T H E R M O P T I M ®

NON NOMINAL

PILOTE POUR CYCLE DE REFRIGERATION

AVEC PERTES DE CHARGE ET

CHARGE DE REFRIGERANT IMPOSEE

RESOLUTION PAR MINPACK 1

VERSION JAVA 1.7

SOMMAIRE

1 Introduction - pré-requis

L'objectif de cette notice est de permettre à un développeur de se familiariser avec l'écriture d'un pilote pour l'étude en régime non-nominal d'un cycle frigorifique avec Thermoptim. Nous faisons l'hypothèse que vous connaissez déjà bien Thermoptim et son mécanisme de classes externes, et que vous avez pris connaissance de l'ensemble des quatre tomes du manuel de référence du progiciel. Nous vous conseillons par ailleurs de vous référer au tome 3 du livre Systèmes Energétiques[1] pour de plus amples développements sur la problématique du dimensionnement technologique et du non-nominal.

Le compresseur utilisé sera d'abord volumétrique (classe de pilotage PiloteFrigoAuxChargeVolum), puis centrifuge (classe de pilotage PiloteAuxFrigoChargeTurbo, avec cartographie définie dans le fichier dataCentrifrRefR134a.txt). Dans les deux cas, le fluide frigorigène considéré est le R134a, ce qui permet de comparer les résultats obtenus.

2 Principe de calcul des composants

Le dimensionnement technologique des composants nécessite d'affiner les modèles phénoménologiques utilisés dans le noyau de Thermoptim, en les complétant pour prendre en compte les mécanismes de fonctionnement à charge partielle s'il en existe. Le progiciel a pour cela été doté de nouveaux écrans, dits de dimensionnement technologique, qui permettent de définir les caractéristiques géométriques représentatives des différentes technologies utilisées, ainsi que les paramètres nécessaires pour le calcul de leurs performances.

Ce nouvel environnement, développé sous forme de classes externes, doit pouvoir travailler de manière à la fois complémentaire de celle des composants du noyau, et en même temps parfaitement cohérente avec eux. Selon les moments, les calculs sont en effet soit effectués par le noyau du progiciel, soit réalisés par les classes de dimensionnement technologique, le pilote assurant la synchronisation entre les deux modes.

2.1 Calcul des échangeurs en régime non-nominal

La méthode du NUT peut être présentée comme suit :

$$NUT = \frac{UA}{(\dot{m}\cdot c_p)_{min}} \qquad (1)$$

$$R = \frac{(\dot{m}c_p)_{min}}{(\dot{m}c_p)_{max}} \leq 1 \qquad (2)$$

L'efficacité de l'échangeur vaut : $\varepsilon = \frac{\Delta T_{max}}{\Delta T_e}$ (3)

Avec ces définitions, il est possible de montrer qu'il existe une relation générale du type :

ε = f(NUT, R, configuration d'écoulement)

En **mode dimensionnement**, on connaît les débits des deux fluides, leurs températures d'entrée et le flux à échanger, on opère de la manière suivante :

- on commence par déterminer les températures de sortie des fluides ;
- on en déduit les débits de capacité thermique $\dot{m}\, c_p$ des fluides et leur rapport R ;
- on calcule l'efficacité ε à partir de l'équation (3) ;
- on détermine la valeur du NUT à partir de la relation (NUT, ε) appropriée ;
- on calcule le produit UA à partir de l'équation (1).

[1] GICQUEL R., Systèmes Energétiques, Tome 3 : cycles et modélisations avancés, systèmes innovants a faible impact environnemental, Presses de l'Ecole des Mines de Paris, janvier 2009.

En **régime non-nominal**, on connaît les températures d'entrée et les débits des deux fluides, la surface A de l'échangeur et sa géométrie (configurations d'écoulement et paramètres technologiques). Le calcul se fait alors en trois étapes :

- détermination de U par des corrélations dépendant de la configuration d'écoulement et de la géométrie de l'échangeur ;
- calcul de UA, produit de U et de A, puis de NUT par (1) ;
- détermination de l'efficacité de l'échangeur par la relation (NUT, ε) appropriée, et calcul des températures de sortie par (3) et les équations de bilan.

Connaissant Tce, Tfe, mc, mf et U, il est ainsi possible de calculer R, puis NUT, d'en déduire ε, et de déterminer les températures de sortie Tcs et Tfs.

Rappelons que la méthode du NUT fait l'hypothèse que les propriétés thermophysiques du fluide sont constantes dans l'échangeur, alors que ce n'est vrai qu'en première approximation. Si on considère U variable, dépendant comme c'est le cas aussi bien des températures d'entrée que des températures de sortie, on obtient un système d'équations implicite très difficile à résoudre, surtout si les échangeurs sont multizones comme pour les évaporateurs ou condenseurs.

En pratique, on peut cependant souvent considérer que U ne varie qu'au deuxième ordre, et rechercher une solution approchée en considérant U constant, puis recalculer sa valeur pour les nouvelles conditions de fonctionnement, et itérer jusqu'à obtenir une précision raisonnable. Il est en particulier nécessaire d'opérer ainsi lorsque l'échangeur est multizones, car seule la surface totale est connue, et non sa répartition entre les différentes zones. C'est précisément comme cela que nous opérerons.

Les calculs dans le simulateur sont donc effectués en appliquant la méthode du NUT, le UA étant une inconnue intermédiaire, tandis que les calculs dans les écrans technologiques se font de manière détaillée, conduisant, pour un jeu des valeurs d'entrée-sortie de l'échangeur donné et compte tenu de la valeur de U correspondante, à une estimation de la surface requise. La cohérence entre les deux calculs est assurée lorsque la valeur du UA est telle que la surface totale d'échange est égale à celle de dimensionnement.

2.2 Calcul des compresseurs volumétriques en régime non-nominal

Un compresseur volumétrique est défini géométriquement par sa cylindrée et les paramètres technologiques permettant de calculer ses rendements volumétrique et isentropique, en fonction de sa vitesse de rotation et des conditions d'aspiration et de refoulement

Les modèles implémentés dans Thermoptim sont de ce fait basés sur l'hypothèse que le comportement des compresseurs volumétriques peut être représenté avec une précision raisonnable par deux grandeurs : leur rendement volumétrique λ qui caractérise le volume balayé réel (4), et leur rendement isentropique classique η_s (5).

$$\lambda = a_0 - a_1 \frac{P_{ref}}{P_{asp}} \qquad (4)$$

$$\eta_s = K_1 + K_2 \cdot \frac{P_{asp}}{P_{ref}} + K3 \cdot \left(\frac{P_{asp}}{P_{ref}} \right)^2 \qquad (5)$$

On notera que (5) s'exprime linéairement en fonction de l'inverse du rapport de compression et de son carré.

