Modeling of a direct contact cooling tower in Thermoptim

A direct contact cooling tower has the distinction of being crossed by two separate streams: air and water, that exchange matter and energy through an interface. It behaves like a quadrupole receiving two input fluid, and of out which come two others (Figure 1).

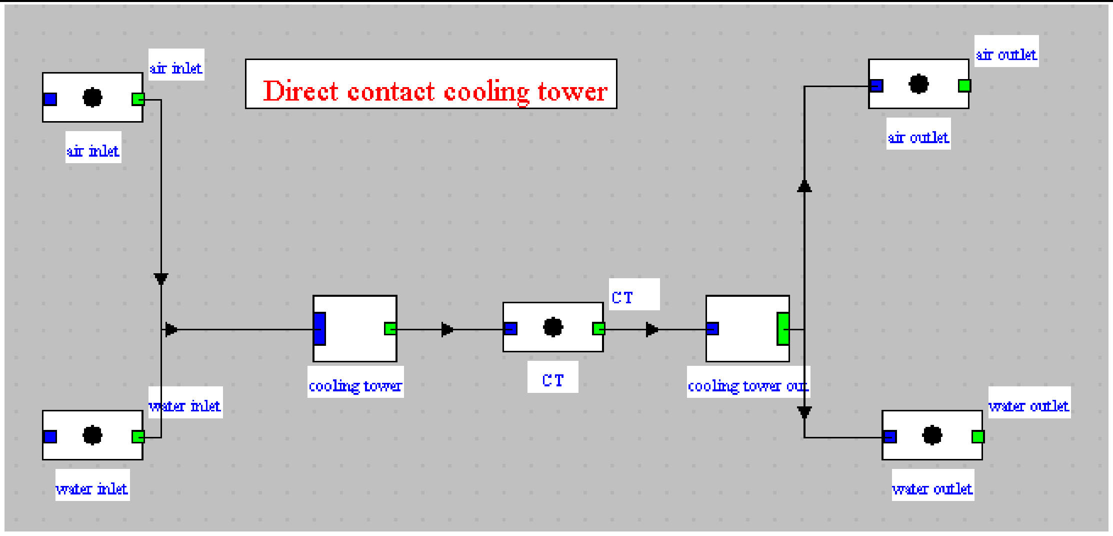


Figure 1: Diagram of the direct contact cooling tower component

Model of the direct contact cooling tower

In these conditions, we can reason on the overall enthalpy level knowing the inlet and outlet air side conditions and the inlet ones on the water side: model results are consistent with experimental values and those supplied by manufacturers.

Flow rates of both inlet streams are set by conditions upstream of the component and not recalculated. If water flow is insufficient for its cooling (up to moist bulb temperature of incoming air) to provide air with the enthalpy required, a message warns the user.

The functions for calculating properties of Thermoptim moist gases and points have been made available from external classes. We advise you to refer to the note: "Calculations of wet gas from external classes[1]"for a detailed presentation of the available methods.

Recall that, as usual in calculations of moist functions, values are referred to dry gas, whose composition is invariant, while other calculations in Thermoptim are relative to the actual gas composition, i.e. are referred to the moist gas. It is therefore necessary to make the corresponding conversions.

Another point to mention is that when you save a point properties, moisture is not taken into account because it is derived from the gas composition. If we desire to save a change in the humidity of the air inlet, we must also change the gas composition from the moist screen calculations of the inlet point.

The model that we can choose, when we know the exit air temperature, is the following:

1) we begin by calculating inlet moist air properties, and determining the dry gas mass flow from that of moist gas; inlet relative humidity ε is displayed on the screen;
2) the outlet relative humidity is set equal to 1, and the outlet moist air properties are calculated, which gives the specific and total enthalpy to be brought to air;
3) the flow of water carried by air is determined and the outlet moist air composition is changed;

