Batteries de refroidissement avec condensation

Pour refroidir un mélange humide, on le fait passer dans un échangeur particulier appelé batterie froide ou de refroidissement, qui peut être refroidi par de l'eau glacée ou par évaporation directe d'un fluide frigorigène (figure 1). Le mélange étant en contact avec des surfaces froides voit sa température diminuer. Selon les cas, il peut y avoir ou non condensation. S'il n'y a pas de condensation, l'humidité spécifique reste constante, et le refroidissement peut être représenté par un segment horizontal orienté vers la gauche dans le diagramme psychrométrique, et vertical orienté vers le bas dans le diagramme de Mollier. S'il y a condensation, ce qui est très souvent le cas, il s'agit d'un segment orienté en bas et à gauche dans les deux diagrammes (figure 2).

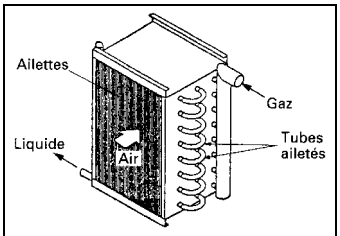

Figure 1 (Extrait de Techniques de l'Ingénieur - Génie énergétique)

Un refroidissement théorique parfait dans une batterie froide de dimension infinie conduirait à refroidir le mélange humide jusqu'à la température de la batterie à l'état saturé. On a coutume de qualifier une transformation réelle en prenant ce refroidissement théorique comme référence, en introduisant l'efficacité ε de la batterie froide.

$$\varepsilon = \frac{w_2 - w_1}{w_{sat} - w_1}$$

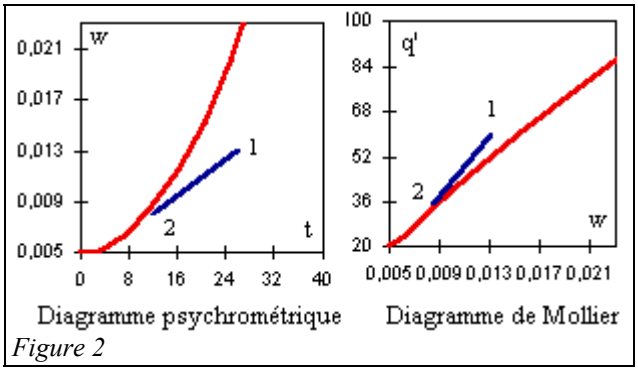

Figure 2

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.

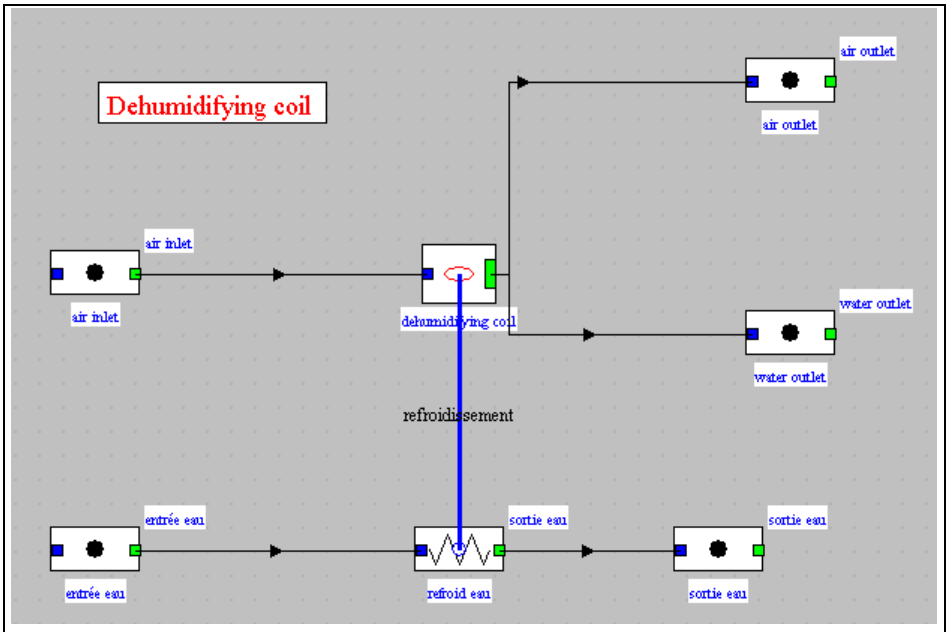

Figure 3 : Schéma du composant batterie de refroidissement avec condensation

La structure du modèle est donnée figure 3.