Le calcul d'un compresseur se fait de la manière suivante :

- les rendements volumétrique et isentropique sont calculés à partir des équations (4) et (5)
- si l'on connaît la vitesse de rotation, la cylindrée et le débit volumique demandé, le débit volumique peut être calculé :

$$\dot{V} = \frac{\lambda \, N \, V_S}{60} \qquad (6)$$

- connaissant la masse volumique à l'aspiration v, on déduit le débit massique :

$$\dot{m} = \frac{\dot{V}}{v} = \frac{\lambda N V_s}{60\, v} \qquad (7)$$

En mode dimensionnement, on détermine la vitesse de rotation ou la cylindrée permettant de fournir le débit souhaité. Le calcul est effectué en prenant en compte les pressions d'entrée et de sortie et la valeur du débit de la transfo du compresseur dans le simulateur.

En régime non-nominal, le rapport de compression détermine λ et η_s, ce qui fixe le débit et la température de sortie compresseur.

3 Choix et paramétrage des écrans technologiques

Supposons que l'on veuille dimensionner un cycle de réfrigération simple (figure 1) pour qu'il fournisse une puissance frigorifique de 130 kW pour une température extérieure de 30 °C, la température de l'eau glycolée étant égale à -10 °C en entrée de l'évaporateur.

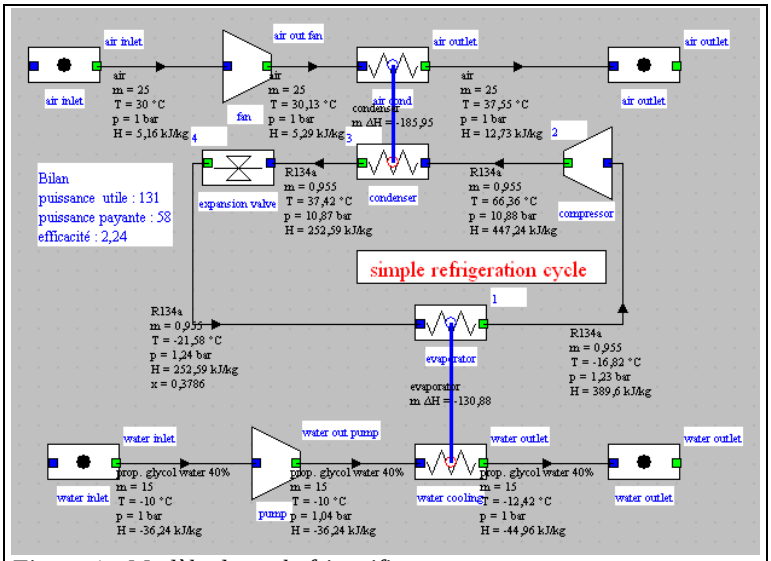

Figure 1 : Modèle de cycle frigorifique

Pour ce cycle, le fluide froid est du propylène glycol à 40 % en volume, disponible dans les corps externes. La température d'évaporation doit bien évidemment être inférieure à celle de l'eau glycolée. Nous avons retenu environ –21,6 °C, c'est-à-dire une pression de 1,24 bars pour du R134a.

On suppose que le rendement isentropique du compresseur vaut environ 0,8 au point nominal. Pour simplifier un peu le modèle on fait l'hypothèse que la valeur de la surchauffe à l'évaporation reste constante (5 K). Celle du sous-refroidissement à la condensation est en revanche déterminée sur la base de la masse globale de frigorigène, qui reste constante dans la machine.

La surchauffe à l'évaporation ΔT_{surch} vaut 5 K, ce qui pour un débit de réfrigérant de 0,955 kg/s fixe la puissance frigorifique à environ 131 kW. Avec un débit d'eau glycolée de 15 kg/s, cela conduit à un refroidissement de 2,42 °C.

Pour le condenseur, la température de l'air vaut 30 °C, et son débit 25 kg/s. La température de condensation est estimée à 42 °C, ce qui, avec un sous-refroidissement initial ΔT_{ssrefr} de 5 K, correspond à une pression de condensation d'environ 10,9 bars. Le COP de la machine vaut dans ces conditions 2,24.

La prise en compte des pertes de charge du réfrigérant demande quelques précautions pour éviter des difficultés de calcul dans les phases de condensation et d'évaporation. Nous avons choisi de les affecter uniquement aux points dont l'état est monophasique : en totalité en aval de l'évaporateur, et pour moitié en amont et en aval du condenseur.

Les pertes de charge sur l'air et l'eau glycolée sont calculées dans les écrans technologiques des échangeurs, puis affectées comme surpression à réaliser en amont de l'échangeur. Les ventilateurs et la pompe à eau glycolée sont ici modélisés par des transfos compression classiques, sans prise en compte de leurs caractéristiques détaillées. Il serait possible de le faire, mais au prix d'une complexité accrue qui n'apparaît pas vraiment justifiée.

Le dimensionnement se fait en deux étapes distinctes, la première correspond au paramétrage classique du cycle dans Thermoptim, tandis que la seconde est effectuée à partir des écrans technologiques. Précisons, et ceci est très important, que la deuxième étape ne peut être effectuée que lorsque la première a été menée à son terme et a permis d'obtenir un modèle parfaitement cohérent. Si ce n'est pas le cas, le dimensionnement technologique effectué au cours de la seconde étape risque d'être aberrant et d'entraîner de grandes difficultés de convergence.

Nous ne détaillerons pas ici, pour simplifier les choses, la manière de construire le modèle Thermoptim du cycle correspondant à la première étape. S'il n'existait pas, il faudrait d'abord le faire, puis la méthode serait la même. Ce cycle de réfrigération est semblable à ceux qui ont été présentés dans les guides de prise en mains du progiciel, la différence principale étant que le condenseur est modélisé par un seul échangeur multizone, la désurchauffe n'étant pas dissociée dans l'éditeur de schémas de la condensation proprement dite.

3.1 Création des écrans technologiques

La création des écrans technologiques peut être effectuée en utilisant soit le pilote générique, soit un pilote particulier, ce qui est nécessaire lorsque l'on veut faire des simulations en régime non-nominal comme c'est le cas ici.

