Modeling of an indirect contact cooling tower in Thermoptim

An indirect 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).

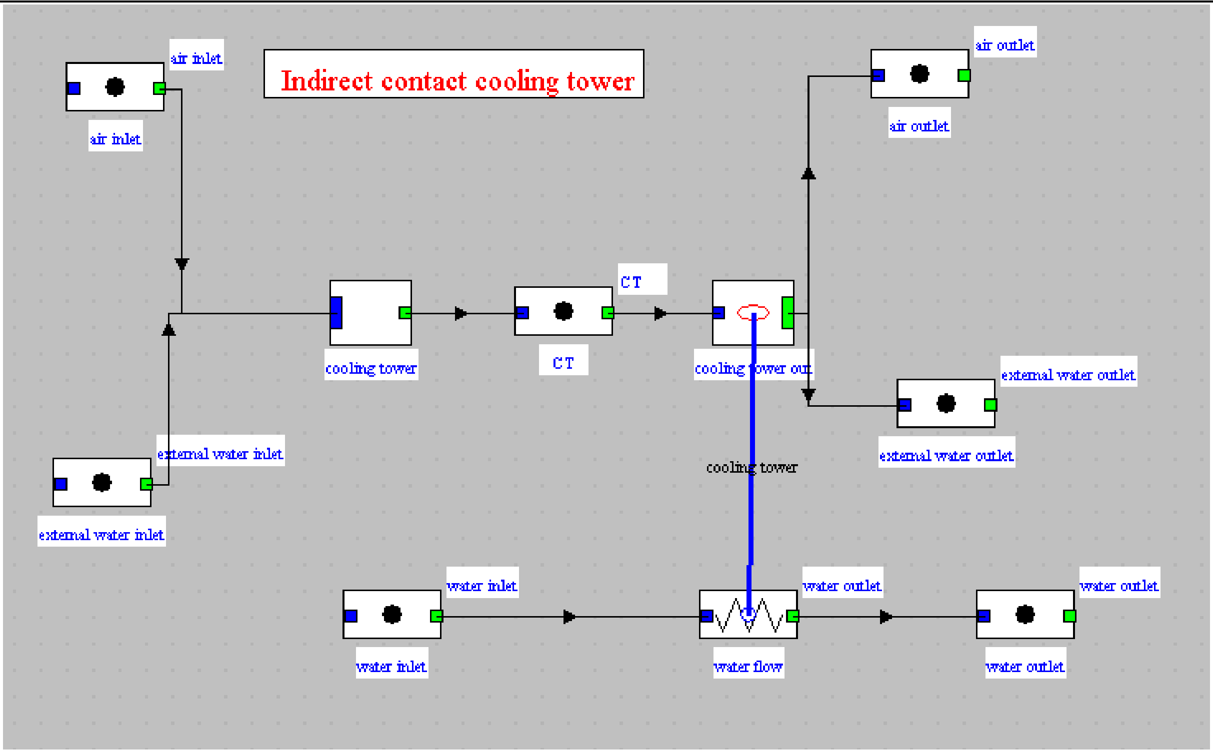

Figure 1: Diagram of the indirect contact cooling tower component

Model of the indirect contact cooling tower

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;

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

2) the outlet relative humidity is read on the screen, 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;
4) the water enthalpy balance provides the thermocoupler load, which gives the outlet temperature of the fluid to be cooled;
5) values downstream the node are updated.

This model is very simple and does not include any estimate of the size of the tower, such as calculation of a NTU. It would be possible to complete it on this point, similarly to what is presented in the direct contact cooling tower model.

As indicated above, the tower is represented by an external mixer connected to an external divider, the calculations being made by the latter. The classes are called InDirectCoolingTowerInlet and InDirectCoolingTowerOutlet.

