0,05, et que 50 % de l'humidité participe à la combustion.

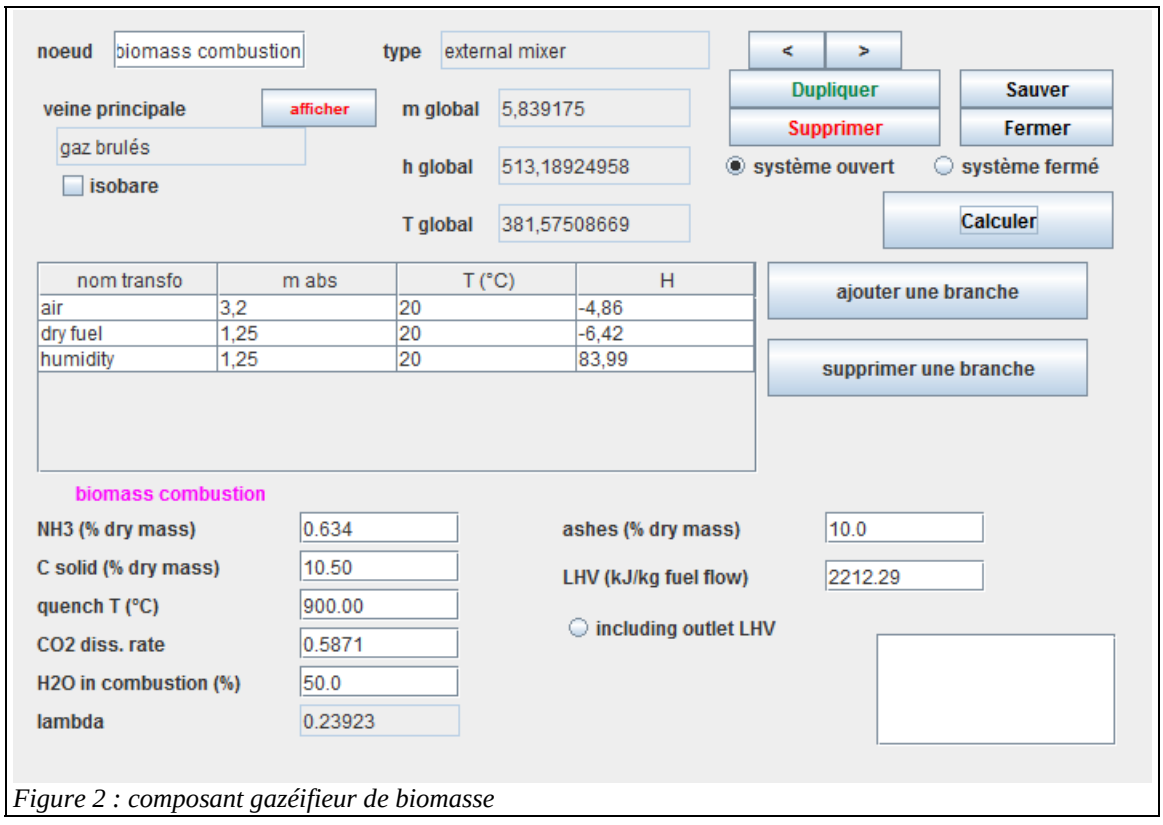

Figure 2 : composant gazéifieur de biomasse

Comme nous l'avons dit, les débits de combustible, d'eau et de comburant en entrée jouent bien évidemment un rôle fondamental : leurs rapports influencent directement la composition du gaz de synthèse (figure 3).

Des cendres peuvent être prises en compte dans le calcul du PCI. Deux modes sont prévus : soit inclure la valeur du PCI des gaz sortant du composant, soit ne pas en tenir compte. Dans le cas d'un gazéifieur comme ici, ce PCI sera valorisé en aval, ce qui justifie qu'il ne soit pas comptabilisé.

Le PCI affiché est ramené au kg de combustible, en incluant son humidité et les cendres éventuelles.

L'enthalpie totale correspondant à ce PCI est prise en compte dans le bilan global comme énergie payante.