Dans cet exemple, ces écrans sont créés de la manière suivante : la première étape dans la construction du pilote est celle de l'instanciation d'une part des différents PointThopt[2] qui vont permettre d'accéder aux points du simulateur (un pour chaque point du cycle), et d'autre part des TechnoDesign.

```
amontEvap=new PointThopt(proj,"4");
avalEvap=new PointThopt(proj,"1");
amontCond=new PointThopt(proj,"2");
avalCond=new PointThopt(proj,"3");
waterInlet=new PointThopt(proj,"water inlet");
waterOutlet=new PointThopt(proj,"water outlet");
inAir=new PointThopt(proj,"air inlet");
outAir=new PointThopt(proj,"air outlet");
avalCond.getProperties();
refrig=(rg.corps.Corps)avalCond.lecorps;

 waterPumpOutlet=new PointThopt(proj,"water out pump");
 waterPumpOutlet.getProperties();
 outFanAir=new PointThopt(proj,"air out fan");
 outFanAir.getProperties();

evaporatorName="evaporator";
 condenserName="condenser";
compressorName="compressor";
```

Les TechnoDesign sont du type TechnoEvaporator, TechnoCondensor et VolumCompr, trois classes spécifiquement créées pour ce type de composants. Les deux premières permettent de calculer l'évaporateur et le condenseur comme des échangeurs multizones, selon les équations données en annexe 1. La troisième implémente les équations (4) à (7). Nous ne les détaillerons pas ici, nous contentons d'expliquer comment les utiliser, et renvoyant le lecteur au code de leurs classes pour plus de précisions.

```
technoEvap=new TechnoEvaporator(proj, evaporatorName, waterInlet, waterOutlet, amontEvap, avalEvap);
addTechnoVector(technoEvap);
technoCond=new TechnoCondensor(proj, condenserName, amontCond, avalCond, inAir, outAir);
addTechnoVector(technoCond);
technoCompr=new VolumCompr(proj, compressorName, avalEvap, amontCond);
addTechnoVector(technoCompr);
setupTechnoDesigns(vTechno);
```

La dernière ligne du code ci-dessus permet de transférer ces écrans technologiques dans le noyau du progiciel. Les valeurs de la cylindrée et de la vitesse de rotation du TechnoDesign sont alors affichées à l'écran.

[2] Cette classe permet de créer dans une classe externe des sortes de clones des points du noyau de Thermoptim, afin d'avoir un accès aisé à leurs valeurs, qui ne sont pas directement accessibles. Elle apporte davantage de confort et de lisibilité que ne le fait l'utilisation des méthodes getProperties() et updatePoint de Projet, documentées dans le tome 3 du manuel de référence.

```
VsValue=Util.lit_d(technoCompr.Vs_value.getText());
Vs_value.setText(Util.aff_d(VsValue,8));
N_value=Util.lit_d(technoCompr.N_value.getText());
Nref_value.setText(Util.aff_d(N_value,4));
```

3.2 Ecran du pilote

La partie supérieure de l'écran du pilote est donnée figure 2.

Elle permet de modifier d'une part les surfaces des échangeurs, et d'autre part la longueur de la ligne liquide (cf. section 6), la température d'air, la vitesse de rotation ou la cylindrée, et possède des options de guidage de l'algorithme qui seront précisées plus loin.

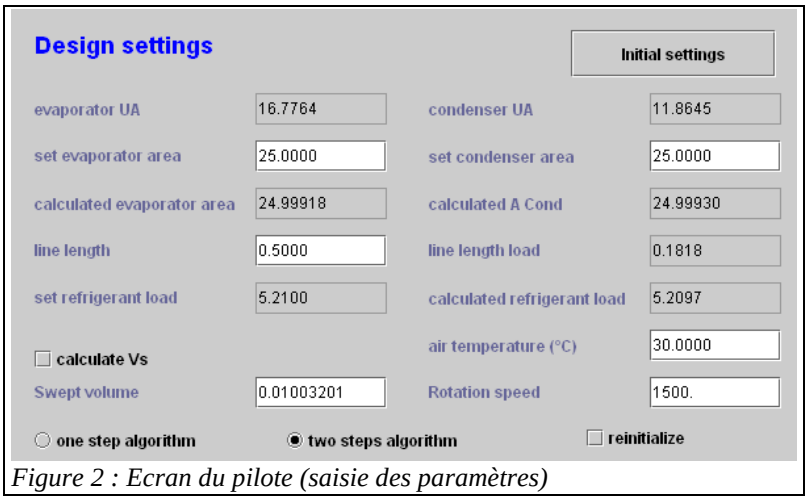

Figure 2 : Ecran du pilote (saisie des paramètres)

Par défaut, c'est la vitesse de rotation qui est calculée, pour la valeur de la cylindrée entrée à l'écran. Si vous choisissez l'option "Calculate Vs", c'est la cylindrée qui est calculée, pour la valeur de la vitesse de rotation saisie.

Commencez par cliquer sur "Initial settings" pour instancier les TechnoDesign et effectuer un premier dimensionnement technologique, en l'occurrence calculer la vitesse de rotation ou la cylindrée du compresseur ainsi que les surfaces des deux échangeurs correspondant au paramétrage du fichier de projet, sur la base des valeurs par défaut des paramètres des TechnoDesign.

3.3 Paramétrage des écrans technologiques

Les écrans technologiques ayant été créés, vous pouvez passer à leur paramétrage puis à leur dimensionnement. Cette étape doit être faite avec soin, car elle suppose d'effectuer toute une série de choix quant aux configurations internes, aux dimensions géométriques…

Pour accéder aux écrans technologiques, faites-le à partir des tables de l'écran général depuis le simulateur (Ctrl T), ou bien à partir des boutons "tech. design" des écrans classiques des composants.

