Pilote représentant un compresseur à air refroidi

Pour illustrer la capacité de Thermoptim à effectuer des calculs en régime non-nominal, nous allons étudier le comportement d'un compresseur volumétrique à air qui remplit un stockage d'air comprimé de volume donné à pression variable. L'air comprimé est refroidi avant stockage grâce à un échangeur à eau.

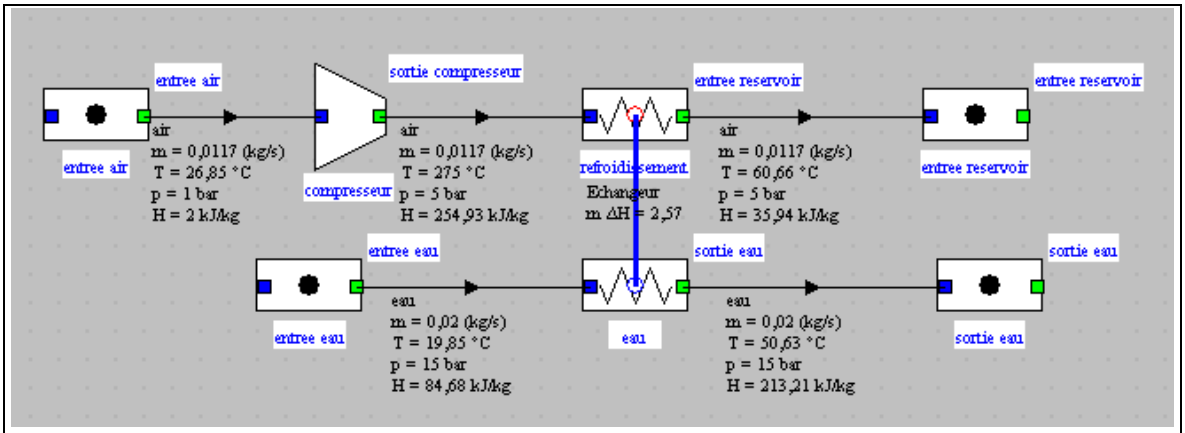

Figure 1 : Exemple de compresseur refroidi

Le système peut être facilement modélisé dans Thermoptim et conduit à un schéma du type de la figure 1.

L'écran de dimensionnement technologique du compresseur est donné figure 2. Celui de l'échangeur est donné figure 3.

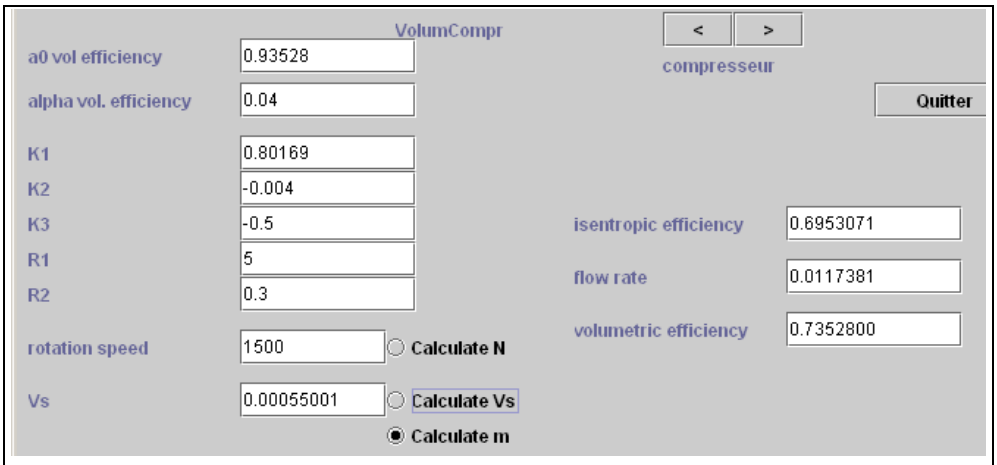

Figure 2 : Ecran technologique du compresseur

Notez qu'un double-clic sur le nom du composant (ici "compresseur" situé sous les petites flèches en haut à droite) permet d'accéder à son écran classique dans le simulateur.