Dans cet exemple, l'humidité du combustible est égale à 50 % en masse, les débits des deux transfos "dry fuel" et "humidity" étant égaux. Le débit d'air étant inférieur à celui qui aurait assuré une combustion stoechiométrique (environ 7,8 kg/s), la combustion a lieu en défaut d'air et le taux de dissociation du CO2 est recalculé.

nom du composant	fraction molaire	fraction massique
CO2	0,09095625	0,1780174
H2O	0,2800306	0,2243506
O2	0	0
N2	0,3332404	0,4151491
CO	0,1293294	0,1611006
H2	0,1626134	0,01457812
Ar	0,003829836	0,006804206

Figure 3 : composition du gaz de synthèse brut (PCI : 3,4 MJ/kg)

La température du gaz de synthèse dépend beaucoup la chaleur de réaction, elle-même fonction de l'oxygène disponible : s'il y en a très peu, c'est essentiellement du CO qui est produit, avec peu de CO_2.

Ce gaz de synthèse brut, très humide, peut être lavé et partiellement déshydraté, en utilisant la classe WaterQuench.

Accès aux calculs de combustion à partir des classes externes

Afin que les classes externes puissent accéder aux calculs de combustion disponibles dans Thermoptim, diverses modifications ont été apportées au progiciel. En particulier, une classe appelée ExternalCombustion, qui hérite directement de Combustion, a été ajoutée. Accessible à partir de MixerExterne, elle permet d'effectuer des calculs de combustion de la manière suivante :

- on commence par construire un Vector vSettings, qui comprend les trois gaz idéaux mis en jeu (comburant, combustible, gaz brûlés), ainsi que l'ensemble des paramétrages nécessaires pour spécifier les calculs à effectuer ;
- la méthode public void calcExternalCombustion(Vector vSettings) de MixerExterne initialise la combustion puis effectue les calculs demandés
- les résultats sont ensuite récupérés par la méthode public Vector getExternalCombustionResults() de MixerExterne, qui renvoie un Vector comprenant la composition des gaz brûlés, la température de fin de combustion, lambda, les énergies mises en jeu…

Depuis les classes externes (dans des mélangeurs externes uniquement, une chambre de combustion étant structurellement analogue à un mélangeur), l'accès se fait grâce aux méthodes de même nom d'ExtMixer, par des appels du type :
calcExternalCombustion(vSettings);
Vector results=getExternalCombustionResults();

Le paramétrage de la combustion est en tout point le même que celui qui est requis pour les combustions réalisées dans le noyau de Thermoptim, à la réserve près qu'il est possible d'imposer une charge thermique à prendre en compte pour le calcul de la température de fin de combustion.

Présentation de la classe externe

La classe est un mélangeur externe (figure 1), qui reçoit en entrée d'une part de l'eau, représentant l'humidité du combustible, et d'autre part deux gaz : le comburant, identifié par le fait qu'il contient à la fois de l'oxygène et de l'azote, et le combustible, qui ne doit pas contenir ces deux gaz à la fois. La veine principale de sortie correspond au gaz de synthèse. Les tests de cohérence sont effectués sur ces bases.

Le déroulement des calculs est le suivant :

1) calcul de la combustion de l'ammoniac et de celle du carbone solide (pour simplifier les choses, on a simplement modifié la composition du comburant, en faisant l'hypothèse implicite qu'il s'agissait d'air sec ; on aurait aussi pu appauvrir le comburant en oxygène et augmenter les fractions molaires en H2O et CO2 du combustible)

```
//initialisation des débits molaires de NH3 et de carbone
double NH3Flow=fuelFlow*Util.lit_d(NH3_value.getText())/100;//mass flow
double CFlow=fuelFlow*Util.lit_d(C_solid_value.getText())/100;//mass flow
double additionalFlow=NH3Flow+CFlow;//pour correction débit comburant
NH3Flow=NH3Flow/17.034;//molar flow
CFlow=CFlow/12;//molar flow

//analyse de la composition du comburant
Vector comburComp=comburSubstance.getGasComposition();
double fractO2=Util.molarComp(comburComp,"O2");//fraction molaire de O2
double comburMolFlow=comburFlow/comburM;
double O2MolFlow=comburMolFlow*fractO2-0.75*NH3Flow-CFlow;//débit d'oxygène restant dans le comburant
if(O2MolFlow<0){
    String msg = "Not enough oxygen in oxidizer for NH3 and C";
    JOptionPane.showMessageDialog(me, msg);
}
double fractN2=Util.molarComp(comburComp,"N2");//fraction molaire de N2
double fractAr=Util.molarComp(comburComp,"Ar");//fraction molaire de Ar
double totalCombFlow=O2MolFlow+(fractN2+fractAr)*comburMolFlow
+NH3Flow*(0.5// 1/2 N2
          +1.5// 3/2 H2O
          )+CFlow;// CO2
fractN2=(fractN2*comburMolFlow+0.5*NH3Flow)/totalCombFlow;
fractAr=fractAr*comburMolFlow/totalCombFlow;
double fractH2O=(1.5*NH3Flow)/totalCombFlow;
double fractCO2=CFlow/totalCombFlow;
fractO2=O2MolFlow/totalCombFlow;
```

```
//détermination de la composition du comburant modifié
Vector vComp=new Vector();
Double[]fracMol= new Double[5],molarMass= new Double[5];
String[] nom_gaz_comp = new String[5];
vComp.addElement(new Integer(5));
nom_gaz_comp[0]="N2";
nom_gaz_comp[1]="O2";
nom_gaz_comp[2]="Ar";
nom_gaz_comp[3]="H2O";
nom_gaz_comp[4]="CO2";
fracMol[0]=new Double(fractN2);
fracMol[1]=new Double(fractO2);
fracMol[2]=new Double(fractAr);
fracMol[3]=new Double(fractH2O);
fracMol[4]=new Double(fractCO2);
molarMass[0]=new Double(28.02);
molarMass[1]=new Double(32);
molarMass[2]=new Double(39.95);
molarMass[3]=new Double(18.02);
molarMass[4]=new Double(44.01);
vComp.addElement(nom_gaz_comp);
vComp.addElement(fracMol);
vComp.addElement(fracMol);
vComp.addElement(molarMass);

NewComburSubstance=Util.createIdealGas("comburant");//création d'un nouveau gaz composé
NewComburSubstance.updateGasComposition(vComp);//modification de la composition du comburant
```

