Etude d'un cycle CLC pour la capture du CO2 par oxy-combustion

Description du cycle

Le cycle CLC (Chemical Looping Combustion) est l'un des cycles innovants de production d'électricité faisant appel à l'oxy-combustion. Son schéma de principe est donné figure 1.

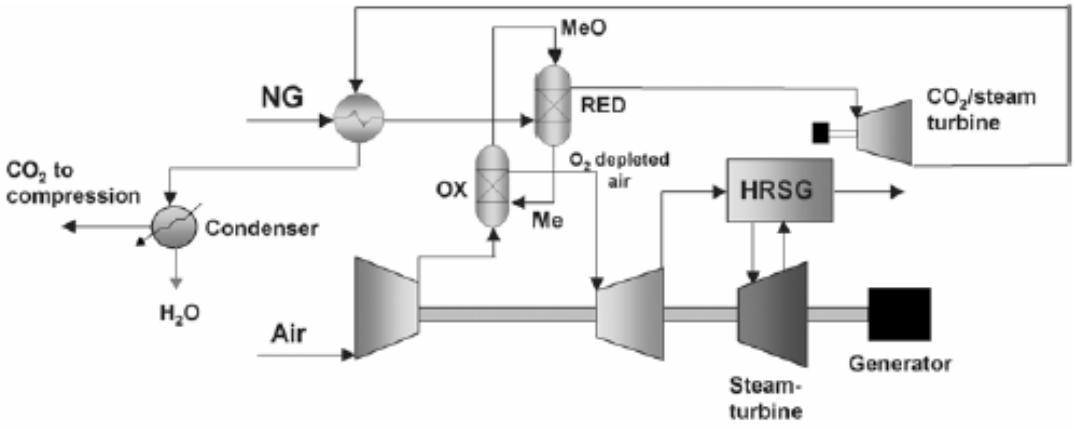


Figure 1 : Schéma d'un cycle CLC

Un débit d'air de 630 kg/s est aspiré par le compresseur d'une turbine à gaz modifiée, où la chambre de combustion est remplacée par une enceinte à deux compartiments entre lesquels circule un oxyde métallique comme NiO. Dans l'une des enceintes, l'air est appauvri en oxygène du fait de l'oxydation du métal. Dans l'autre, l'oxyde est réduit et l'oxygène dégagé brûle avec un combustible.

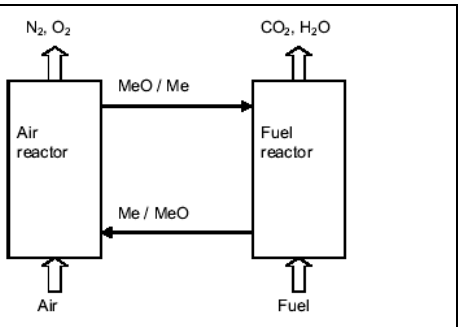


Fig. 1. Chemical-looping combustion. MeO/Me denote recirculating oxygen-carrier particles. (A full conversion from MeO to Me and back to MeO, is not needed in the reactors.)

Figure 2 : Schéma de principe

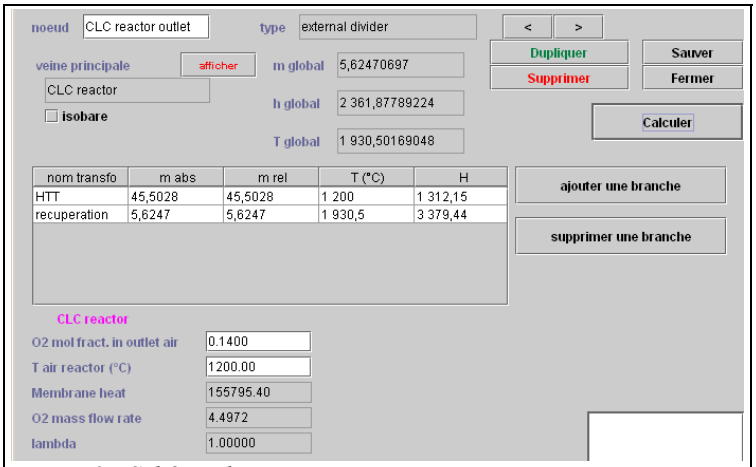


Figure 3 : Schéma de principe

Le schéma de principe du "chemical looping" est donné figure 2. Dans le modèle très simple que nous avons implémenté, on fixe d'une part la température du réacteur à air et d'autre part la fraction molaire d'oxygène dans l'air appauvri, comme indiqué sur l'écran de la figure 3. Le modèle en déduit le débit d'oxygène et la puissance thermique transférés, sachant que la réaction est stoechiométrique ($\lambda = 1$).