---

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

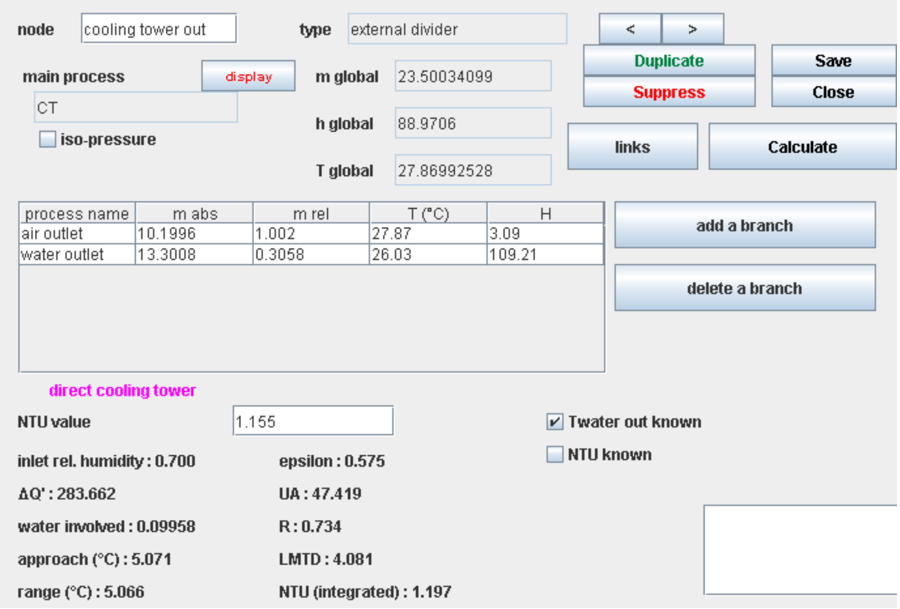
4) the water enthalpy balance provides its outlet temperature, which must exceed the moist bulb temperature of air entering;

5) values downstream the node are updated;

6) NTU is calculated by integrating the Merkel equation (NTU integrated) and on the basis of the mean logarithmic enthalpies; effectiveness can be deduced from the classical relationship for counter-flow heat exchangers.

The tower is as we said represented by an external mixer connected to an external divider, calculations being made by the latter. Classes are called DirectCoolingTowerInlet and DirectCoolingTower.

The cooling tower component screen is given in Figure 2. We wish here to cool 1kg/s of water from 30 °C to 25 °C. With a rate of 0.9 kg/s of air at 18 °C and relative humidity equal to 0.5, the exhaust air temperature is 20.1 °C and 7.5 g/s water are evaporated. The tower capacity is 20.9 kW.

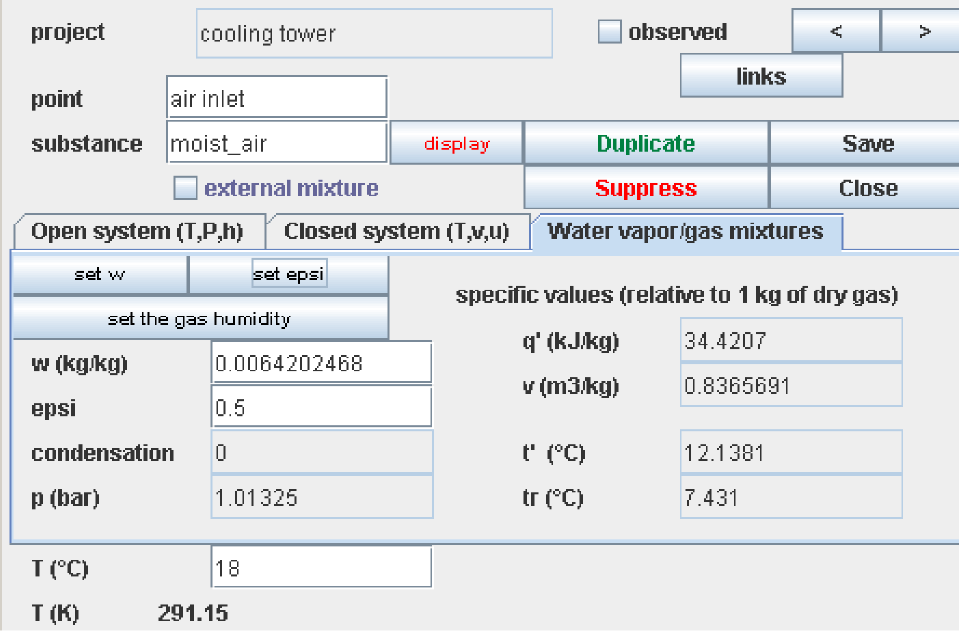


node: cooling tower out
type: external divider
main process: CT
☐ iso-pressure
m global: 23.50034099
h global: 88.9706
T global: 27.86992528

process name	m abs	m rel	T (°C)	H
air outlet	10.1996	1.002	27.87	3.09
water outlet	13.3008	0.3058	26.03	109.21