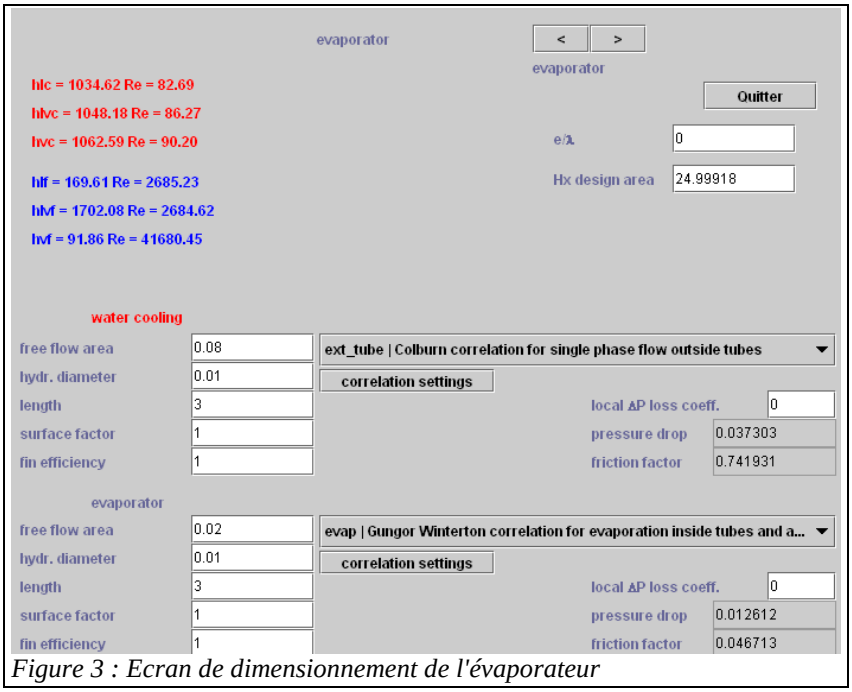

Figure 3 : Ecran de dimensionnement de l'évaporateur

3.3.1 Evaporateur

On considère que l'évaporateur est du type tubes et calandre et qu'il a côté eau une section de passage du fluide de 8 dm^2 et un diamètre hydraulique de 1 cm, et côté réfrigérant une section de passage de 2 dm^2 et un diamètre hydraulique de 1 cm, pour une longueur de tubes de 3 m. Choisissez le type de configuration ("evap Gungor Winterton" pour évaporation à l'intérieur des tubes pour la partie "évaporateur" (le frigorigène), et "ext tube Colburn correlation" pour l'eau glycolée, et paramétrez l'écran comme indiqué figure 3.

3.3.2 Condenseur

Le condenseur est du type batterie à ailettes. Côté air il a une section de passage du fluide de 1 m^2 et un diamètre hydraulique de 1 cm avec un facteur de surfaces étendues de 20, sur une longueur de 15 cm (épaisseur de l'aérocondenseur), et côté réfrigérant une section de passage de 0,5 dm^2 et un diamètre hydraulique de 2 cm, pour une longueur de tubes de 3 m.

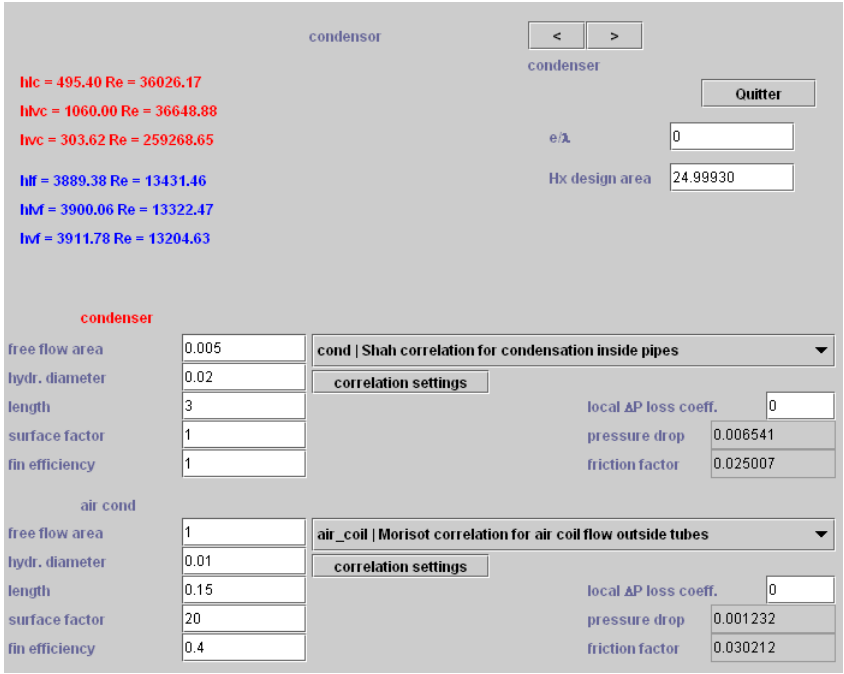

Figure 4 : Ecran de dimensionnement du condenseur

Choisissez le type de configuration ("cond Shah correlation" pour la condensation à l'intérieur des tubes pour la partie "condenseur" (le frigorigène), et "air coil Morizot correlation" pour l'air, et paramétrez l'écran comme indiqué figure 4.

3.3.3 Compresseur

Pour le compresseur (figure 5), il faut fournir d'une part la cylindrée Vs, et d'autre part les valeurs des paramètres qui interviennent dans les équations des rendements volumétrique et isentropique.

Comme nous employons l'équation à trois paramètres, les paramètres R1 et R2, non utilisés, sont égaux à 0.

Pour réaliser le dimensionnement une fois les écrans technologiques renseignés, recliquez sur le bouton "Initial settings" de l'écran du pilote (figure 2).

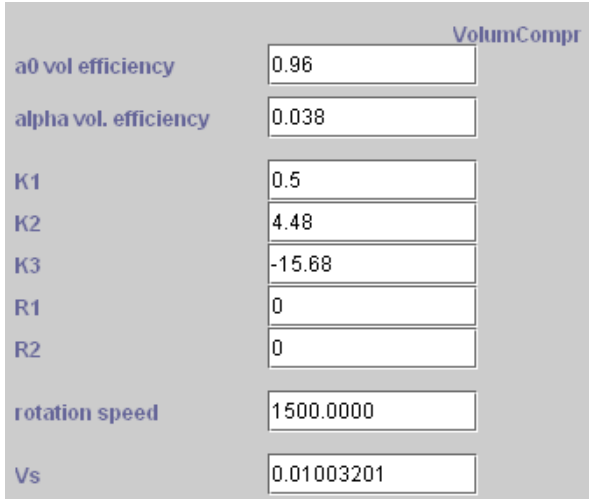

Figure 5 : Ecran de dimensionnement du compresseur

Les résultats sont affichés dans les écrans technologiques : surfaces d'échanges de 25 m^2 pour l'évaporateur et le condenseur ; vitesse de rotation de 1500 tr/mn et cylindrée d'environ 0,01 m^3 pour le compresseur.

