Driver modeling a cooled air compressor

To illustrate the ability of Thermoptim to perform off-design calculations, we will study the behavior of an air compressor that fills a compressed air storage of given volume at variable pressure. The compressed air is cooled before storage thanks to a water exchanger.

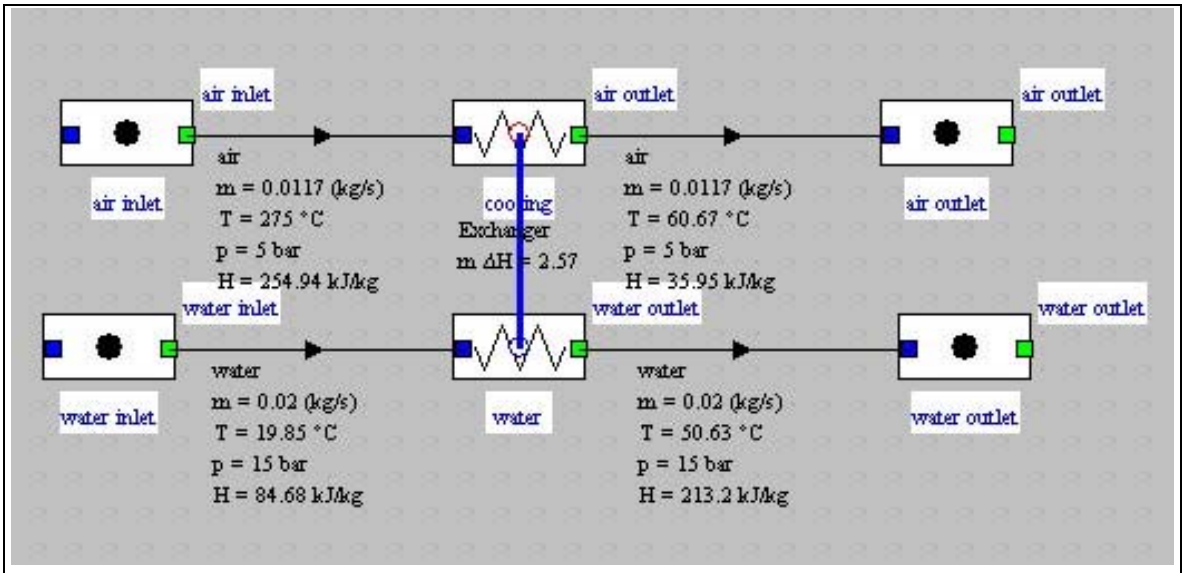

Figure 1: Example of a cooled compressor

The system can be easily modeled in Thermoptim and leads to a diagram such as in Figure 1.

The compressor technological design screen is given in Figure 2. The exchanger is given in Figure 3.

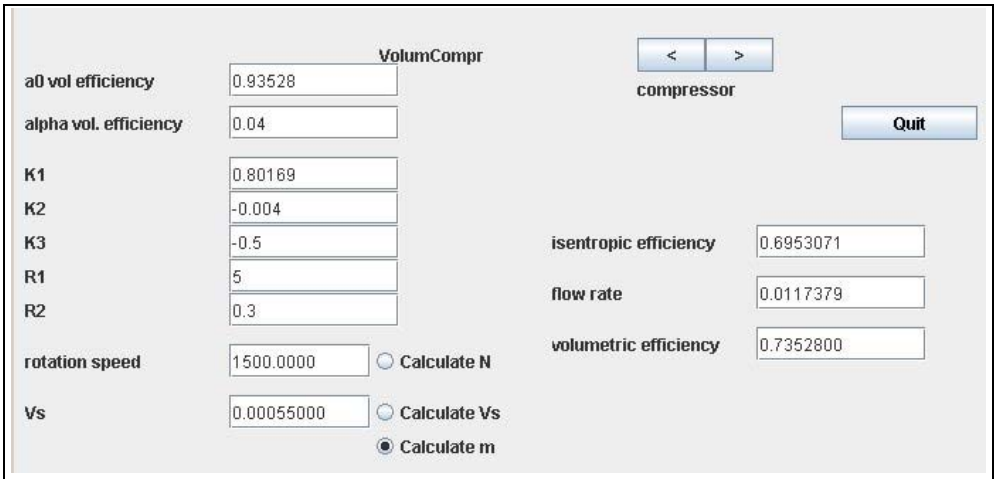

Figure 2: Compressor technological design screen

Note that double-clicking the component name (here "compressor" located under the small arrows at the top right) provides access to its classic screen in the simulator.