NTU value: 1.155
☑ Twater out known
☐ NTU known
inlet rel. humidity : 0.700
epsilon : 0.575
ΔQ' : 283.662
UA : 47.419
water involved : 0.09958
R : 0.734
approach (°C) : 5.071
LMTD : 4.081
range (°C) : 5.066
NTU (integrated) : 1.197

Figure 2: Cooling tower component screen

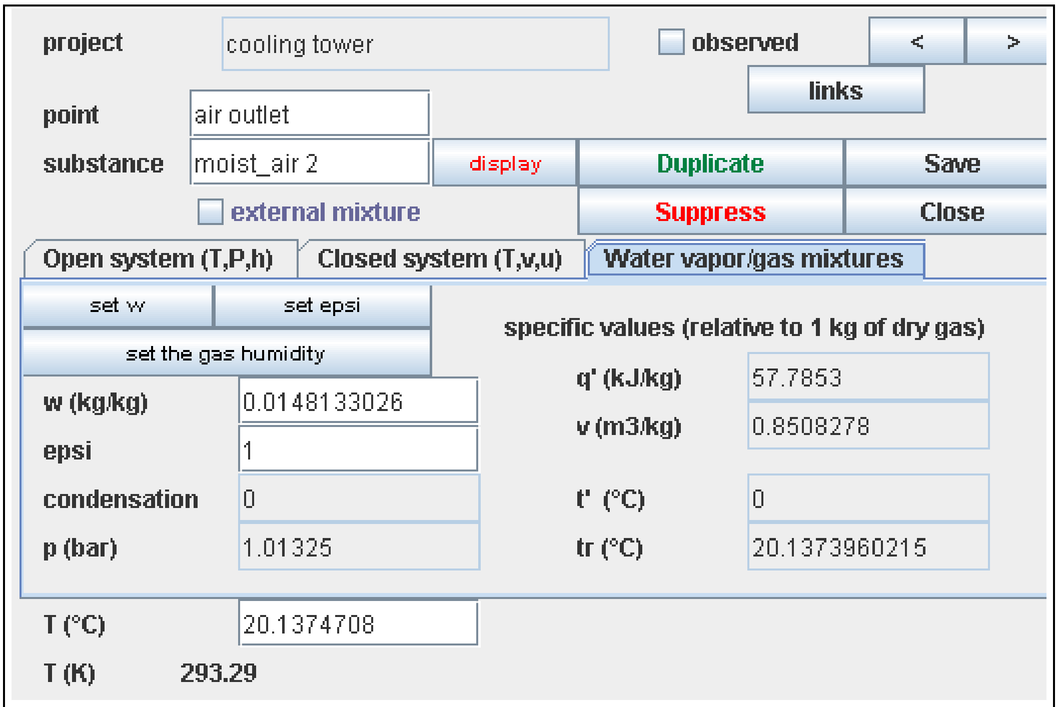


project: cooling tower
☐ observed
point: air inlet
substance: moist_air
☐ external mixture
w (kg/kg): 0.0064202468
epsi: 0.5
condensation: 0
p (bar): 1.01325
q' (kJ/kg): 34.4207
v (m3/kg): 0.8365691
t' (°C): 12.1381
tr (°C): 7.431
T (°C): 18
T (K): 291.15

Figure 3: Inlet point screen

On this screen we see two values of NTU. That which is in the editable field is calculated by the approximate formula based on the logarithmic mean differences of enthalpies of saturated air between the tower inlet and outlet, while the one displayed at the bottom is the result of the integration of the Merkel differential equation for a counter-flow tower. It is more accurate, but you can see that their values are close, so that we have not chosen it in this model to keep it simple in terms of thermodynamics. This simplification allows us in addition to build a model that calculates in direct (estimate NTU) as well as indirect mode (NTU known).