Le paramétrage de ces écrans technologiques est expliqué dans diverses pages du portail Thermoptim-UNIT[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

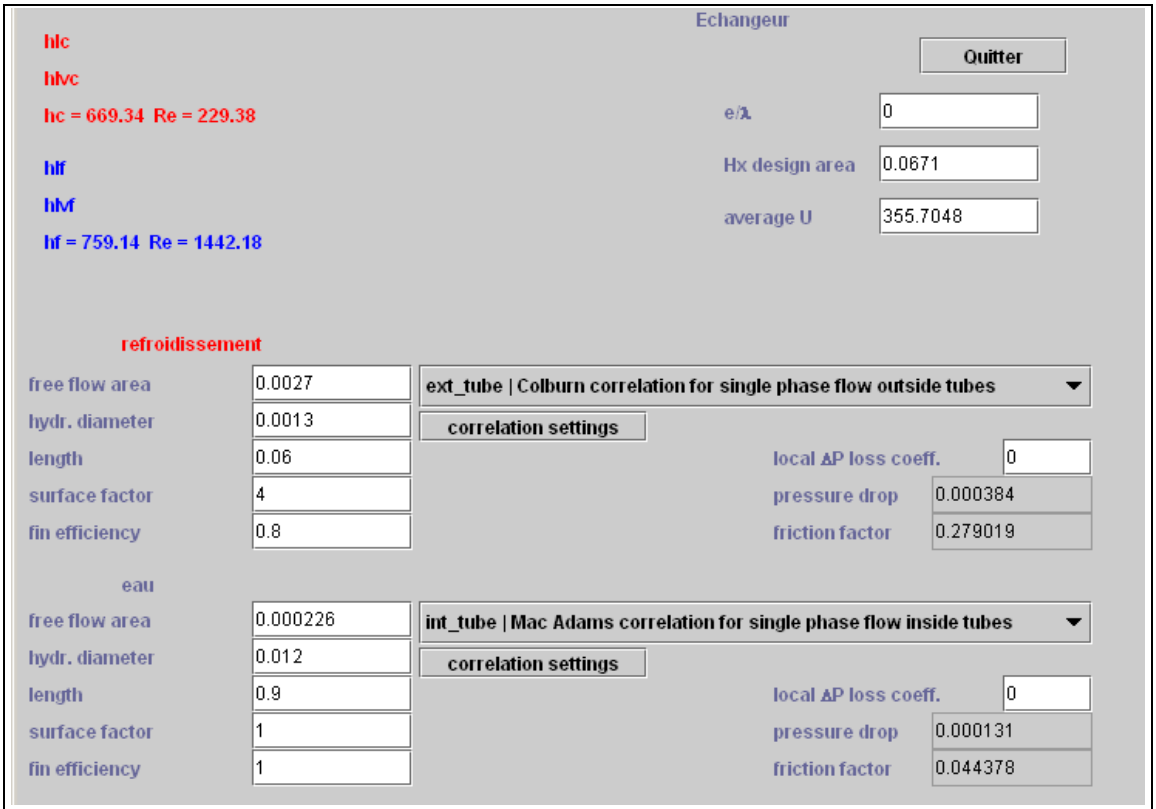

Figure 3 : Ecran technologique de l'échangeur

1.1 Résultats obtenus

Une fois le pilote réalisé, il est très facile de faire varier le rapport de compression pour obtenir les évolutions des principales grandeurs lorsque la pression du réservoir varie. Elles peuvent être exploitées avec la macro Excel de post-traitement des fichiers de simulation de Thermoptim présentée au tome 1 du manuel de référence. Il suffit pour cela de sauvegarder le fichier de projet sous un nom différent après chaque simulation, puis de charger ces fichiers dans la macro et d'en extraire les valeurs intéressantes.

La figure 4 montre, en fonction de la pression du stockage, les évolutions du rendement isentropique du compresseur et du travail consommé, le débit d'air aspiré, la température de l'air entrant dans le réservoir, la charge de l'échangeur et la valeur de U.