The setting of these technological screens is explained in various pages of the Thermoptim-UNIT portal [1].

[1] http://www.thermoptim.org/sections/technologies/composants/compresseurs
http://www.thermoptim.org/sections/technologies/composants/echangeurs
http://www.thermoptim.org/sections/base-methodologique/dimensionnement/exemples-dtnn
http://www.thermoptim.org/sections/enseignement/pedagogie/fils-d-ariane/fil-compr-volum

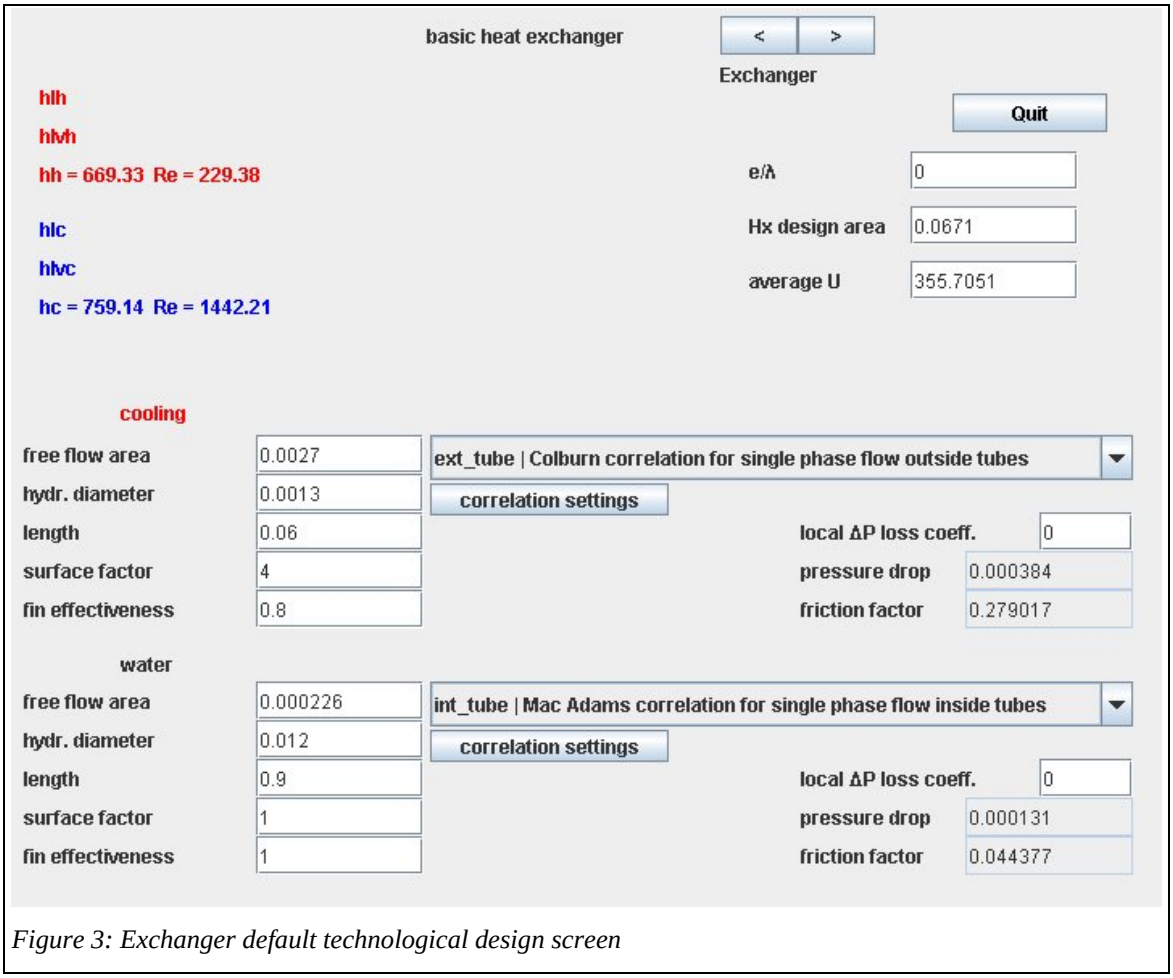

Figure 3: Exchanger default technological design screen

1.1 Results

Once the driver made, it very easy to vary the compression ratio for determine the main variables when the tank pressure varies. They can be exploited with the Excel Thermoptim simulation files post-processing macro presented in Volume 1 of the reference manual. Simply save the project file under a different name after each simulation, then load these files into the macro and extract the values of interest.