2) humidification du combustible et calcul de la charge thermique à prendre en compte dans le reste de la combustion (ramenée à 1 kmol de combustible)

```
double waterFlow=massWaterFlow*Util.lit_d(H2O_comb_value.getText())/100.;//partie de l'eau prise en compte dans la combusti

double fractH2OFuel=1./(1.+fuelFlow/fuelM*18.01528/waterFlow);//fraction molaire de H2O

Vector fuelComp=fuelSubstance.getGasComposition();
double inlet_w=18.01528*fractH2OFuel/fuelM/(1-fractH2OFuel);//(M_H2O)(x_H2O)/(M_gs)/(x_gs)

//modification de la composition du combustible
NewFuelSubstance=Util.createIdealGas("wetFuel");//création d'un nouveau gaz composé

//dans un premier temps on charge la composition du gaz sec
NewFuelSubstance.updateGasComposition(fuelComp);

//puis on impose l'humidité
vComp=new Vector();
vComp.addElement(new Integer(-1));
vComp.addElement("setGasHum");
vComp.addElement(Util.aff_d(inlet_w));
NewFuelSubstance.updateGasComposition(vComp);

Vector vSubst=NewFuelSubstance.getSubstProperties();
Double z=(Double)vSubst.elementAt(7);
fuelM=z.doubleValue();//masse molaire du combustible humide

//la charge est égale à la chaleur libérée par la combustion de NH3 et C, moins la chaleur de vaporisation de l'eau
double Q=46190*NH3Flow+393520*CFlow-2444*massWaterFlow;//charge totale (kJ)
double heatLoad=Q*fuelM/(fuelFlow+waterFlow);//ramené à 1 kmol de combustible
```

3) initialisation et calcul de la combustion, puis récupération des résultats

```
//Initialisation de la combustion
Vector vSettings=new Vector();
vSettings.addElement(NewComburSubstance);
vSettings.addElement(NewFuelSubstance);
vSettings.addElement(new Double(comburFlow+additionalFlow));//oxidizer flow rate
vSettings.addElement(new Double(fuelFlow+waterFlow));//fuel flow rate
vSettings.addElement(new Double(3.));//lambda (inutilisé ici)
vSettings.addElement(new Double(1000.));//Tcomb (inutilisé ici)
vSettings.addElement(new Double(Util.lit_d(k_CO_value.getText())));//dissociation rate
vSettings.addElement("true");//dissociation boolean
vSettings.addElement(new Double(Util.lit_d(Tfig_value.getText())+273.15));//Tfigeage
vSettings.addElement(new Double(0.));//a (inutilisé ici)
vSettings.addElement(new Double(0.));//hfOcarb (inutilisé ici)
vSettings.addElement("false");//comb CHa
vSettings.addElement(new Double(Hcarb));//hc
vSettings.addElement(new Double(Hcarb));//uc
vSettings.addElement("true");//setFuelFlow
vSettings.addElement(new Double(heatLoad));//heat load
vSettings.addElement("true");//open system
vSettings.addElement("true");//IcalcDirect //si true on calcule Tad à partir de lambda
vSettings.addElement(synGasSubstance);
vSettings.addElement(new Double(Tcombur));//T amont
vSettings.addElement(new Double(Hcombur));//H amont
vSettings.addElement(new Double(1.));//rendement chambre

calcExternalCombustion(vSettings);//lancement des calculs de combustion

Vector results=getExternalCombustionResults();//Vector des résultats
```