Modèle de la batterie de refroidissement avec condensation

Le modèle que nous développons ici est celui qui est présenté section 4.8.4.1 du livre "Systèmes Energétiques".

Connaissant l'efficacité ε de la batterie froide et sa température moyenne de surface t_s, le calcul de l'état de l'air en sortie est effectué de la manière suivante.

On commence par rechercher les conditions de saturation $w_{sat}(t_s)$ et $q'(t_s,w_{sat})$

On calcule ensuite les conditions humides finales compte tenu de l'efficacité :

$w_2 = \varepsilon\, w_{sat} + (1 - \varepsilon)\, w_1$

$q'_2 = \varepsilon\, q'(t_s,w_{sat}) + (1 - \varepsilon)\, q'_1$

On cherche alors t_2 tel que $q'_2 = q'(t_2,w_2)$.

Un problème peut survenir lorsque, dans le diagramme psychrométrique, la droite issue du point 1 coupe la courbe de saturation en deux points. En effet, dans ce cas, il se peut que le point calculé à partir de l'efficacité se trouve dans la zone saturée. Il y a alors condensation anticipée. Deux possibilités existent : si l'on impose l'humidité spécifique du point aval w2, le point final se trouve alors sur la courbe de saturation pour wsat = w2 ; si l'on impose l'efficacité, il se trouve sur la courbe de saturation, pour q'sat = q'2.

Si l'on veut faire un calcul exact, les coordonnées du deuxième point d'intersection de la droite issue du point amont avec la courbe de saturation peuvent être obtenues en éliminant ε entre les deux équations précédentes, avec w2 et q'2 égaux à leurs valeurs de saturation.

Une manière approchée consiste à déterminer la température $t_{app} = \varepsilon\, t_{sat} + (1 - \varepsilon)\, t_1$, et à la considérer comme une estimation de t_2, puis à calculer wsat(tapp). Si w2 > wsat(tapp), le point est dans la zone saturée. C'est ce test qui a été implémenté dans la classe présentée ici.

Implémentation du modèle

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[1]" pour une présentation détaillée des méthodes disponibles.