L'air appauvri (fraction molaire d'oxygène voisine de 0,14) est ensuite détendu dans une turbine, puis utilisé comme source chaude pour un cycle à vapeur. Les gaz brûlés sont eux aussi détendus dans une turbine, puis utilisés comme source chaude pour le cycle à vapeur, avant d'être condensés pour extraction de l'eau. Le CO2 restant peut alors être capturé.

Un schéma possible du cycle est donné figure 4. Avec le paramétrage retenu, qui néglige en particulier toutes les pertes de charge, il conduit à un rendement de près de 55 %.

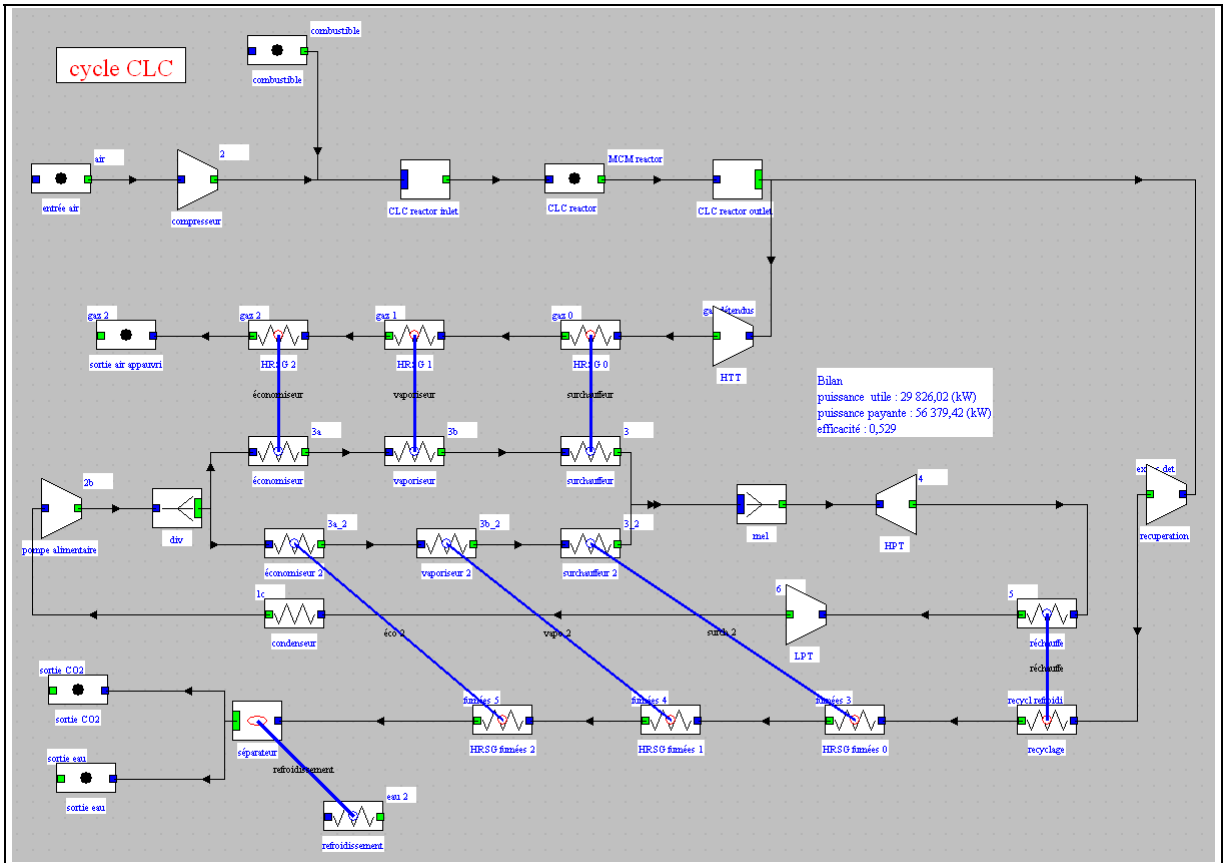


Figure 4 : Schéma du cycle CLC

Dans ce cycle, nous avons choisi de scinder le générateur de vapeur en deux parties, l'une chauffée par l'air appauvri, et l'autre par les gaz brûlés. La pression et la température maximales du cycle vapeur seront initialement fixées à 70 bars et 400 °C. Elles pourront ultérieurement faire l'objet d'études de sensibilité.La répartition du débit entre les deux veines dépend du paramétrage retenu. Dans le cas présenté, elle est de l'ordre de 70%/30%. Le débit de vapeur d'eau sera fixé initialement à 80 kg/s, puis sera augmenté jusqu'à ce que le pincement dans le générateur de vapeur soit de l'ordre de 12 °C.

Les rendements polytropiques des compresseurs et turbines sont tous supposés égaux à 0,85.

On dispose, en sortie de générateur de vapeur, de fumées contenant CO2 et H2O. Elles sont refroidies par une source froide externe, jusqu'à une température suffisamment basse pour que presque toute l'eau soit condensée.

En sortie de condenseur, le gaz composé principalement de CO2 est capturé. On n'a pas fait apparaître sur le schéma les compresseurs correspondants.