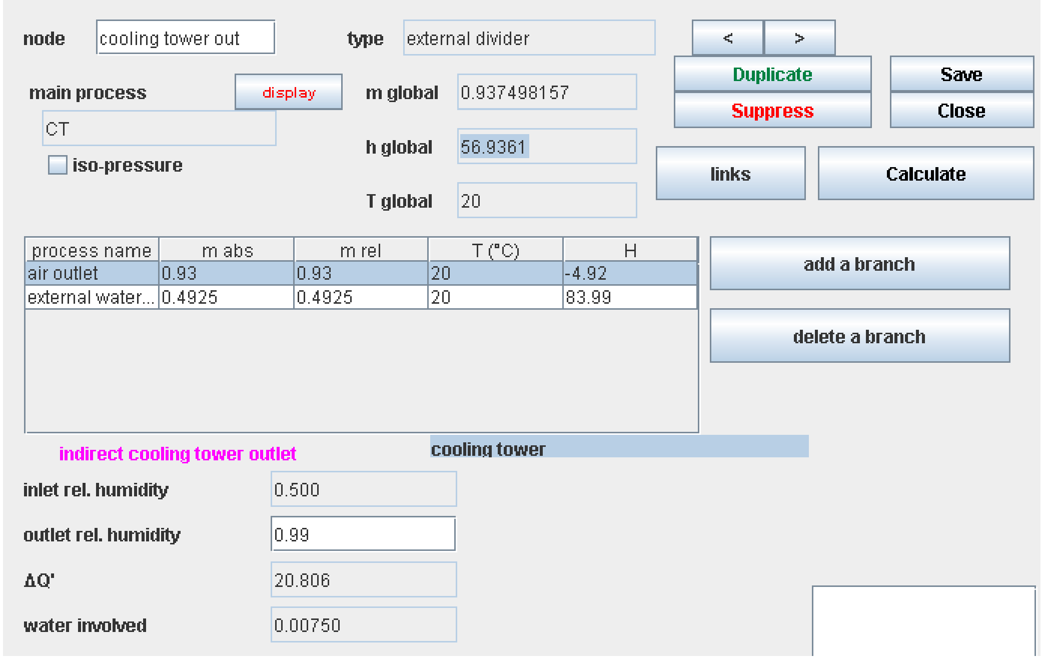

Figure 2: Cooling tower component screen

The cooling tower component and thermocoupler screens are given in Figures 2 and 3. We wish here to cool 1kg/s of water from 30 °C to 25 °C. With a rate of 0.93 kg/s of air at 18 °C and relative humidity equal to 0.5, the exhaust air temperature is 20 °C and 7.5 g/s of water are evaporated. The tower capacity is 20.8 kW.

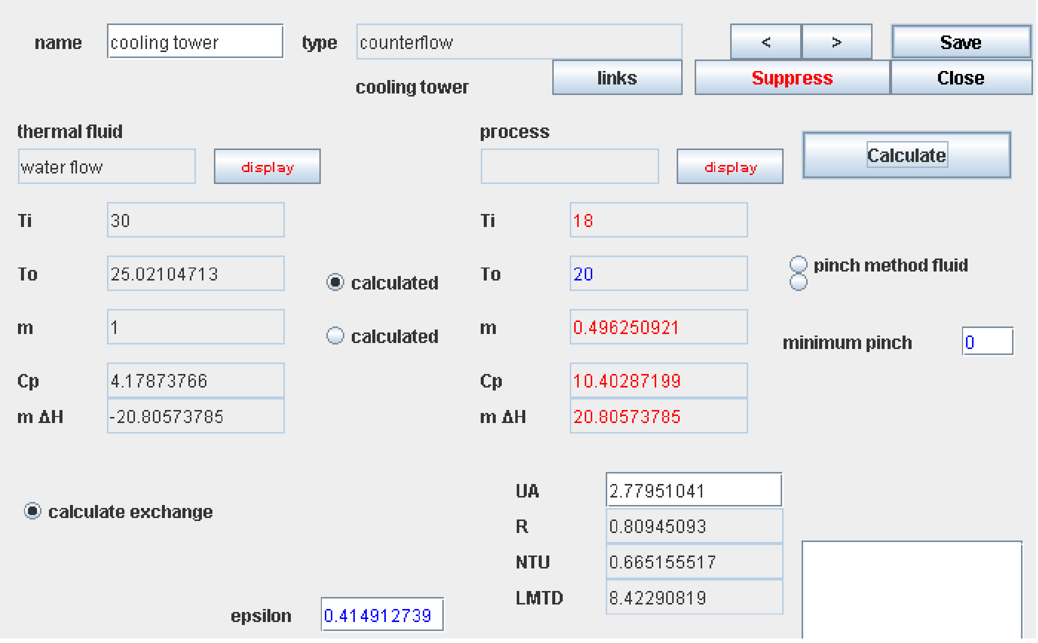

Figure 3: Thermocoupler screen

Figures 4 and 5 provide the properties of moist air entering and leaving the tower, and Figure 6 is a synoptic view of the model.