Figures 3 and 4 provide the properties of moist air entering and leaving the tower, and Figure 5 is a synoptic view of the model.



project: cooling tower

☐ observed

point: air outlet

substance: moist_air 2

☐ external mixture

specific values (relative to 1 kg of dry gas)

w (kg/kg)	0.0148133026	q' (kJ/kg)	57.7853
epsi	1	v (m3/kg)	0.8508278
condensation	0	t' (°C)	0
p (bar)	1.01325	tr (°C)	20.1373960215

T (°C): 20.1374708

T (K) 293.29

Figure 4: Outlet point screen

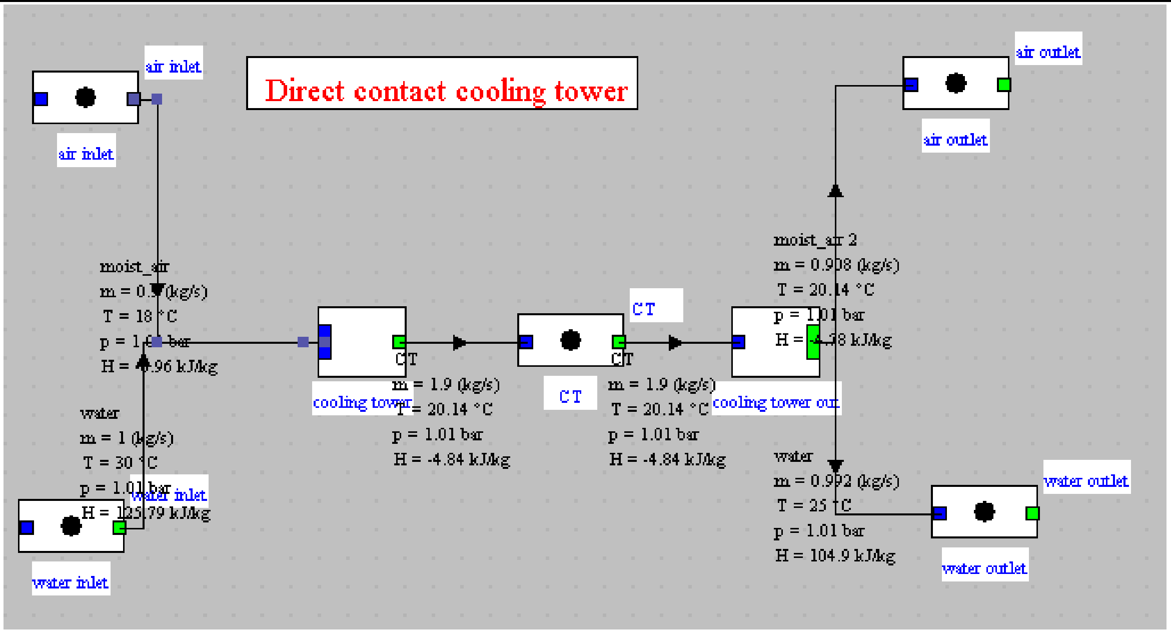


Figure 5: Synoptic view

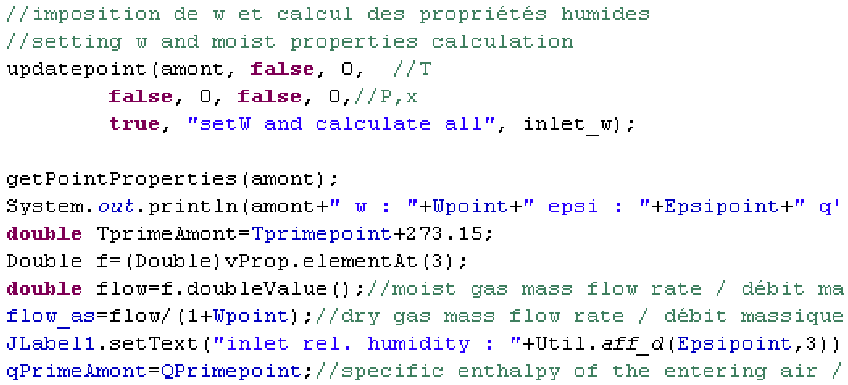
Study of external class DirectCoolingTower

To ensure model consistency (avoiding that the inlet mixer is connected to an inadequate outlet divider), each of the two nodes tries to instantiate the other in its class from the project external components and verifies that both are connected to the same process-point. If the operation fails, a message warns the user that the construction is incorrect. This is checked by methods setupOutlet() and setupInlet().