4) mise à jour de l'humidité du gaz de sortie

```
//mise à jour de la composition du gaz pour tenir compte de l'eau non prise en compte dans la combustion
double waterExcessFlow=massWaterFlow*(1.-Util.lit_d(H2O_comb_value.getText())/100.);//partie de l'eau non prise en compte

synGasFlow=comburFlow+fuelFlow+additionalFlow+waterFlow;
double fractExcessH2O=1./(1.+synGasFlow/Msubst*18.01528/waterExcessFlow);//fraction molaire de H2O

fuelComp=synGasSubstance.getGasComposition();
Integer taille=(Integer)fuelComp.elementAt(0);
int nbComp=taille.intValue();

vComp=new Vector();
vComp.addElement(new Integer(nbComp));
fracMol= new Double[nbComp];molarMass= new Double[nbComp];
nom_gaz_comp = new String[nbComp];

for(int i=0;i<nbComp;i++){
    String[]noms=(String[])(fuelComp.elementAt(1));
    nom_gaz_comp[i]=noms[i];
    if(!nom_gaz_comp[i].equals("H2O")){
        Double[]frac=(Double[])(fuelComp.elementAt(2));
        double x=frac[i].doubleValue();
        fracMol[i]=new Double(x*(1-fractExcessH2O));
        Double[]M=(Double[])(fuelComp.elementAt(4));
        molarMass[i]=M[i];
    }
    else{
        Double[]frac=(Double[])(fuelComp.elementAt(2));
        double x=frac[i].doubleValue();
        fracMol[i]=new Double(x*(1-fractExcessH2O)+fractExcessH2O);
        Double[]M=(Double[])(fuelComp.elementAt(4));
        molarMass[i]=M[i];
    }
}
vComp.addElement(nom_gaz_comp);
vComp.addElement(fracMol);
vComp.addElement(fracMol);
vComp.addElement(molarMass);

synGasSubstance.updateGasComposition(vComp);//modification de la composition du gaz en sortie
```

5) calcul de la température de sortie et mise à jour du mélangeur

```
//recherche de la température de sortie (on suppose que l'enthalpie de l'eau est égale à
double Hout=Houtlet*synGasFlow+Hcarb*waterExcessFlow;
DeltaH=Hout-Hcarb*(massWaterFlow+fuelFlow)-Hcombur*comburFlow;
synGasFlow=synGasFlow+waterExcessFlow;
Hout=Hout/synGasFlow;
double Tout=synGasSubstance.getT_from_hP(Hout,Pamont);


Vector vCorps =fuelSubstance.getSubstProperties();
double PCI=(Double)vCorps.elementAt(13);

vCorps =synGasSubstance.getSubstProperties();
double outPCI=(Double)vCorps.elementAt(13);
purchasedEnergy=fuelFlow*PCI+46190*NH3Flow+393520*CFlow;
if(!outletLHV_RadioButton.isSelected())purchasedEnergy=purchasedEnergy-synGasFlow*outPCI
fuelPCI=purchasedEnergy/(fuelFlow+massWaterFlow)/(1.+Util.lit_d(ashes_value.getText())/1
PCI_value.setText(Util.aff_d(fuelPCI, 2));

//mise à jour du noeud en utilisant les méthodes génériques
vTransfo= new Vector[nBranches+1];
vPoints= new Vector[nBranches+1];
setupVector(fuelProcess, fuelPoint, 0, fuelFlow, Tcarb, Pamont, 1);
setupVector(comburProcess, comburPoint, 1, comburFlow, Tcombur, Pamont, 1);
setupVector(waterProcess, waterPoint, 2, massWaterFlow, Twater, Pamont, 1);
setupVector(synGasProcess, synGasPoint, 3, synGasFlow, Tout, Pamont, 1);
updateMixer(vTransfo,vPoints,Tout,Hout);

//définit l'énergie payante au niveau global
me.updateProcess(setEnergyTypes(synGasProcess,0,purchasedEnergy,0));//DeltaH énergie pay
}
```