The figure 4 shows, depending on the storage pressure, changes in the compressor isentropic efficiency, compression work, intake air flow, temperature of air entering the tank, the exchanger load and UA.

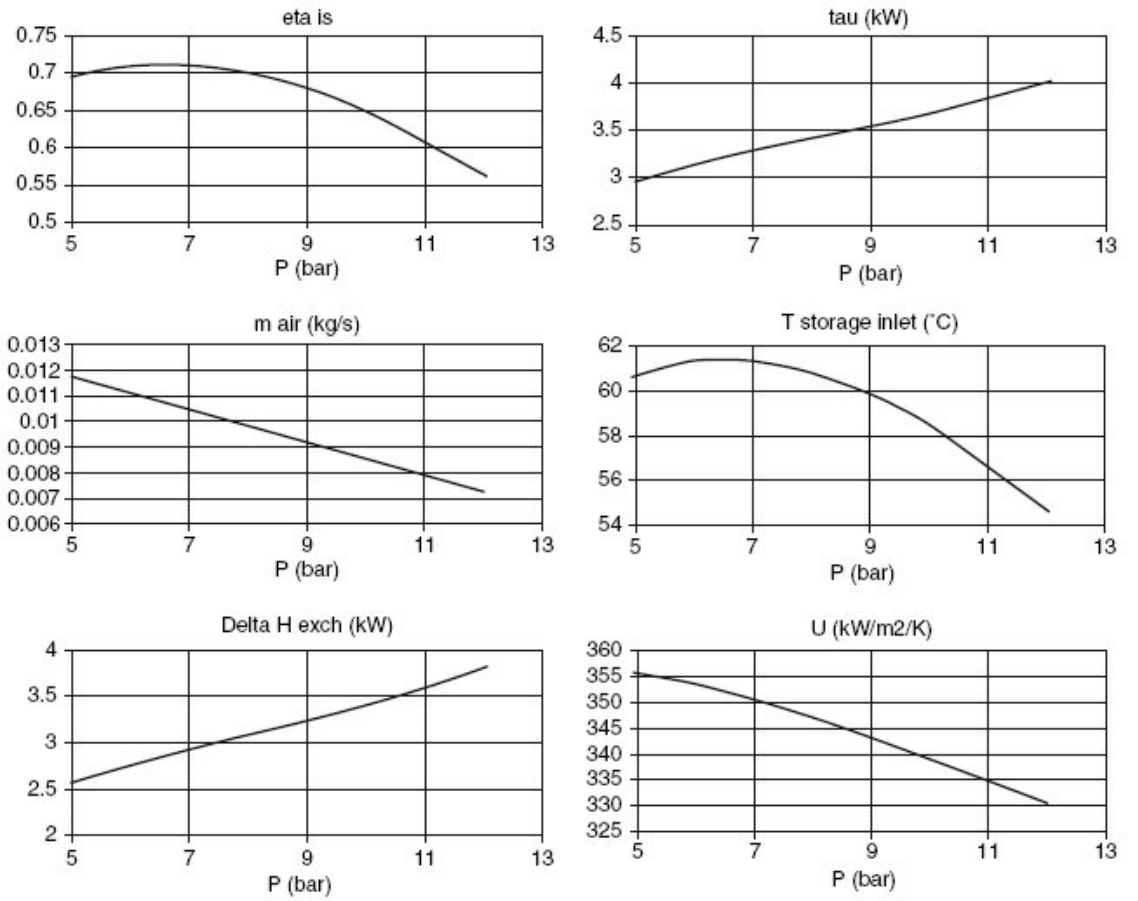

Figure 4: Simulation results

1.2 Driver design

The driver class is called VsCompressorDriver.java. As its graphical interface is simple and classic, we do not detail here.

The driver screen is shown in Figure 5.