Ce schéma fait appel à un diviseur externe ("cold battery", voir note "Utilisation des composants externes") pour représenter la condensation de l'eau contenue dans les fumées. Il s'agit d'un modèle simplifié, qui comporte deux paramètres, la température de l'eau, que l'on prendra égale à 30 °C, et l'efficacité d'extraction de l'eau, qui représente le pourcentage (en volume) de l'eau condensée par rapport à l'eau entrante.

Modèle du réacteur

L'ensemble composé par les deux réacteur présente la particularité de mettre en jeu deux flux séparés : l'air et les gaz qui entrent en jeu dans la réaction de combustion, qui échangent de la matière et de l'énergie par l'intermédiaire de l'oxyde métallique. Il se comporte donc comme un quadripôle recevant deux fluides en entrée, et dont en sortent deux autres.

Pour le représenter dans Thermoptim, on forme ce quadripôle en associant un mélangeur en entrée (classe CLCReactorInlet) et un diviseur en sortie (classe CLCReactor), les deux étant reliés par une transfo-point qui joue un rôle purement passif. Les intitulés des classes sont "CLC reactor inlet" et "CLC reactor".

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.

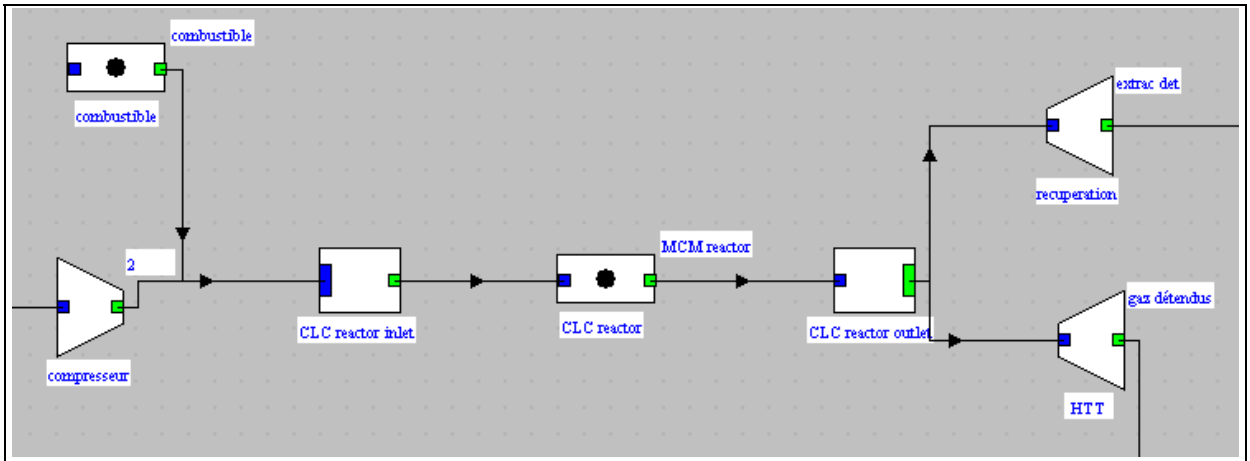


Figure 5 : Schéma du composant réacteur

La structure du modèle est donnée figure 5. Les réacteurs sont représentés par les 3 composants "CLC reactor inlet", "CLC reactor" et "CLC reactor outlet". Le mélangeur d'entrée CLC reactor inlet reçoit d'une part l'air comprimé, et d'autre part le combustible. En sortie du diviseur externe, on retrouve d'une part l'air appauvri , qui est détendu dans la turbine HTT, et d'autre part les gaz brûlés.