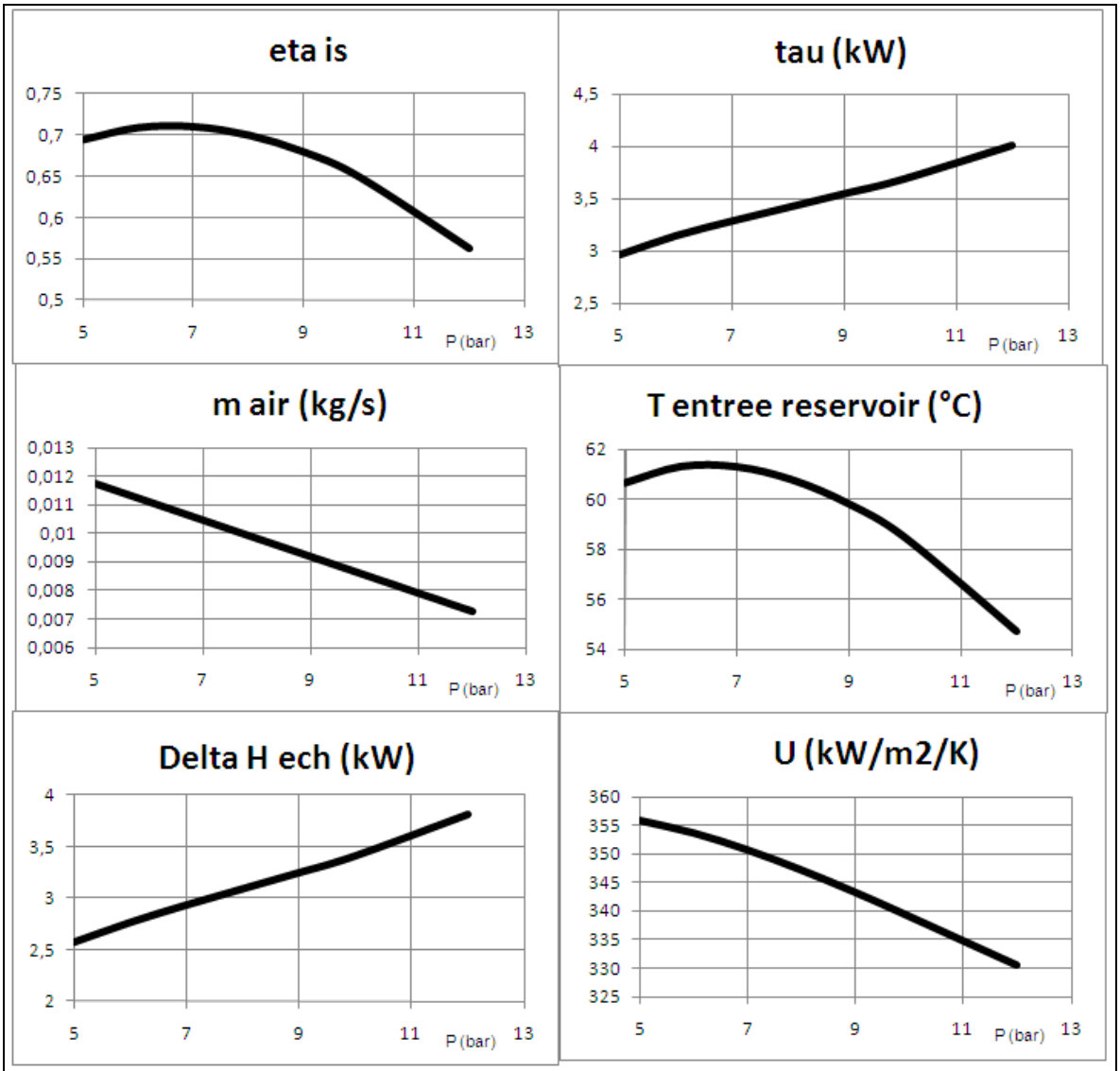

Figure 4 : Résultats de simulation

1.2 Conception du pilote

La classe du pilote est appelée PiloteCompresseurVs. Son interface graphique étant simple et classique, nous ne la détaillerons pas ici.

L'écran du pilote est présenté figure 5.