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

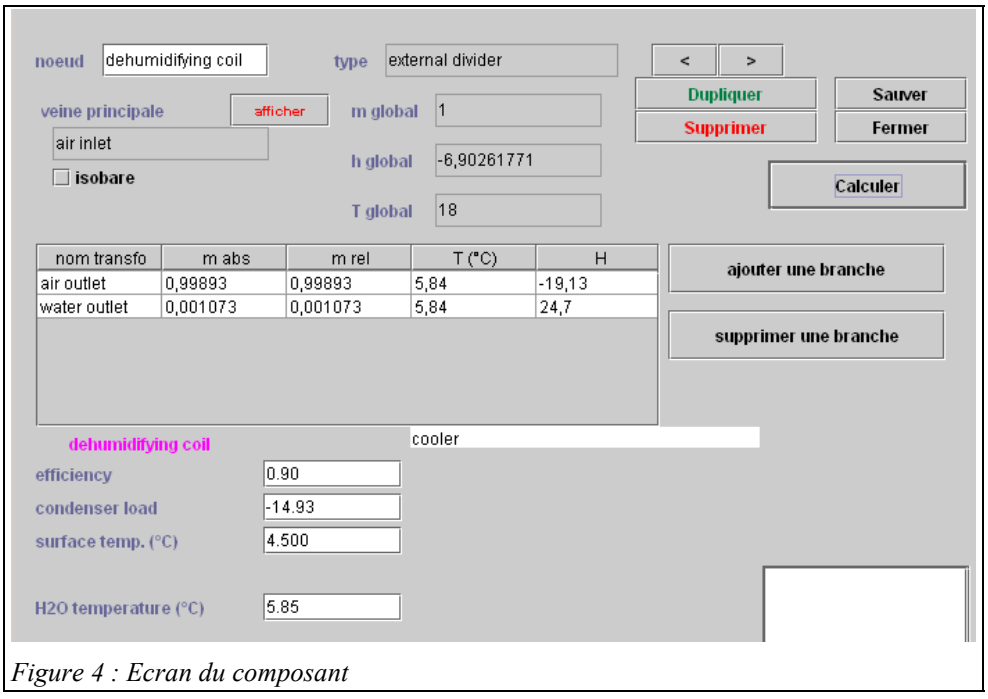

Figure 4 : Ecran du composant

[1] http://www.thermoptim.org/sections/base-methodologique/extensions-thermoptim/calculs-gaz-humides

1) les seuls paramètres sont d'une part la valeur de l'efficacité ε de la batterie, lue à l'écran, et d'autre part celle de la température de surface, que nous supposerons être égale à celle du fluide de refroidissement à l'entrée de la batterie ;
2) on commence par calculer les propriétés humides de l'air entrant, et on détermine le débit-masse de gaz sec ;
3) on calcule l'état de l'air en sortie à la saturation, et l'état final ("2" dans les équations précédentes) ;
4) la température en sortie de batterie est imposée, et les propriétés humides de l'air de sortie sont calculées, ce qui fournit les enthalpies spécifique et totale à extraire de l'air
5) le débit d'eau cédé par l'air est déterminé et la composition de l'air humide en sortie est modifiée
6) le bilan enthalpique fournit la charge du thermocoupleur
7) les valeurs en aval du noeud sont mises à jour

L'écran du composant est donné figure 4 et celui du thermocoupleur figure 5.

Afin de comparer les calculs effectués par la classe externe et par la transfo refroidissement humide interne à Thermoptim, on a instancié cette dernière en la couplant au même point d'entrée et à un point de sortie complémentaire.

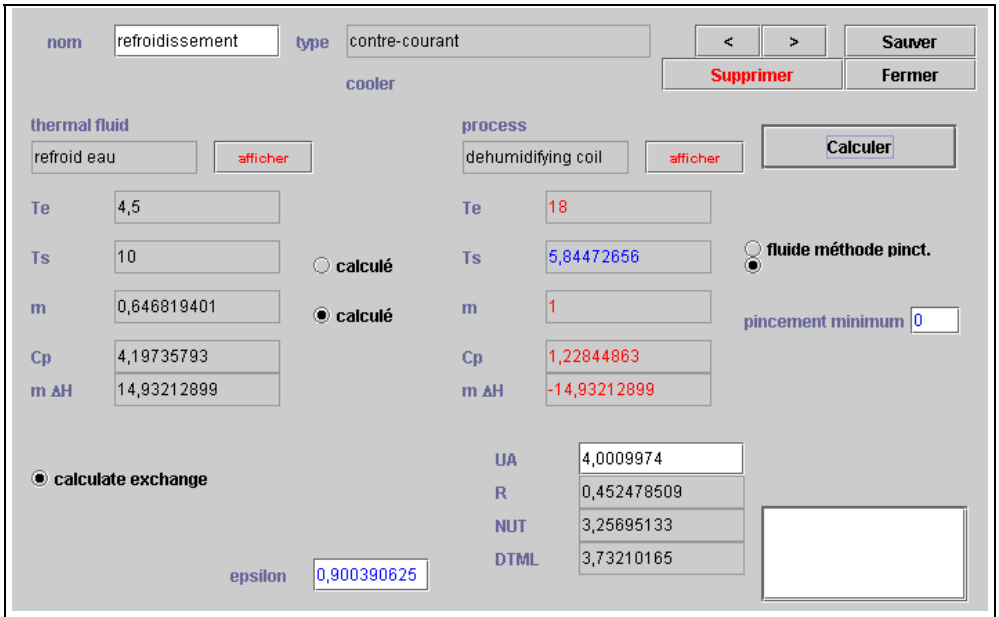

Figure 5 : Ecran du thermocoupleur

Les résultats obtenus sont tout à fait les mêmes, aux erreurs d'arrondi près, comme le montre la figure 6.