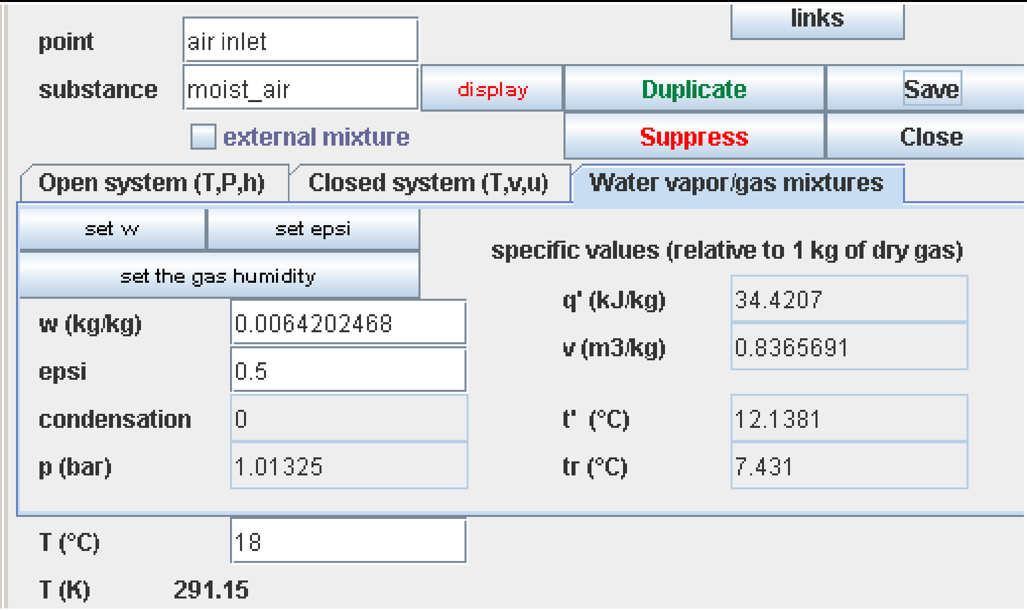

Figure 4: Inlet point screen

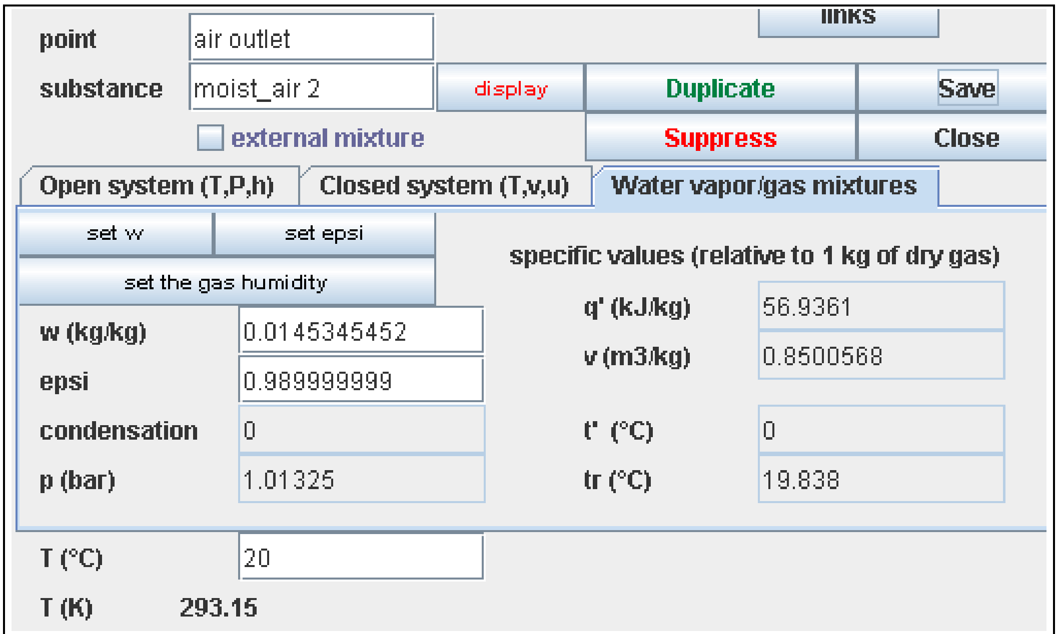

Figure 5: Outlet point screen

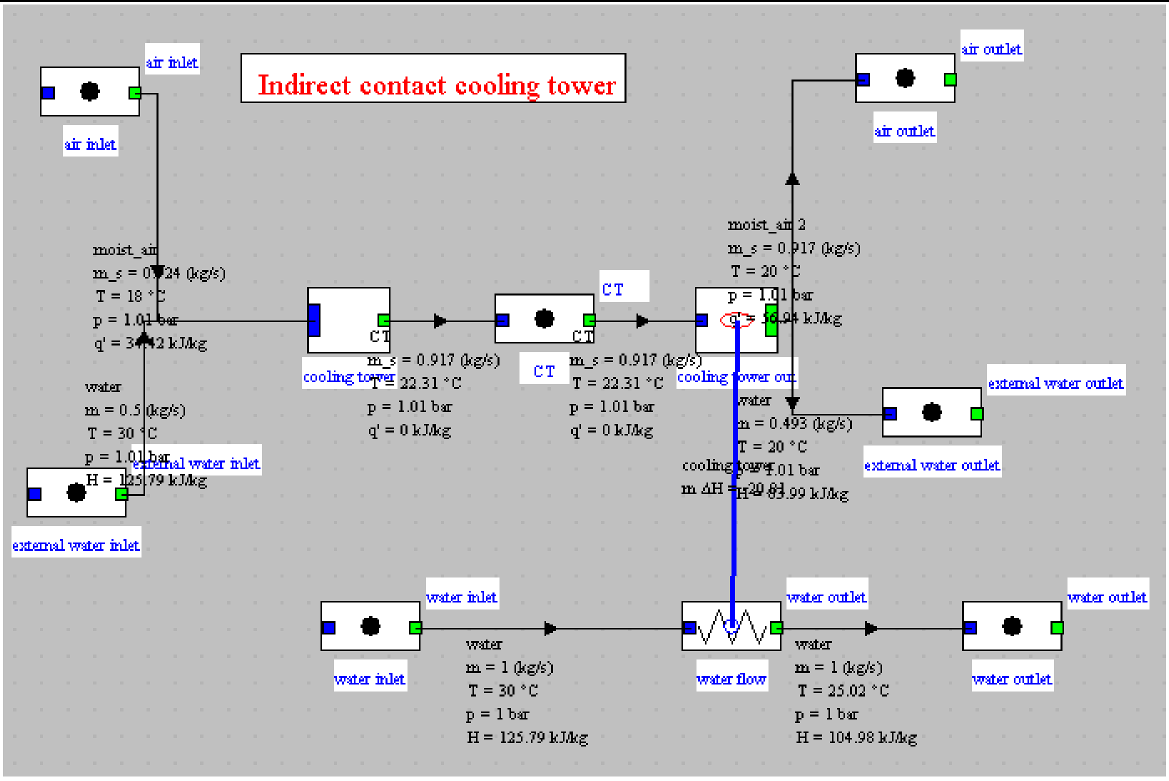

Figure 6: Synoptic view

Study of external class InDirectCoolingTowerOutlet

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 InDirectCoolingTowerOutlet 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", old_w);

getPointProperties(amont);
System.out.println(amont+" w : "+Wpoint+" epsi : "+Epsipoint+" q' : "+QPrimepoint+" t' : "+Tprimepoi

Double f=(Double)vProp.elementAt(3);
double flow=f.doubleValue();
double flow_as=flow/(1+Wpoint);//débit massique de gaz sec
```

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);
String aval=(String)vProp.elementAt(1);//gets the upstream point name
getPointProperties(aval);
outletT=Tpoint;

double epsi=Util.lit_d(outletEpsi_value.getText());
//imposition de epsilon et mise à jour de w
updatepoint(aval, false, 0,  //T
        false, 0, false, 0,//P,x
        true, "setEpsi and calculate", epsi);
getPointProperties(aval);

double haval=QPrimepoint;//enthalpie spécifique de l'air sortant

double DeltaQprime=flow_as*(haval-hamont);//enthalpie totale acquise par l'air
DeltaQprime_value.setText(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
double outletFlow=flow_as*(1+Wpoint);

double waterFlow=(Wpoint-old_w)*flow_as;
water_value.setText(Util.aff_d(waterFlow,5));

approach_value.setText(Util.aff_d(waterT-outletT,3));
```

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

```
//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, airPoint, 0, flow, airT, airP, 0);
setupVector(waterProcess, waterPoint, 1, icti.waterFlow-waterFlow, waterT, airP, 0);
setupVector(mainProcess, aval, 2, outletFlow, outletT, airP, 0);
updateDivider(vTransfo,vPoints,outletT,haval);
```