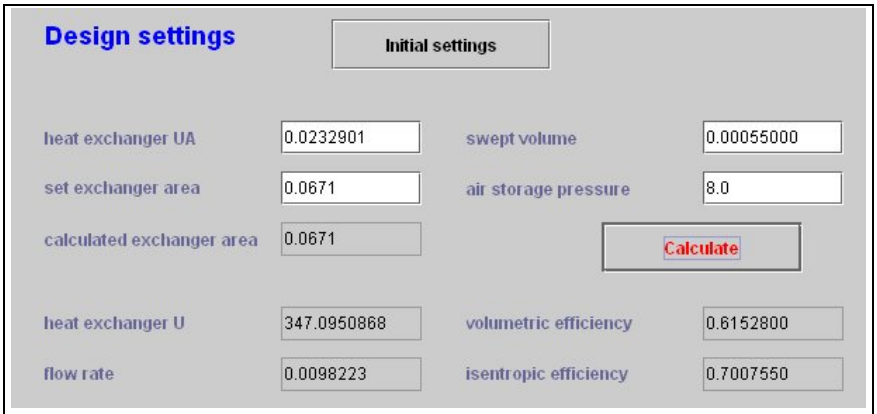

Figure 5 : Ecran du pilote

Pour réaliser la classe correspondante, opère de la manière suivante :

- on commence par instancier les écrans technologiques du compresseur et de l'échangeur ;
- on les initialise ;

- on définit les calculs et modifications du paramétrage du simulateur qu'il convient d'effectuer lorsqu'on fait varier la pression aval.

1.2.1 Instanciations

```
//initialisations des PointThopt et des TechnoDesign
//attention : les noms des points et composants doivent être exacts, sous peine de gén

hxName="Echangeur";
compressorName="compresseur";
amontChaud=new PointThopt(proj,"sortie compresseur");
avalChaud=new PointThopt(proj,"entree reservoir");
amontFroid=new PointThopt(proj,"entree eau");
avalFroid=new PointThopt(proj,"sortie eau");
amontCompr=new PointThopt(proj,"entree air");

//instanciation des TechnoDesign dans les classes externes
technoEchangeur=new TechnoHx(proj, hxName, amontChaud, avalChaud, amontFroid, avalFroi
addTechnoVector(technoEchangeur);
technoCompr=new VolumCompr(proj, compressorName, amontCompr, amontChaud);
addTechnoVector(technoCompr);

//initialisation des TechnoDesign dans Thermoptim
setupTechnoDesigns(vTechno);
```

1.2.2 Initialisations

Lors de l'initialisation, l'écran technologique de l'échangeur est mis à jour à partir des valeurs du simulateur, et sa surface calculée :

```
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;
    amontChaud.getProperties();
    avalChaud.getProperties();
    amontFroid.getProperties();
    avalFroid.getProperties();

    mCpEau=DeltaH/(avalFroid.T-amontFroid.T);
    mCpAir=DeltaH/(amontChaud.T-avalChaud.T);
    UAech_value.setText(Util.aff_d(UAech,4));

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

La cylindrée du compresseur Vs permettant d'obtenir le débit souhaité est ensuite déterminée, sur la base du paramétrage de l'écran technologique :

```
if(!compressorName.equals("")){//initialisation du compresseur
    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*amontCompr.V/N_value/lambdaVol;
    Vs_value.setText(Util.aff_d(Vs,8));
    technoCompr.setVs(Vs);
    technoCompr.setN(N_value);
}
```

1.2.3 Calculs

Les calculs à effectuer sont les suivants ;

- commencer par initialiser la cylindrée et mettre à jour la pression de sortie du compresseur
- calculer les rendements volumétrique et isentropique, déterminer le débit massique mis en jeu et le propager en amont et en aval
- recalculer le compresseur et la transfo aval, sachant que celle-ci, paramétrée comme isobare, propage la nouvelle pression
- mettre à jour les entrées et sorties de l'échangeur puis le recalculer en non-nominal après avoir actualisé le débit calorifique de l'air qui a été modifié
- mettre à jour le simulateur et recalculer le projet plusieurs fois pour garantir la stabilisation des valeurs.

Les cinq premières étapes correspondent au code ci-dessous :