Le modèle que nous avons construit est purement global. On considère que l'on se donne la température et la composition de l'air appauvri. Il devient alors possible de déterminer le débit d'oxygène transféré entre les deux compartiments du réacteur.

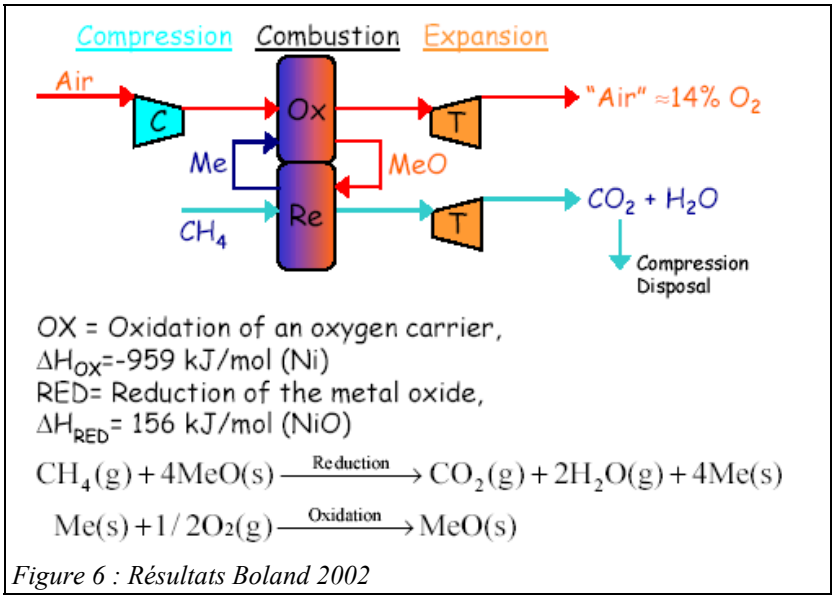


Figure 6 : Résultats Boland 2002

Dans la chambre de combustion, nous supposerons que la réaction est stoechiométrique et complète. Etant donné que la température de fin de combustion atteinte dépasse la limite de 2600 K retenue pour le calcul des gaz dans Thermoptim, on impose une charge thermique fictive égale à 80 % du PCI, ce qui fournit une température fictive. On effectue ensuite un bilan global du réacteur, en écrivant que la somme des enthalpies sortantes est égale à celle des enthalpies entrantes, plus les 80 % du PCI non pris en compte, ce qui permet de trouver la température de sortie des gaz brûlés, composés uniquement de CO2 et de H2O.

Il est clair que ce modèle est insuffisant, notamment du fait qu'il ne tient pas compte des pertes dues à la circulation de l'oxyde métallique. Il pourra être complété par la suite. En jouant sur le paramétrage, on arrive à retrouver pour ce cycle des valeurs proches de celles annoncées par Bolland[1] (figures 6 et 7).