Divers résultats de calcul sont affichés sur les écrans technologiques, comme, pour les échangeurs, les pertes de charge et les valeurs du nombre de Reynolds Re et des coefficients d'échange locaux.

4 Initialisations de dimensionnement au point nominal

Le paramétrage des écrans technologiques permet de déterminer les surfaces d'échange, la masse totale de réfrigérant et la vitesse de rotation du compresseur, à partir des initialisations correspondant au point nominal, qui servent à fixer un certain nombre de valeurs à partir de celles du projet Thermoptim, comme les ΔT de surchauffe et de sous-refroidissement initial.

Le calcul précis des échangeurs multizones est quant à lui effectué par la méthode makeDesign() des TechnoDesign des échangeurs, alors que la valeur globale de UA est obtenue de manière classique, grâce aux modèles phénoménologiques.

Expliquons pour commencer l'initialisation du condenseur, celle de l'évaporateur étant analogue.

Les premières lignes du code permettent, en utilisant la méthode getProperties() du projet de Thermoptim (proj), connaissant le nom de l'échangeur (condenserName), de récupérer la valeur de UAcond et le nom du fluide froid, puis l'enthalpie mise en jeu DeltaH.

```
if(!condenserName.equals("")){//initialisation du Condenseur
 String[] args=new String[2];
 args[0]="heatEx";
 args[1]=condenserName;
 Vector vProp=proj.getProperties(args);
 Double f=(Double)vProp.elementAt(15);
 UAcond=f.doubleValue();
 String fluideFroid=(String)vProp.elementAt(1);
 args[0]="process";
 args[1]=fluideFroid;
 vProp=proj.getProperties(args);
 f=(Double)vProp.elementAt(4);
 double DeltaH=f.doubleValue();
```

Les méthodes getProperties() des PointThopt fournissent directement l'état thermodynamique complet des points amont et aval du condenseur, ce qui permet d'initialiser le sous-refroidissement.

```
 avalCond.getProperties();
 amontCond.getProperties();
 DTssrefr=avalCond.DTsat;
```

De la même manière, la valeur de la température de l'air est obtenue du simulateur et affichée dans l'écran du pilote.

```
 inAir.getProperties();
 outAir.getProperties();
 Tair=inAir.T;
 Ta_value.setText(Util.aff_d(Tair-273.15,4));
```

Ces valeurs permettent d'initialiser les valeurs des débits calorifiques du condenseur, qui seront utilisées ultérieurement.

```
 mCpCalopCond=DeltaH/(outAir.T-outFanAir.T);
 mCpRefrigCond=DeltaH/(amontCond.T-avalCond.T);
 UAcond_value.setText(Util.aff_d(UAcond,4));
```

Le TechnoDesign est alors initialisé à son tour, ce qui permet de connaître la surface d'échange nécessaire compte tenu du dimensionnement réalisé, puis les pertes de charge sont mises à jour.

```
 //initialisations du TechnoDesign
 technoCond.makeDesign();
 AcondReel=Util.lit_d(technoCond.ADesign_value.getText());
 AcalculatedCond_value.setText(technoCond.ADesign_value.getText());

//calcul des pertes de charge
 dPcond=technoCond.techc.getPressureDrop()*dPmult;
```

La charge initiale de frigorigène est alors calculée compte tenu des dimensions géométriques et de l'état thermodynamique du fluide.

```
 //calcul de la charge de frigorigène
 mEvap=technoEvap.techf.getFluidLoad();
 mCond=technoCond.techc.getFluidLoad();

 lineLength=Util.lit_d(lineLength_value.getText());
 double dh=Util.lit_d(technoCond.techc.Dh_value.getText());
 volLigne=Math.PI*dh*dh/4.*lineLength;
 mLigne=volLigne/avalCond.V;
 mLine_value.setText(Util.aff_d(mLigne,4));

 mTot=mCond+mEvap+mLigne;
 mTot_value.setText(Util.aff_d(mTot,4));
}
```

L'initialisation du compresseur permet de connaître le débit nominal et de déterminer la vitesse de rotation du compresseur correspondante, qui est affichée dans l'écran du pilote.

```
if(!compressorName.equals("")){//initialisation du compresseur
 String[] args=new String[2];
 args[0]="process";
 args[1]=compressorName;
 Vector 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();
 lambdaVol=technoCompr.getLambdaVol();
 if(!jCheckVs.isSelected()){//calcul de la vitesse de rotation
 N_value=massFlow*60*avalEvap.V/VsValue/lambdaVol;
 technoCompr.setN(N_value);
 Nref_value.setText(Util.aff_d(N_value,4));
 }
 else{//calcul de la cylindrée
 N_value=Util.lit_d(Nref_value.getText());
 technoCompr.setN(N_value);
 VsValue=massFlow*60*avalEvap.V/N_value/lambdaVol;
 technoCompr.setVs(VsValue);
 Vs_value.setText(Util.aff_d(VsValue,8));
 }
}
```

Les consommations des auxiliaires sont ensuite recalculées :

```
 //mise à jour des auxiliaires

 outFanAir.P=inAir.P+dPair;
 outFanAir.update(!UPDATE_T,UPDATE_P,!UPDATE_X);
 updateprocess(fanName, "Compression",RECALCULATE,!IS_SET_FLOW, !UPDATE_FLOW, 0,
!UPDATE_ETA, 0);
 outFanAir.getProperties();

 waterPumpOutlet.P=waterInlet.P+dPwater;
 waterPumpOutlet.update(!UPDATE_T,UPDATE_P,!UPDATE_X);
 updateprocess(pumpName, "Compression",RECALCULATE,!IS_SET_FLOW, !UPDATE_FLOW, 0,
!UPDATE_ETA, 0);
 waterPumpOutlet.getProperties();
```

5 Résolution du système d'équations de la machine frigorifique en non-nominal

Dans toute étude de comportement en régime non-nominal, il faut bien identifier quelles sont les variables indépendantes du système considéré, en les distinguant des variables liées qui s'en déduisent.

Dans cet exemple, il s'agit du quadruplet (débit de réfrigérant, température d'évaporation, température de condensation, sous-refroidissement), les variables liées étant les pressions du fluide et les puissances thermiques et mécaniques mises en jeu. A ces quatre variables "naturelles" , il faut ajouter les deux variables intermédiaires UAevap et UAcond (cf. section 2.1).

La recherche de ce sextuplet correspond à celle de la solution d'un jeu d'équations non linéaires relativement complexe qui met en jeu celles que nous venons de présenter et d'autres qui seront détaillées section 6.