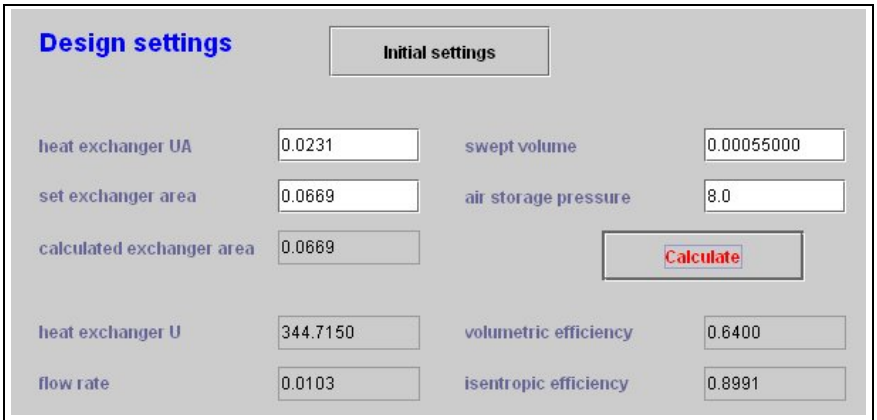

Figure 5: Driver screen

The class operates as follows:
- We first instantiate the technological design screen of the compressor and heat exchanger;
- They are initialized;
- We implement the changes in the calculations and the settings of the simulator that the class should perform when the downstream pressure varies.

1.2.1 Instantiation

```
//initializations for the simulation
//watch out: the names of points and components must be correct, otherwise an error wil
hxName="Exchanger";
compressorName="compressor";
hotDownstream=new PointThopt(proj,"storage inlet");
hotUpstream=new PointThopt(proj,"compressor outlet");
comprUpstream=new PointThopt(proj,"air inlet");
coldUpstream=new PointThopt(proj,"water inlet");
coldDownstream=new PointThopt(proj,"water outlet");

//instantiation of the TechnoDesign in the external classes
technoExchanger=new TechnoHx(proj, hxName, hotUpstream, hotDownstream, coldUpstream, cc
addTechnoVector(technoExchanger);
technoCompr=new VolumCompr(proj, compressorName, comprUpstream, hotUpstream);
addTechnoVector(technoCompr);

//initialization of the TechnoDesign in Thermoptim
setupTechnoDesigns(vTechno);
```

1.2.2 Initialization

During initialization, the technological design screen of the heat exchanger is updated from the values of the simulator, and the surface calculated:

```
if(!hxName.equals("")){//initialisation de l'évaporateur
    args=new String[2];
    args[0]="heatEx";
    args[1]=hxName;
    vProp=proj.getProperties(args);
    Double f=(Double)vProp.elementAt(15);
    UAech=f.doubleValue();
    String fluideChaud=(String)vProp.elementAt(0);
    args[0]="process";
    args[1]=fluideChaud;
    vProp=proj.getProperties(args);
    f=(Double)vProp.elementAt(4);
    DeltaH=-f.doubleValue();

    DeltaHech=DeltaH;
    hotUpstream.getProperties();
    hotDownstream.getProperties();
    coldUpstream.getProperties();
    coldDownstream.getProperties();

    mCpEau=DeltaH/(coldDownstream.T-coldUpstream.T);
    mCpAir=DeltaH/(hotUpstream.T-hotDownstream.T);
    UAech_value.setText(Util.aff_d(UAech,4));

    //initialisations du TechnoDesign
    technoExchanger.UA=UAech;
    technoExchanger.makeDesign();
    AechReel=Util.lit_d(technoExchanger.ADesign_value.getText());
    U_ech=UAech/AechReel;
    AcalculatedEch_value.setText(Util.aff_d(AechReel,4));
    AdesignEch_value.setText(technoExchanger.ADesign_value.getText());
}
```

The displacement of the compressor Vs allowing one to obtain the desired flow rate is then determined, based on the technological design screen setting:

```
if(!compressorName.equals("")){//initialization of the compressor
    args=new String[2];
    args[0]="process";
    args[1]=compressorName;
    vProp=proj.getProperties(args);
    String amont=(String)vProp.elementAt(1);
    String aval=(String)vProp.elementAt(2);
    Double f=(Double)vProp.elementAt(3);
    massFlow=f.doubleValue();
    N_value=Util.lit_d(technoCompr.N_value.getText());
    lambdaVol=technoCompr.getLambdaVol();
    Vs=massFlow*60*comprUpstream.V/N_value/lambdaVol;
    Vs_value.setText(Util.aff_d(Vs,8));
    technoCompr.setVs(Vs);
    technoCompr.setN(N_value);
}
```

1.2.3 Calculations

The calculations made are as follows:

- Start by initializing the displacement and update the compressor output pressure
- Calculate the volumetric and isentropic efficiencies, determine the mass flow into play and propagate it upstream and downstream
- Recalculate the compressor and downstream process, knowing that, as it is set as isobaric pressure, it propagates the new pressure
- Update the inlet and outlet of the heat exchanger and then recalculate it in off-design mode after having updated the heat flow of the air that has been modified
- Update the simulator and recalculate the project several times to ensure the stabilization of values.