In addition, consistency tests for each node are carried out by method checkConsistency() to verify that fluids are appropriately connected: in this case, humid air as well as inlet and outlet water. Refer to Volume 3 of the reference manual for explanations on this point, valid for all external nodes.

The study of external class DirectCoolingTower allows one to understand how the model has been implemented. As can be seen, if the outlet air temperature is known, six steps are enough to perform calculations (in other cases, the approach is analogous):

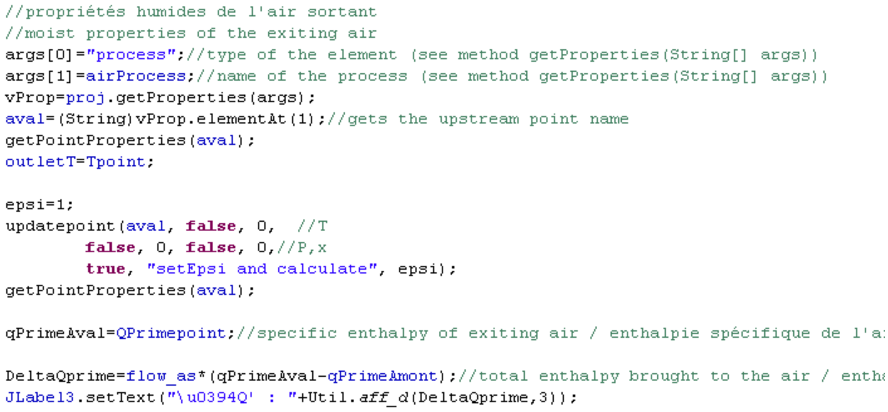
1) we begin by calculating the inlet humid air properties thanks to generic method updatepoint(), then we initialize the dry air flow-rate and the upstream specific enthalpy:



```
//imposition de w et calcul des propriétés humides
//setting w and moist properties calculation
updatepoint(amont, false, 0,  //T
        false, 0, false, 0,//P,x
        true, "setW and calculate all", inlet_w);

getPointProperties(amont);
System.out.println(amont+" w : "+Wpoint+" epsi : "+Epsipoint+" q'
double TprimeAmont=Tprimepoint+273.15;
Double f=(Double)vProp.elementAt(3);
double flow=f.doubleValue();//moist gas mass flow rate / débit ma
flow_as=flow/(1+Wpoint);//dry gas mass flow rate / débit massique
JLabel1.setText("inlet rel. humidity : "+Util.aff_d(Epsipoint,3))
qPrimeAmont=QPrimepoint;//specific enthalpy of the entering air /
```

2) we then calculate the moist air properties and deduce the total enthalpy into play:



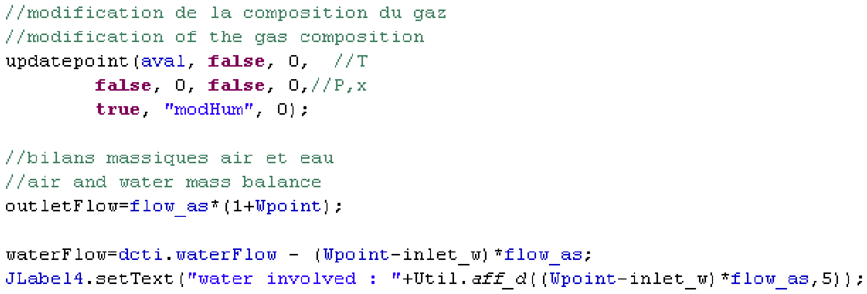
```
//propriétés humides de l'air sortant
//moist properties of the exiting air
args[0]="process";//type of the element (see method getProperties(String[] args))
args[1]=airProcess;//name of the process (see method getProperties(String[] args))
vProp=proj.getProperties(args);
aval=(String)vProp.elementAt(1);//gets the upstream point name
getPointProperties(aval);
outletT=Tpoint;

epsi=1;
updatepoint(aval, false, 0,  //T
        false, 0, false, 0,//P,x
        true, "setEpsi and calculate", epsi);
getPointProperties(aval);

qPrimeAval=QPrimepoint;//specific enthalpy of exiting air / enthalpie spécifique de l'a

DeltaQprime=flow_as*(qPrimeAval-qPrimeAmont);//total enthalpy brought to the air / enth
JLabel3.setText("\u0394Q' : "+Util.aff_d(DeltaQprime,3));
```