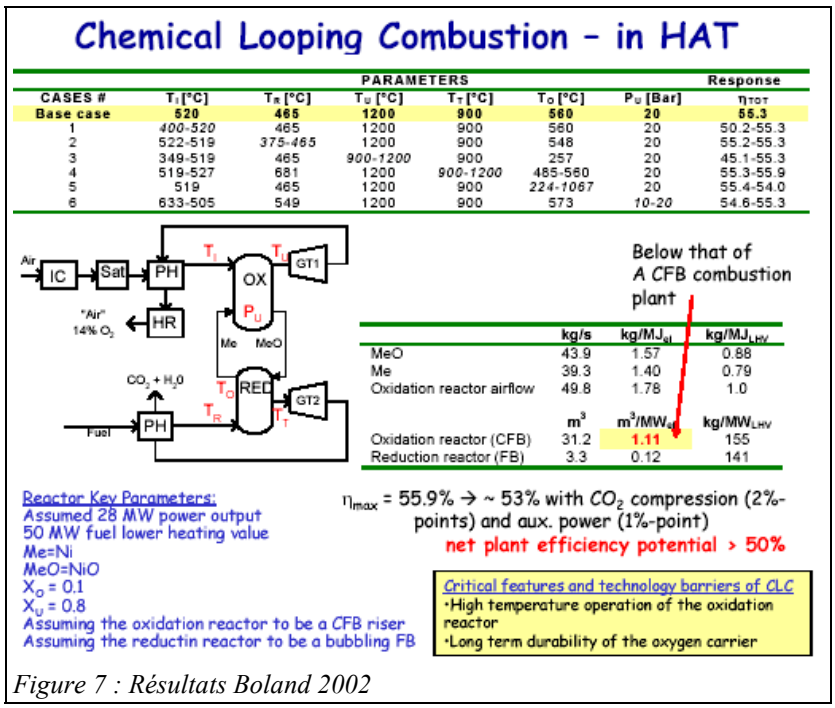


Figure 7 : Résultats Boland 2002

Toutefois, nous manquons de données précises sur le cycle pour pouvoir valider précisément notre modèle.

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

1) la température et la fraction molaire de l'air appauvri sont des paramètres lus à l'écran ;
2) on détermine la composition de l'air appauvri;
3) on calcule la réaction de combustion, ce qui détermine la température de fin de combustion et le débit de combustible
4) la température de fin de combustion est recalculée comme expliqué ci-dessus ;
5) on modifie en conséquence le débit d'entrée de combustible et les valeurs de paramétrage des transfos de sortie du composant

Etude de la classe externe CLCReactor

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 CLCReactor permet de voir comment le modèle a été implémenté. Six étapes suffisent pour effectuer les calculs :

1) on on commence par calculer le débit molaire d'oxygène mis en jeu :

[1] O. Bolland, Options for oxy-fuels & pre-combustion decarbonisation cycles, Presentation for Alstom Cross Segment CO2 Mitigation Group, October 11th 2002

```
//analyse de la composition de l'air
Vector airComp=mcmi.airSubstance.getGasComposition();
double fractO2=Util.molarComp(airComp,"O2");//fraction molaire de O2
double airMolFlow=mcmi.airFlow/mcmi.airM;//débit molaire d'air
double PO2=mcmi.Pair*fractO2;//pression partielle d'oxygène dans l'air
double Tmemb=Util.lit_d(Tmemb_value.getText())+273.15;//température de membrane lue à l'écran
double fractO2out=Util.lit_d(O2outFract_value.getText());
double totalDepletedAirMolFlow=airMolFlow*(1.-fractO2)/(1.-fractO2out);

double O2DepletedAirMolFlow=airMolFlow-totalDepletedAirMolFlow;//débit d'oxygène pour la combustion
```

2) on calcule ensuite la composition et le débit de l'air appauvri

```
//détermination de la composition de l'air appauvri
Vector vComp=new Vector();
Double[]fracMol= new Double[3],molarMass= new Double[3];
String[] nom_gaz_comp = new String[3];
vComp.addElement(new Integer(3));
nom_gaz_comp[0]="N2";
nom_gaz_comp[1]="O2";
nom_gaz_comp[2]="Ar";
fracMol[0]=new Double(fractN2);
fracMol[1]=new Double(fractO2);
fracMol[2]=new Double(fractAr);
molarMass[0]=new Double(28.02);
molarMass[1]=new Double(32);
molarMass[2]=new Double(39.95);
vComp.addElement(nom_gaz_comp);
vComp.addElement(fracMol);
vComp.addElement(fracMol);
vComp.addElement(molarMass);
O2outFract_value.setText(Util.aff_d(fractO2,4));

depletedAirSubstance.updateGasComposition(vComp);//modification de la composition de l'air appauvri

Vector vSubst=depletedAirSubstance.getSubstProperties();
Double z=(Double)vSubst.elementAt(7);
double airM=z.doubleValue();//masse molaire de l'air appauvri
double depletedAirFlow=totalDepletedAirMolFlow*airM;//débit massique d'air appauvri
O2Flow_value.setText(Util.aff_d(mcmi.airFlow-depletedAirFlow,4));//débit massique d'oxygène transféré
```

3) on calcule ensuite la réaction de combustion partielle en procédant comme indiqué plus haut :