La solution que nous avons retenue consiste à programmer un pilote externe qui assure la résolution de ce système d'équations et met à jour Thermoptim une fois la solution trouvée.

5.1 Méthode de résolution

Nous utilisons la méthode de Levendberg-Marquardt correspondant aux algorithmes mis au point en Fortran sous le nom de minPack 1, et traduits sous Java. Cette méthode combine l'algorithme de Gauss-Newton et la méthode de descente de gradient. Son intérêt principal est d'être très robuste et de ne nécessiter comme initialisation qu'une solution approchée.

Son implémentation sous Java se fait en utilisant une interface appelée optimization.Lmdif_fcn, qui contraint les classes appelantes (ici notre pilote) à disposer d'une fonction appelée fcn().

Cette fonction fcn() reçoit comme principaux arguments un vecteur (tableau x[n])[3] contenant les variables et un vecteur (tableau fvec[m]) renvoyant les résidus des fonctions que l'on cherche à annuler. Leur nombre peut excéder celui des variables, mais dans notre cas il sera le même.

Le guidage de l'algorithme se fait en pratique en jouant sur deux critères de précision, l'un portant sur la somme des résidus, et l'autre sur la précision du calcul des dérivées partielles, estimées par différences finies. N'oublions pas que nous cherchons à résoudre un système de six équations non linéaires à six inconnues, ce qui peut se révéler numériquement difficile. A l'usage, il est apparu intéressant de proposer plusieurs options de calcul.

Tout d'abord, l'option "reinitialize" offre la possibilité de réinitialiser les valeurs des températures d'évaporation et de condensation en fonction de celles de l'eau glycolée et de l'air ambiant, pour éviter tout croisement de température dans les échangeurs.

Ensuite, deux options exclusives sont proposées : soit exécuter l'algorithme en une seule étape, pour des valeurs intermédiaires de précision des critères de convergence ("one step algorithm"), soit l'exécuter en deux étapes, la première permettant une convergence grossière, et la seconde plus précise ("two steps algorithm").

Selon les cas, l'utilisateur pourra opter pour l'une ou l'autre, en fonction des difficultés numériques rencontrées. Un indicateur de précision, correspondant à la norme L2 des résidus, est affiché dans la partie résultat de l'écran du pilote (figure 6).

S'il lance les calculs depuis l'écran général des écrans technologiques, l'utilisateur peut de surcroît s'il le désire interrompre les calculs proprement en cliquant sur le bouton "Stop", ce qui lui permet de changer d'option. Attention toutefois, car cette manière d'opérer peut conduire à des erreurs.

5.1.1 Vecteur des variables

Le vecteur des variables est ici le suivant :

[3] Attention : afin de conserver les mêmes indices qu'en Fortran, l'implémentation Java déclare des tableaux de dimension n+1 au lieu de n, l'indice 0 n'étant pas utilisé

```
x[1] = Tcond;
x[2] = Tevap;
x[3] = massFlow;
x[4] = UAevap;
x[5] = UAcond;
x[6] = DTssrefr;
```

5.1.2 Initialisations et appel à l'algorithme de résolution

Les initialisations se font sur la base des valeurs affichées dans l'écran du pilote, pour les surfaces d'échange de l'évaporateur et du condenseur, la vitesse de rotation et la cylindrée du compresseur, ainsi que la température extérieure. Les autres valeurs sont celles des écrans du simulateur.

```
 //lecture à l'écran du pilote des paramètres et variables, éventuellement modifiés après initialisation
 AdesignEvap=Util.lit_d(AdesignEvap_value.getText());//surface de l'évaporateur
 AdesignCond=Util.lit_d(AdesignCond_value.getText());//surface du condenseur

N_value=Util.lit_d(Nref_value.getText());//vitesse de rotation du compresseur
technoCompr.setN(N_value);
VsValue=Util.lit_d(Vs_value.getText());//cylindrée du compresseur
 technoCompr.setVs(VsValue);

 Tair=Util.lit_d(Ta_value.getText())+273.15;
 inAir.T=Tair;
 inAir.update(UPDATE_T,!UPDATE_P,!UPDATE_X);
 inAir.getProperties();

 amontEvap.getProperties();
 avalEvap.getProperties();
 amontCond.getProperties();
 avalCond.getProperties();
 DTsurch=avalEvap.DTsat;
 DTssrefr=avalCond.DTsat;

algorithmResults.setText("");
```

Comme indiqué plus haut, selon que l'on coche ou non l'option "reinitialize", les temperatures de changement d'état sont celles du simulateur ou déterminées à partir des températures du caloporteur (air) et du frigoporteur (eau glycolée). Ces initialisations permettent notamment d'éviter des inversions de température dans les échangeurs lorsque la recherche se fait loin de l'état de départ.

```
 //initialisation des températures de changement d'état
 if(jcheckReInitialize.isSelected()){//si l'option "reinitialize" est cochée
 Tcond= Tair+10;
 Tevap = Tf-10;
}
else{
 Tcond=refrig.getSatTemperature(avalCond.P, 1);
 Tevap=refrig.getSatTemperature(amontEvap.P, 1);
}
```

L'appel à l'algorithme de résolution se fait, une fois qu'il est initialisé, par :

```
int m = 6; int n = 6;

double fvec[] = new double[m+1];
double x[] = new double[n+1];
 int info[] = new int[2];
int iflag[] = new int[2];
```

```
x[1] = Tcond;
x[2] = Tevap;
x[3] = massFlow;
x[4] = UAevap;
x[5] = UAcond;
x[6] = DTssrefr;

double residu0;
double residu1;

 fcn(m,n,x,fvec,iflag);

residu0 = optimization.Minpack_f77.enorm_f77(m,fvec);

nfev2 = 0; njev2 = 0;

double epsi=0.0005;//précision globale demandée sur la convergence
double epsfcn = 1.e-6;//précision calcul des différences finies

 if(twoStepAlgorithm.isSelected()){
 epsi=0.01;
 }

 //appel modifié de lmdiff avec modification précision calcul des différences finies
optimization.Minpack_f77.lmdif2_f77(this, m, n, x, fvec, epsi, epsfcn, info);
residu1 = optimization.Minpack_f77.enorm_f77(m,fvec);

if(twoStepAlgorithm.isSelected()){
 epsi=0.00005;
 epsfcn = 1.e-9;//précision calcul des différences finies

 //appel modifié de lmdiff avec modification précision calcul des différences finies
 optimization.Minpack_f77.lmdif2_f77(this, m, n, x, fvec, epsi, epsfcn, info);
 residu1 = optimization.Minpack_f77.enorm_f77(m,fvec);
}
```

5.1.3 Fonction fcn()

Pour pouvoir estimer les résidus, il faut, comme le montre le code ci-dessous, commencer par d'une part mettre à jour les variables de Thermoptim correspondant au vecteur x, et d'autre part calculer les variables liées.

