Updating and calculating the process

Here we refer to the example of the class SolarCollector, whose model is presented in the note "SolarCollector.doc"

The sequence of operations is as follows:

1) update the component before calculation by loading the values of the process and the upstream point
2) read the parameters on the external component screen
3) calculate the power used and the state of the downstream point
4) calculate the thermal loads of the thermocouplers
5) update the external component screen

Let us now look at the practical problems encountered during each of these steps. A few portions of the code are provided below, but we recommend reading the rest of this note while referring to the entire class SolarCollector.java.

1) update the component by loading the values of the process and the upstream point

The problem here is that an external component does not have direct access to the simulator variables: these values are obtained by very general methods, which construct Vectors with different structures depending on the desired object.

The procedure is not complicated, but it must be followed carefully:

```
String[] args=new String[2]; //array of arguments
args[0]="process";//type of element desired (a process in this case)
args[1]=tfe.getCompName();//name of the process (obtained by the reference tfe)
Vector vProp=proj.getProperties(args);//Project method given in Appendix 2
Double f=(Double)vProp.elementAt(3);
double flow=f.doubleValue();//flow rate value, automatically propagated from the upstream process
String amont=(String)vProp.elementAt(1);//name of the upstream point
getPointProperties(amont);//automatic decoding of the Vector (method of the ExtProcess class)
Tamont=Tpoint;//here T1
```

The method getPointProperties() of ExtProcess automatically loads the state of a point in easily manipulable values, called Tpoint, Point, lecorps… The method is given below.

```
public void getPointProperties(String nom){
        String[] args=new String[2];
        args[0]="point";//type of the element (see method getProperties(String[] args))
        args[1]=nom;//name of the process (see method getProperties(String[] args))
        Vector vProp=proj.getProperties(args);
   lecorps=(Corps)vProp.elementAt(0);
   nomCorps=(String)vProp.elementAt(1);
   Double y=(Double)vProp.elementAt(2);
   Tpoint=y.doubleValue();
   y=(Double)vProp.elementAt(3);
        Ppoint=y.doubleValue();
   y=(Double)vProp.elementAt(4);
        Xpoint=y.doubleValue();
   y=(Double)vProp.elementAt(5);
        Vpoint=y.doubleValue();
   y=(Double)vProp.elementAt(6);
        Upoint=y.doubleValue();
   y=(Double)vProp.elementAt(7);
        Hpoint=y.doubleValue();
   y=(Double)vProp.elementAt(9);
        DTsatpoint=y.doubleValue();
```

```
        String dum=(String)vProp.elementAt(8);
        isTsatSet=Util.lit_b(Util.extr_value(dum));
        dum=(String)vProp.elementAt(10);
        isPsatSet=Util.lit_b(Util.extr_value(dum));
}
```

2) read the parameters on the external component screen

The package extThopt provides a number of simple but robust methods for converting the Strings displayed in the JTextField fields used on the graphic interface to doubles, and vice versa for displaying the doubles in these fields. They are implemented as static methods of the extThopt.Util class (see Appendix 3):

```
P=Util.lit_d(P_value.getText());
A=Util.lit_d(A_value.getText());
tau=Util.lit_d(tau_value.getText());
K=Util.lit_d(K_value.getText());
Tex=Util.lit_d(Tex_value.getText())+273.15;
```

3) calculate the power used and the state of the downstream point

We begin by estimating the Cp of the heat-conducting fluid by making a limited development of the enthalpy function, which means we use the method CalcPropCorps() of the package Corps and the method getSubstProperties() of ExtProcess, which automatically loads the state of a point in easily manipulable values, called Tsubst, Psubst, etc. :

```
public void getSubstProperties(String nom){
        String[] args=new String[2];
        args[0]="subst";//type of the element (see method getProperties(String[] args))
        args[1]=nom;//name of the process (see method getProperties(String[] args))
        Vector vProp=proj.getProperties(args);
   Double y=(Double)vProp.elementAt(0);
   Tsubst=y.doubleValue();//temperature
   y=(Double)vProp.elementAt(1);
        Psubst=y.doubleValue();//pressure
   y=(Double)vProp.elementAt(2);
        Xsubst=y.doubleValue();//value
   y=(Double)vProp.elementAt(3);
        Vsubst=y.doubleValue();//mass volume
   y=(Double)vProp.elementAt(4);
        Usubst=y.doubleValue();//internal mass energy
   y=(Double)vProp.elementAt(5);
        Hsubst=y.doubleValue();//mass enthalpy
   y=(Double)vProp.elementAt(6);
        Ssubst=y.doubleValue();mass entropy
y=(Double)vProp.elementAt(7);
      Msubst=y.doubleValue();//molar mass
      Integer i=(Integer)vProp.elementAt(8);
      typeSubst=i.intValue();//type of substance (1 for water, 2 for a vapor, 3 for a pure gas, 4 for a compound
gas, 5 for an external substance)
 }
```

Then we can calculate the Cp as follows:

```
double H=Hpoint;//enthalpy of the upstream point
lecorps.CalcPropCorps(Tpoint+1, Ppoint, Xpoint);// recalculates the upstream substance by a Thermoptim
function
getSubstProperties(nomCorps);//retrieves the recalculation values (method of the ExtSubstance class)
double Cp=(Hsubst-H);//estimated value of Cp
```

We then calculate an estimated Taval value to be able to determine the absorbed thermal power Qex:

```
double DT0=tau*P/K-Tamont+Tex;
double T=Tamont+(DT0)*(1-Math.exp(-K*A/flow/Cp));
double DT=T-Tpoint;
double Qex=Cp*DT*flow;
```

We determine the value of the mass enthalpy of the downstream point, then we invert this equation to determine the exact value of Taval (method of the Corps class)

```
double hAval=Qex/flow+Hpoint;
Tpoint=lecorps.getT_from_hP(hAval,Ppoint);
```

```
getSubstProperties(nomCorps);//retrieves the recalculation values (method of the ExtProcess class)
Xpoint=Xsubst;//updates the value of the vapor quality in case the state is diphasic
```

4) calculate the thermal loads of the thermocouplers

In this simple example, there is no problem, since the component does not use a thermocoupler. Brief instructions are given later in the manual, as well as in the section on external nodes.

5) update the external component screen

The Thermoptim method setupPointAval() updates the downstream point from the values loaded in a Vector constructed here by the method getProperties() of ExtProcess:

```
tfe.setupPointAval(getProperties());
```

The solar collector yield value is then determined and displayed.

```
eff_value.setText(Util.aff_d(Qex/P/A, 4));
```

For this example, the updates before and after recalculation are very simple. In other cases, it may be necessary to access other data from the simulator. We will explain how to do this below.