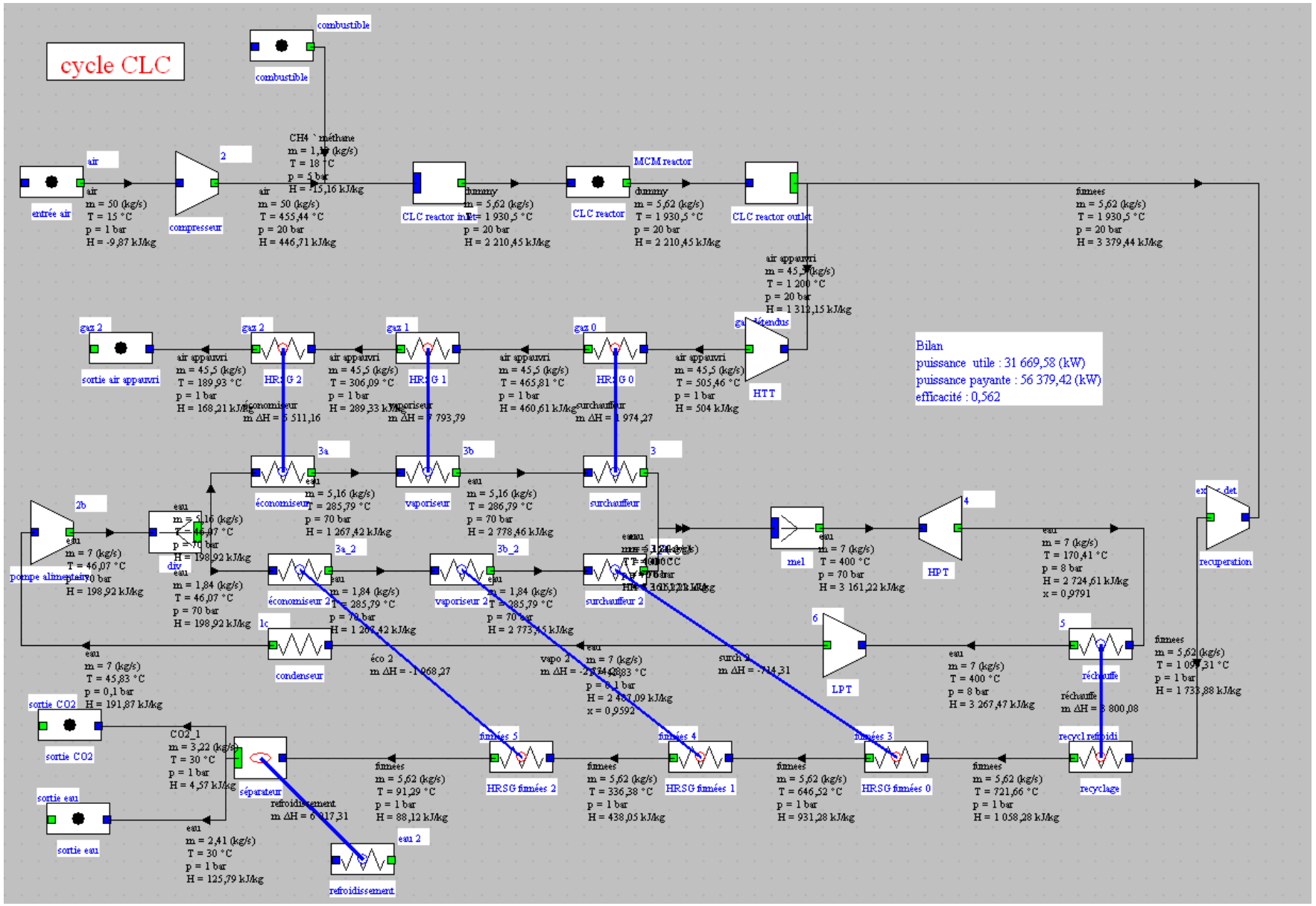
```
//on impose une charge artificielle pour éviter d'avoir une température trop élevée
double heatLoad=-PCI*mcmi.fuelM*0.8;//ramené à 1 kmol de combustible
System.out.println("heatLoad : "+heatLoad);

//Initialisation de la combustion
//on fait l'hypothèse d'une combustion stoechiométrique complète
//Sinon, il faut prévoir une conversion shift pour transformer le CO
Vector vSettings=new Vector();
vSettings.addElement(NewComburSubstance);
vSettings.addElement(mcmi.fuelSubstance);
vSettings.addElement(new Double(totalCombFlow));//oxidizer flow rate
vSettings.addElement(new Double(mcmi.fuelFlow));//fuel flow rate
vSettings.addElement(new Double(1.));//lambda (combustion stoechiométrique)
vSettings.addElement(new Double(Tmemb));//Tcomb
vSettings.addElement(new Double(0.));//dissociation rate (inutilisé ici)
vSettings.addElement("false");//dissociation boolean (pas de dissociation)
vSettings.addElement(new Double(273.15));//Tfigeage (inutilisé ici)
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("false");//setFuelFlow
vSettings.addElement(new Double(heatLoad));//heat load
vSettings.addElement("true");//open system
vSettings.addElement("true");//IcalcDirect //si true on calcule Tcomb à partir de lambda
vSettings.addElement(burntGasSubstance);
vSettings.addElement(new Double(mcmi.Tair));//T amont
vSettings.addElement(new Double(mcmi.Hair));//H amont
vSettings.addElement(new Double(1.));//rendement chambre

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

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

4) on détermine enfin la température des gaz brûlés en faisant la bilan enthalpique global des réacteurs et en réinjectant les 80 % du PCI artificiellement enlevés