```
void bCalc_actionPerformed(java.awt.event.ActionEvent event)
{
    Vs=Util.lit_d(Vs_value.getText());//mise à jour de la cylindrée du compresseur
    technoCompr.setVs(Vs);
    Preservoir=Util.lit_d(P_value.getText());
    double UA_ech=Util.lit_d(UAech_value.getText());

    amontChaud.P=Preservoir;// mise à jour de la pression de sortie du compresseur
    amontChaud.update(!UPDATE_T,UPDATE_P,!UPDATE_X);
    amontChaud.getProperties();

    massFlow=technoCompr.getMassFlow(amontCompr.V);
    double eta_is=technoCompr.getRisentr();// calcul du rendement isentropique du compresseur
    lambdaVol=technoCompr.getLambdaVol();

    //recalcul du compresseur et de la transfo aval
    updateprocess(compressorName, "Compression",RECALCULATE,IS_SET_FLOW, UPDATE_FLOW, massFlow,  U
    amontChaud.getProperties();
    updateprocess("refroidissement", "Exchange",RECALCULATE,IS_SET_FLOW, UPDATE_FLOW, massFlow,  U
    updateprocess("entree air", "Exchange",RECALCULATE,IS_SET_FLOW, UPDATE_FLOW, massFlow,  UPDATE

    amontChaud.getProperties();//mise à jour des entrées et sorties de l'échangeur
    avalChaud.getProperties();
    amontFroid.getProperties();
    avalFroid.getProperties();
```

Cet exemple illustre l'intérêt qu'il y a à utiliser les PointThopt pour communiquer entre le pilote et le simulateur. La syntaxe des mises à jour est beaucoup plus lisible qu'en utilisant seulement les méthodes getProperties() et updatePoint() de Projet.

Le calcul du compresseur se fait grâce aux deux méthodes getMassFlow() et getLambdaVol() de l'écran technologique. La méthode updateprocess() modifie le débit et le rendement isentropique du compresseur puis le recalcule. Son point aval, qui s'appelle ici "amontChaud" car il correspond à l'amont du fluide chaud de l'échangeur, est ensuite mis à jour.

Le calcul de l'échangeur est fait de la manière suivante : on fait un premier calcul avec la méthode updateHx() en imposant la valeur de UA lue à l'écran, UA_ech (l'échangeur est paramétré pour un calcul en mode non-nominal, afin que ce soit la valeur de UA qui soit prise en compte). Ce calcul permet d'initialiser l'échangeur pour les nouvelles conditions de fonctionnement.

On fait ensuite appel à la méthode makeDesign() du TechnoDesign, ce qui met à jour la valeur de U. Le véritable UA s'obtient en multipliant le nouveau U avec la valeur de A initiale (AechReel), ce qui permet de recalculer l'échangeur correctement.

Etant donné que U dépend des températures moyennes des fluides, et donc de celles de sortie, on itère cinq fois les calculs pour garantir une bonne stabilisation, en réinitialisant à chaque fois UA_ech.

```
//calcul de l'échangeur (on itère plusieurs fois)
for(int i=0;i<5;i++){
    updateHx(hxName, RECALCULATE, UPDATE_UA, UA_ech, !UPDATE_EPSI, 0, !UPDATE_DTMIN, 0, UPDATE_CALC_MODE, OFF_DESIGN_MODE);
    avalFroid.getProperties();
    avalChaud.getProperties();

    technoEchangeur.UA=UA_ech;//détermination de U
    technoEchangeur.makeDesign();
    double U=technoEchangeur.getU();

    UAech=U*AechReel/1000.;//mise à jour de UA et recalcul de l'échangeur
    UAech_value.setText(Util.aff_d(UAech,4));
    updateHx(hxName, RECALCULATE, UPDATE_UA, UAech, !UPDATE_EPSI, 0, !UPDATE_DTMIN, 0, UPDATE_CALC_MODE, OFF_DESIGN_MODE);

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

//mise à jour du simulateur et des affichages
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(technoEchangeur.ADesign_value.getText());
```