The first five stages correspond to the code below:

```
void bCalc_actionPerformed(java.awt.event.ActionEvent event)
{
    Vs=Util.lit_d(Vs_value.getText());//reads the compressor swept volume
    technoCompr.setVs(Vs);
    Preservoir=Util.lit_d(P_value.getText());
    double UA_ech=Util.lit_d(UAech_value.getText());

    hotUpstream.P=Preservoir;// updates the compressor outlet pressure
    hotUpstream.update(!UPDATE_T,UPDATE_P,!UPDATE_X);
    hotUpstream.getProperties();

    massFlow=technoCompr.getMassFlow(comprUpstream.V);
    double eta_is=technoCompr.getRisentr();// calculates compressor isentropic efficiency
    lambdaVol=technoCompr.getLambdaVol();

    //recalculates the compressor and the downstream process
    updateprocess(compressorName, "Compression",RECALCULATE,IS_SET_FLOW, UPDATE_FLOW, massFlo
    hotUpstream.getProperties();
    updateprocess("refroidissement", "Exchange",RECALCULATE,IS_SET_FLOW, UPDATE_FLOW, massFlo
    updateprocess("entree air", "Exchange",RECALCULATE,IS_SET_FLOW, UPDATE_FLOW, massFlow,  U

    hotUpstream.getProperties();//updates heat exchanger inlets and outlets
    hotDownstream.getProperties();
    coldUpstream.getProperties();
    coldDownstream.getProperties();
```

This example illustrates the value of using PointThopt to communicate between the driver and the simulator. The syntax of updates is much more readable than using only the Project methods getProperties() and updatePoint().

The calculation of the compressor is done using the two methods getMassFlow() and getLambdaVol() of the technological design screen. The method updateprocess() changes the flow and the isentropic efficiency of the compressor and then recalculates it. The downstream point, called here " hotUpstream " because it is upstream of the hot fluid exchanger, is then updated.

The calculation of the heat exchanger is made as follows: first we make a calculation using the method updateHx() by setting the value of UA read on the screen, UA_ech (the heat exchanger is set for an off-design calculation, so that it is the value of UA that is taken into account). This calculation is used to initialize the heat exchanger for the new operating conditions.

We then use the method makeDesign() of TechnoDesign, which updates the value of U. The real UA is obtained by multiplying the new U with the initial value of A (AechReel), which allows the exchanger to recalculate correctly.

Since U depends on the average temperatures of the fluids, and therefore those of outlet, we iterate five times the calculations to ensure proper stabilization, resetting each time UA_ech

```
        //calculates the heat exchanger (several iterations)
        for(int i=0;i<5;i++){
            updateHx(hxName, RECALCULATE, UPDATE_UA, UA_ech, !UPDATE_EPSI, 0, !UPDATE_DTMIN, 0, U
            coldDownstream.getProperties();
            hotDownstream.getProperties();

            technoExchanger.UA=UA_ech;//calculates U
            technoExchanger.makeDesign();
            double U=technoExchanger.getU();

            UAech=U*AechReel/1000.;//updates UA and recalculates the heat exchanger
            UAech_value.setText(Util.aff_d(UAech,4));
            updateHx(hxName, RECALCULATE, UPDATE_UA, UAech, !UPDATE_EPSI, 0, !UPDATE_DTMIN, 0, UP

            coldDownstream.getProperties();
            hotDownstream.getProperties();
            U_value.setText(Util.aff_d(U,4));
            UA_ech=UAech;
        }

        //updates the simulator and displays
        for(int j=0;j<3;j++)proj.calcThopt();
        flow_value.setText(Util.aff_d(massFlow,4));
        eta_is_value.setText(Util.aff_d(eta_is,4));
        lambdaVol_value.setText(Util.aff_d(lambdaVol,4));
        AcalculatedEch_value.setText(technoExchanger.ADesign_value.getText());
    }
```