```
depletedAirSubstance.CalcPropCorps(Tmemb,Pamont, 1); //recalcul de l'état de sortie (fournit Hsubst)
getSubstProperties(depletedAirSubstanceName);
double HdepletedAir=Hsubst;

burntGasSubstance.CalcPropCorps(T_burntGas,Pamont, 1); //recalcul de l'état de sortie (fournit Hsubst)
getSubstProperties(burntGasSubstanceName);

double HcombArtificiel=Hsubst;

double HoutCO2=HcombArtificiel+0.8*PCI*fuelFlow/burntGasFlow-(depletedAirFlow)/burntGasFlow*(HdepletedAir-mcmi.Hair);
T_burntGas=burntGasSubstance.getT_from_hP(HoutCO2,1.);
double Houtlet=Hsubst;
```

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

```
//mise à jour du noeud aval en utilisant les méthodes génériques
vTransfo= new Vector[nBranches+1];
vPoints= new Vector[nBranches+1];
setupVector(depletedAirProcess, depletedAirPoint, 0, depletedAirFlow, Tmemb, mcmi.Pair, 1);
setupVector(flueGasProcess, flueGasPoint, 1, burntGasFlow, T_burntGas, gasP, 1);
setupVector(mainProcess, linkPoint, 2, burntGasFlow, T_burntGas, gasP, 1);
updateDivider(vTransfo,vPoints,T_burntGas,Houtlet);

de.updateProcess(setEnergyTypes(mainProcess,0,PCI*fuelFlow,0));//DeltaH énergie payante


//mise à jour du noeud amont en utilisant les méthodes génériques
vTransfo= new Vector[mcmi.nBranches+1];
vPoints= new Vector[mcmi.nBranches+1];
setupVector(mcmi.combustProcess, mcmi.combustPoint, 0, fuelFlow, mcmi.Tfuel, mcmi.Pfuel, 1);
setupVector(mcmi.airProcess, mcmi.airProcess, 1, mcmi.airFlow, mcmi.Tair, mcmi.Pair, 1);
setupVector(mainProcess, linkPoint, 2, burntGasFlow, T_burntGas, gasP, 1);
mcmi.updateMixer(vTransfo,vPoints,T_burntGas,Houtlet);
```


cycle CLC
combustible
air
entrée air
compresseur
CH4 ` méthane
CLC reactor inlet
CLC reactor
MCM reactor
CLC reactor outlet
air appauvri
HTT
fumees
sortie air appauvri
HRSG 2
HRSG 1
HRSG 0
économiseur
vaporiseur
surchauffeur
économiseur 2
vaporiseur 2
surchauffeur 2
condenseur
pompe alimentaire
div
mel
HPT
LPT
recuperation
réchauffe
recycl refroidi
recyclage
séparateur
refroidissement
HRSG fumées 2
HRSG fumées 1
HRSG fumées 0
sortie CO2
sortie eau
Bilan
puissance utile : 31 669,58 (kW)
puissance payante : 56 379,42 (kW)
efficacité : 0,562