Comme indiqué dans fcn(), les fonctions fvec sont six méthodes resEvap(), recCond et resFlow(), resAevap(), resAcond() et resLoad() définies ci-dessous.

L'appel aux trois premières se fait avant recalcul du simulateur et des TechnoDesign, et celui aux autres ensuite.

```
public void fcn(int m, int n, double x[], double fvec[], int iflag[]) {

if (iflag[1]==1) this.nfev++;
if (iflag[1]==2) this.njev++;

//mise à jour des variables "physiques" pour une meilleure compréhension du code
 Tcond=x[1];
 Tevap=x[2];
 massFlow=x[3];
 UAevap=x[4];
 UAcond=x[5];
 DTssrefr=x[6];
```

La première étape consiste à mettre à jour tous les points et transfos du modèle pour que les calculs des résidus soient effectués sur la base des valeurs du vecteur x. Les pertes de charge sont affectées pour moitié au point amont et pour moitié au point aval du condenseur, et pour totalité de l'évaporateur.

```
 //mise à jour du point aval du condenseur (avec modification du sous-refroidissement)
 Pcond=refrig.getSatPressure(Tcond, 1);
 avalCond.T=Tcond+DTssrefr;// DTssrefr est négatif
 avalCond.P=Pcond-dPcond/2;
 avalCond.T=refrig.getSatTemperature(avalCond.P,1)+DTssrefr;// DTssrefr est négatif
 avalCond.DTsat=DTssrefr;
 avalCond.update(UPDATE_T, UPDATE_P, !UPDATE_X, false, UPDATE_DTSAT, false, "", 0.);
 avalCond.getProperties();

 //mise à jour du point aval de l'évaporateur (avec modification de la surchauffe)
 Pevap=refrig.getSatPressure(Tevap, 1);
 avalEvap.P=Pevap-dPevap;
 avalEvap.T=refrig.getSatTemperature(Pevap-dPevap,1)+DTsurch;
 avalEvap.DTsat=DTsurch;
 avalEvap.update(UPDATE_T, UPDATE_P, !UPDATE_X, false, UPDATE_DTSAT, false, "", 0.);
 avalEvap.getProperties();

 //mise à jour de la pression du point aval du compresseur
 Pcond=refrig.getSatPressure(Tcond, 1);
 amontCond.P=Pcond;
 amontCond.update(!UPDATE_T,UPDATE_P,!UPDATE_X);
 amontCond.getProperties();

 //mise à jour de l'état du point amont de l'évaporateur
 //le recalcul du titre est fait après convergence
 amontEvap.P=Pevap;
 amontEvap.T=Tevap;
 amontEvap.update(UPDATE_T,UPDATE_P,!UPDATE_X);

 //mise à jour des variables liées
 DeltaHevap=getDeltaHEvap();//directement déterminé à partir de x
 DeltaHcond=DeltaHevap+getTauCompr();//modifie la température de sortie compresseur
```

Une fois ces mises à jour effectuées, les trois premiers résidus correspondant à l'équilibrage du cycle de réfrigération sont calculés.

```
 //calcul des premiers résidus
fvec[1] = resEvap();
fvec[2] = resCond();
fvec[3] = resFlow();
```

Le cycle étant équilibré, le projet est recalculé un certain nombre de fois, ainsi que les échangeurs :

```
//mises à jour du simulateur avant recalcul des TechnoDesign
//on les exécute 3 fois par sécurité, mais ce n'est pas optimisé
 for(int j=0;j<3;j++)proj.calcThopt();
 updateHx(evaporatorName, RECALCULATE, !UPDATE_UA, 0, !UPDATE_EPSI, 0, !UPDATE_DTMIN, 0);
 updateHx(condenserName, RECALCULATE, !UPDATE_UA, 0, !UPDATE_EPSI, 0, !UPDATE_DTMIN, 0);
 updateHx(evaporatorName, RECALCULATE, !UPDATE_UA, 0, !UPDATE_EPSI, 0, !UPDATE_DTMIN, 0);
 updateHx(condenserName, RECALCULATE, !UPDATE_UA, 0, !UPDATE_EPSI, 0, !UPDATE_DTMIN, 0);
 updateHx(evaporatorName, RECALCULATE, !UPDATE_UA, 0, !UPDATE_EPSI, 0, !UPDATE_DTMIN, 0);
 updateHx(condenserName, RECALCULATE, !UPDATE_UA, 0, !UPDATE_EPSI, 0, !UPDATE_DTMIN, 0);
```

Tous les PointThopt sont actualisés, et les auxiliaires mis à jour :

```
 amontEvap.getProperties();
```

```
 avalEvap.getProperties();
 amontCond.getProperties();
 avalCond.getProperties();
 waterInlet.getProperties();
 waterOutlet.getProperties();
 inAir.getProperties();
 outAir.getProperties();

 //mise à jour des auxiliaires

 outFanAir.P=inAir.P+dPair;
 outFanAir.update(!UPDATE_T,UPDATE_P,!UPDATE_X);
 updateprocess(fanName, "Compression",RECALCULATE,!IS_SET_FLOW, !UPDATE_FLOW, 0,
!UPDATE_ETA, 0);
 outFanAir.getProperties();

 waterPumpOutlet.P=waterInlet.P+dPwater;
 waterPumpOutlet.update(!UPDATE_T,UPDATE_P,!UPDATE_X);
 updateprocess(pumpName, "Compression",RECALCULATE,!IS_SET_FLOW, !UPDATE_FLOW, 0,
!UPDATE_ETA, 0);
 waterPumpOutlet.getProperties();
```

Les TechnoDesign des échangeurs sont alors recalculés, ce qui met à jour les pertes de charge et permet d'estimer les surfaces d'échange et la charge de frigorigène correspondant à ce nouvel état :

```
 technoEvap.makeDesign();
 technoCond.makeDesign();
 AcalculatedEvap_value.setText(technoEvap.ADesign_value.getText());
 AcalculatedCond_value.setText(technoCond.ADesign_value.getText());

//calcul des pertes de charge
dPevap=technoEvap.techf.getPressureDrop()*dPmult;
 dPcond=technoCond.tech.getPressureDrop()*dPmult;
```