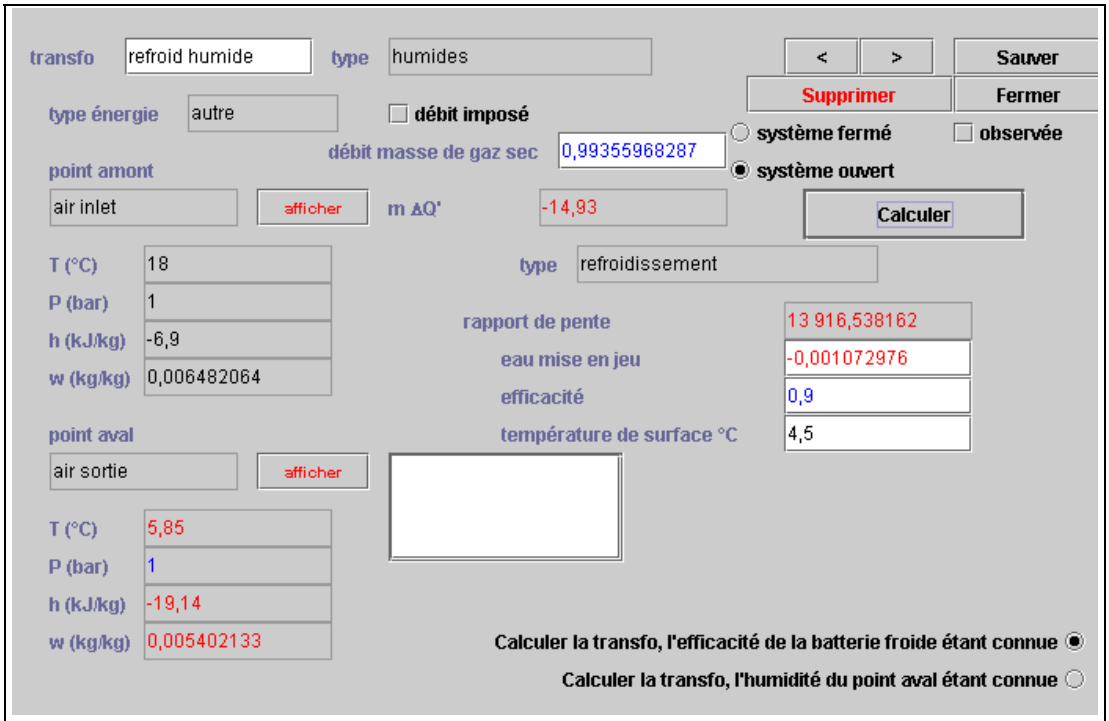

Figure 6 : Ecran de la transfo refroidissement humide interne à Thermoptim

La fraction molaire de l'eau a été divisée par 11 environ (figures 7 et 8)

nom du composant	fraction molaire	fraction massique
Ar	0,008907194	0,01233639
O2	0,2078345	0,2305589
N2	0,7729465	0,7506644
H2O	0,01031183	0,006440317

Figure 7 : Composition de l'air humide en entrée

nom du composant	fraction molaire	fraction massique
Ar	0,008922523	0,01234965
O2	0,2081922	0,2308066
N2	0,7742767	0,7514707
H2O	0,008608637	0,005373107