3) we change the outlet moist air composition, and calculate the mass flow rates exiting:

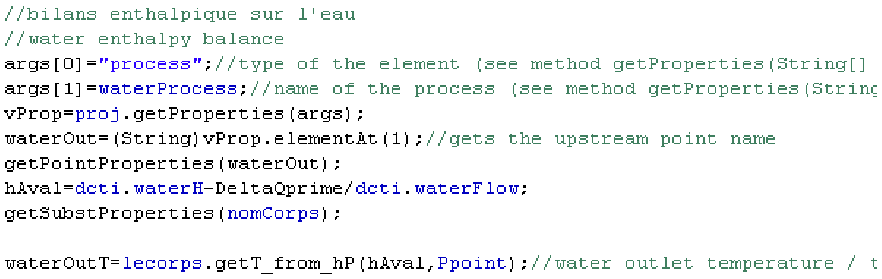


```
//modification de la composition du gaz
//modification of the gas composition
updatepoint(aval, false, 0,  //T
        false, 0, false, 0,//P,x
        true, "modHum", 0);

//bilans massiques air et eau
//air and water mass balance
outletFlow=flow_as*(1+Wpoint);

waterFlow=dcti.waterFlow - (Wpoint-inlet_w)*flow_as;
JLabel4.setText("water involved : "+Util.aff_d((Wpoint-inlet_w)*flow_as,5));
```

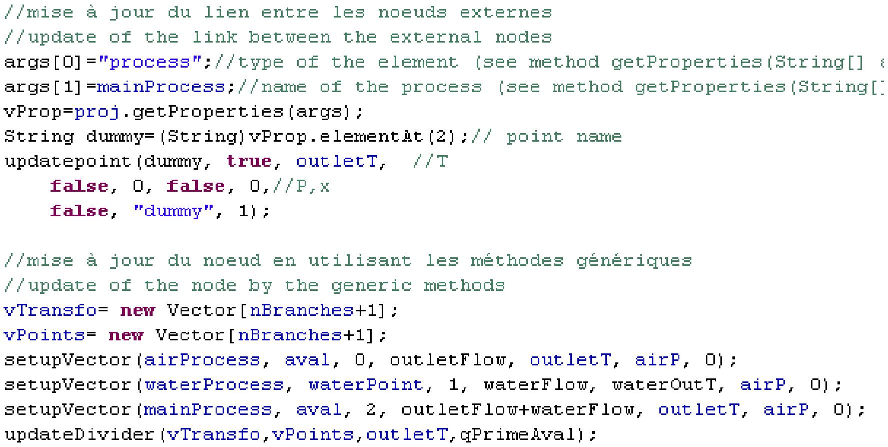
4) the water enthalpy balance then provides the outlet water temperature:



```
//bilans enthalpique sur l'eau
//water enthalpy balance
args[0]="process";//type of the element (see method getProperties(String[]
args[1]=waterProcess;//name of the process (see method getProperties(String
vProp=proj.getProperties(args);
waterOut=(String)vProp.elementAt(1);//gets the upstream point name
getPointProperties(waterOut);
hAval=dcti.waterH-DeltaQprime/dcti.waterFlow;
getSubstProperties(nomCorps);

waterOutT=lecorps.getT_from_hP(hAval,Ppoint);//water outlet temperature / t
```