Figure 8 : Composition de l'air humide en sortie

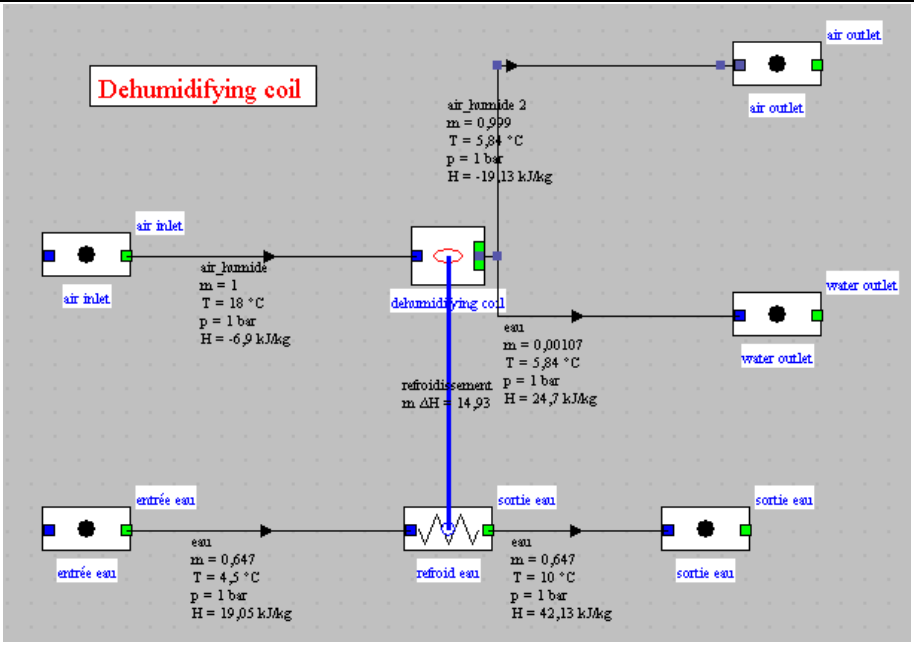

Figure 9 : Synoptique du composant

Etude de la classe externe DehumidifyingCoil

Des tests de cohérence du nœud sont effectués par la méthode checkConsistenc() 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 DehumidifyingCoil 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 les propriétés humides de l'air entrant, grâce à la méthode générique updatepoint(), puis on initialise le débit d'air sec et l'enthalpie spécifique amont :

```
    //Initialisations relatives à l'état de l'air entrant
    getPointProperties(wetGasPoint);
    wInlet=Wpoint;
    double flow_as=wetGasFlow/(1+Wpoint);//débit massique de gaz sec

    //calcul des propriétés humides en entrée du composant
    updatepoint(wetGasPoint, false, 0,  //T
        false, 0, false, 0,//P,x
        true, "setW and calculate all", wInlet);
    getPointProperties(wetGasPoint);//direct parsing of point property vector
    double qprimeInlet=QPrimepoint;
    System.out.println(" winlet : "+Wpoint+" epsi : "+Epsipoint+" q' : "+QPrimepoint+" t' : "+Tprimep
```

2) on initialise ensuite la valeur de la température de surface

```
//récupération du nom de la transfo couplée au composant par le thermocoupleur
String thcoupl=(String)de.de.getThermoCouplerData("cooler").elementAt(0);
    System.out.println(" thermocoupler : "+thcoupl);
if((thcoupl!=null)&&(!thcoupl.equals("null"))){
    String[] args=new String[2];
    args[0]="heatEx";
    args[1]=thcoupl;
    Vector vProp=proj.getProperties(args);
    String trsf=(String)vProp.elementAt(0);
    System.out.println("hot : "+trsf+" cold : "+(String)vProp.elementAt(1));
    args[0]="process";
    args[1]=trsf;
    vProp=proj.getProperties(args);
    String amont=(String)vProp.elementAt(1);
    getPointProperties(amont);
    //Tpoint contient la valeur de la température d'entrée du fluide de refroidissement
    ts_value.setText(Util.aff_d(Tpoint-273.15,3));
}
double ts=Util.lit_d(ts_value.getText());
```