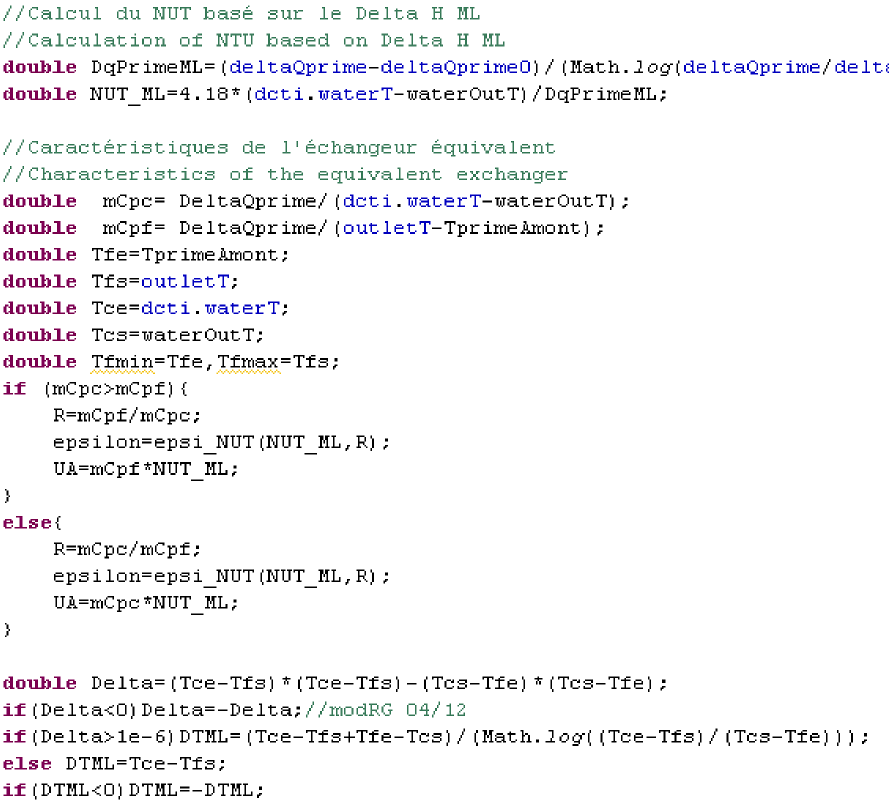
5) the node is updated using generic methods described in the reference manual:



```
//mise à jour du lien entre les noeuds externes
//update of the link between the external nodes
args[0]="process";//type of the element (see method getProperties(String[]
args[1]=mainProcess;//name of the process (see method getProperties(String[
vProp=proj.getProperties(args);
String dummy=(String)vProp.elementAt(2);// point name
updatepoint(dummy, true, outletT,  //T
    false, 0, false, 0,//P,x
    false, "dummy", 1);

//mise à jour du noeud en utilisant les méthodes génériques
//update of the node by the generic methods
vTransfo= new Vector[nBranches+1];
vPoints= new Vector[nBranches+1];
setupVector(airProcess, aval, 0, outletFlow, outletT, airP, 0);
setupVector(waterProcess, waterPoint, 1, waterFlow, waterOutT, airP, 0);
setupVector(mainProcess, aval, 2, outletFlow+waterFlow, outletT, airP, 0);
updateDivider(vTransfo,vPoints,outletT,qPrimeAval);
```

6) finally, the heat exchanger overall characteristics are estimated:



```
//Calcul du NUT basé sur le Delta H ML
//Calculation of NTU based on Delta H ML
double DqPrimeML=(deltaQprime-deltaQprime0)/(Math.log(deltaQprime/delt
double NUT_ML=4.18*(dcti.waterT-waterOutT)/DqPrimeML;

//Caractéristiques de l'échangeur équivalent
//Characteristics of the equivalent exchanger
double  mCpc= DeltaQprime/(dcti.waterT-waterOutT);
double  mCpf= DeltaQprime/(outletT-TprimeAmont);
double Tfe=TprimeAmont;
double Tfs=outletT;
double Tce=dcti.waterT;
double Tcs=waterOutT;
double Tfmin=Tfe,Tfmax=Tfs;
if (mCpc>mCpf){
    R=mCpf/mCpc;
    epsilon=epsi_NUT(NUT_ML,R);
    UA=mCpf*NUT_ML;
}
else{
    R=mCpc/mCpf;
    epsilon=epsi_NUT(NUT_ML,R);
    UA=mCpc*NUT_ML;
}

double Delta=(Tce-Tfs)*(Tce-Tfs)-(Tcs-Tfe)*(Tcs-Tfe);
if(Delta<0)Delta=-Delta;//modRG 04/12
if(Delta>1e-6)DTML=(Tce-Tfs+Tfe-Tcs)/(Math.log((Tce-Tfs)/(Tcs-Tfe)));
else DTML=Tce-Tfs;
if(DTML<0)DTML=-DTML;
```