3) on calcule ensuite les conditions de saturation pour l'air sortant

```
    //calcul des conditions de saturation pour l'air sortant
    updatepoint(dryGasPoint, true, ts+273.15,  //T
        false, 0, false, 0,//P,x
        false, "", 0);

    updatepoint(dryGasPoint, false, 0,  //T
            false, 0, false, 0,//P,x
            true, "setEpsi and calculate", 1);
    getPointProperties(dryGasPoint);//propriétés à la saturation

    double wsat_ts=Wpoint;
    double qprime_ts=QPrimepoint;
    System.out.println(" wsat : "+Wpoint+" epsi : "+Epsipoint+" q' : "+QPrimepoint+" t' : "+
```

4) on effectue le calcul de l'état final recherché

```
    //calcul de l'état final recherché
    w2=epsi*wsat_ts+(1-epsi)*wInlet;
    qprime2=epsi*qprime_ts+(1-epsi)*qprimeInlet;
```

5) on cherche à savoir si le point ainsi déterminé risque d'être en zone saturée

```
    //préparation du test sur la zone saturée
    double tapp=epsi*ts+(1-epsi)*(Tinlet-273.15);
    updatepoint(dryGasPoint, true, tapp+273.15,  //T
        false, 0, false, 0,//P,x
        false, "", 0);

    updatepoint(dryGasPoint, false, 0,  //T
            false, 0, false, 0,//P,x
            true, "setEpsi and calculate", 1);
    getPointProperties(dryGasPoint);//propriétés à la saturation
    double wsat_tapp=Wpoint;
```

6) on recherche de la température sèche correspondante grâce à la fonction d'inversion générique Util.dicho (qui fait appel à f_dicho).

```
    double T=0;
    if(w2>wsat_tapp){//Si le point est dans la zone saturée
        T=Util.dicho_T(this, 0, Pinlet, "qprime2sat", ts+273.15, Tinlet, 0.01);
    }
    else{//sinon
        T=Util.dicho_T(this, 0, Pinlet, "qprime2", ts+273.15, Tinlet, 0.01);
    }
public double f_dicho(double T, double P, String fonc){

    if (fonc.equals("qprime2")){//point final hors zone saturée
        double diff;
        updatepoint(dryGasPoint, true, T,  //T
            false, 0, false, 0,//P,x
            false, "", 0);
        updatepoint(dryGasPoint, false, 0,  //T
            false, 0, false, 0,//P,x
            true, "setW and calculate q'", w2);

        getPointProperties(dryGasPoint);//propriétés humides
        diff=qprime2-QPrimepoint;
        return diff;
    }
    if (fonc.equals("qprime2sat")){//point final saturé
        double diff;
        updatepoint(dryGasPoint, true, T,  //T
            false, 0, false, 0,//P,x
            false, "", 0);
        updatepoint(dryGasPoint, false, 0,  //T
                false, 0, false, 0,//P,x
                true, "setEpsi and calculate", 1);

        getPointProperties(dryGasPoint);//propriétés h"
        diff=qprime2-QPrimepoint;
        return diff;
    }
    return 0;
}
```

7) on modifie la composition de l'air humide en sortie, et on calcule la charge du thermocoupleur et le débits-masse d'eau sortant :

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

        //charge du condenseur
        double DeltaQprime=flow_as*(qprime2-qprimeInlet);//enthalpie totale acquise par l'air
        double condLoad=DeltaQprime;

        //eau mise en jeu
        double H2Oflow=(wInlet-Wpoint)*flow_as;
```

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

```
//affichages à l'écran
condenserLoad_value.setText(Util.aff_d(condLoad,2));
Tout_value.setText(Util.aff_d(T,2));
epsi_value.setText(Util.aff_d(epsi,2));

//mise à jour des thermocoupleurs, comme pinchFluid, DTmin=10
updateThermoCoupler("cooler", Tinlet, T, 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, T, Pinlet, 0);
setupVector(dryGasProcess, dryGasPoint, 1, wetGasFlow-H2Oflow, T, Pinlet, 0);
setupVector(wetGasProcess, wetGasPoint, 2, wetGasFlow, Tinlet, Pinlet, 1);
updateDivider(vTransfo,vPoints,Tinlet,Hinlet*wetGasFlow);

de.updateProcess(setEnergyTypes(wetGasProcess,0,0,0));
```