Enfin, les trois derniers résidus sont estimés :

```
 //calcul des derniers résidus après recalcul des TechnoDesign (adimensionnés)
fvec[4] = resAevap();
fvec[5] = resAcond();
fvec[6] = resLoad();

return;
}
```

5.1.4 Fonctions de résidu

Nous nous contenterons ici de donner deux exemples de résidus, relatifs à l'équilibrage du condenseur et au calcul de sa surface d'échange. Les autres sont présentés section 6. Dans les deux cas, le résidu a été normé pour équilibrer les poids des différentes fonctions d'écart, en utilisant une méthode simple, mais valable uniquement si la valeur à atteindre n'est pas nulle, ce qui est toujours le cas ici.

```
 double resCond(){//équation 17.4.5
 //renvoie la différence entre la température de condensation
 //recalculée à partir de la méthode du NUT et Tcond=x[1]

 mCpRefrigCond=DeltaHcond/(amontCond.T-(Tcond+DTssrefr));

 double[]res=Util.epsi_NUT(mCpRefrigCond,mCpCalopCond,UAcond);
 epsilon=res[0];
 mCpmin=res[1];
```

```
 double Tentree_refrig=outFanAir.T+DeltaHcond/epsilon/mCpmin;
 double Tsortie_refrig=Tentree_refrig-DeltaHcond/mCpRefrigCond;

 //le résidu est l'écart entre les deux températures de condensation
 double z= Tcond-(Tsortie_refrig-DTssrefr);// DTssrefr est négatif
 z= 2*z/(Tcond+Tsortie_refrig-DTssrefr);
 return z;
}

double resAcond(){//renvoie le résidu de la surface du condenseur
 UAcond_value.setText(Util.aff_d(UAcond,4));
 AcalculatedCond_value.setText(Util.aff_d(AcondReel,4));
 AcondReel=technoCond.A;
 double z= AcondReel-AdesignCond;

 z= 2*z/(AcondReel+AdesignCond);
 return z;
}
```

5.2 Affichage des résultats du pilote après calcul en non-nominal

La précision de la solution est écrite dans le fichier texte "output.txt", et l'écran du pilote est mis à jour.

```
 System.out.println();
 System.out.println(" Initial L2 norm of the residuals: " + residu0);
 System.out.println("Final L2 norm of the residuals: " + residu1);
 System.out.println("Number of function evaluations: " + nfev);
 System.out.println("Number of Jacobian evaluations: " + njev);
 System.out.println("Info value: " + info[1]);
 System.out.println("Final approximate solution: " + x[1] + ", " + x[2]+ ", " + x[3] );
 System.out.println(); /**/

algorithmResults.setText("Algorithm precision: " + Util.aff_d(residu1,8));

 UAevap_value.setText(Util.aff_d(UAevap,4));
 UAcond_value.setText(Util.aff_d(UAcond,4));
 AcalculatedEvap_value.setText(technoEvap.ADesign_value.getText());
 AcalculatedCond_value.setText(technoCond.ADesign_value.getText());
 COP_value.setText(Util.aff_d(DeltaHevap/tauCompr,4));
 DeltaHevap_value.setText(Util.aff_d(DeltaHevap,4));
 massFlow_value.setText(Util.aff_d(massFlow,4));
 DeltaHcond_value.setText(Util.aff_d(DeltaHcond,4));
 Pevap_value.setText(Util.aff_d(Pevap,4));
 Pcond_value.setText(Util.aff_d(avalCond.P,4));
 tauCompr_value.setText(Util.aff_d(tauCompr,4));
```

6 Equations de la machine frigorifique en non-nominal

6.1 Bilan de l'évaporateur

La température d'évaporation est fixée par l'équilibre thermique de l'évaporateur, lequel dépend essentiellement d'une part de la température T_f et du débit du frigoporteur (eau glycolée dans cet exemple), et d'autre part du débit de frigorigène.

$$\Delta H_{evap} = \dot{m} \, (h(T_e + \Delta T_{surch}, P_e) - h(T_c - \Delta T_{ssrefr}, P_c)) = U_e \, A_e \, \Delta T_{ml\_ef} \qquad (8)$$

ΔH_{evap} est calculé par :

```
double getDeltaHEvap(){//calcule la puissance thermique de l'évaporateur
 return massFlow*(avalEvap.H-avalCond.H);
}
```

L'équilibrage de l'échangeur se fait en recalculant la température d'évaporation par la méthode du NUT :

```
 double resEvap(){//équation 17.4.3
 //renvoie la différence entre la température d'évaporation,
 //recalculée à partir de la méthode du NUT et Tevap=x[2]

 mCpRefrigEvap=DeltaHevap/DTsurch;

 double[]res=Util.epsi_NUT(mCpRefrigEvap,mCpCalopEvap,UAevap);
 epsilon=res[0];
 mCpmin=res[1];

 double z=Tevap-waterPumpOutlet.T+DeltaHevap/epsilon/mCpmin;//résidu

 z= 2*z/(Tevap+Tf-DeltaHevap/epsilon/mCpmin);
 return z;
 }
```

6.2 Bilan du compresseur

Le bilan du compresseur exprime que le travail de compression est égal au produit du débit par le travail massique de compression isentropique, divisé par le rendement isentropique. Il dépend lui aussi de plusieurs grandeurs et est donné par :

$$\tau = \dot{m} \frac{\Delta h_s (T_e, P_e, P_c)}{\eta_s (P_c/P_e)} \qquad (9)$$

Il est calculé par :

```
 double getTauCompr(){//équation 17.4.4

 // recalcul du compresseur
 double eta_is=technoCompr.getRisentr();
 updateprocess(compressorName, "Compression",RECALCULATE,IS_SET_FLOW, UPDATE_FLOW,
massFlow, UPDATE_ETA, eta_is);
 amontCond.getProperties();

 //ce qui permet de connaître la puissance consommée
 String[] args=new String[2];
 args[0]="process";
 args[1]=compressorName;
 Vector vProp=proj.getProperties(args);
 Double f=(Double)vProp.elementAt(4);
 tauCompr=f.doubleValue();

 return tauCompr;
 }
```

6.3 Premier principe

Le premier principe s'écrit :

$$\Delta H_{cond} = \tau + \Delta H_{evap} \qquad (10)$$

Il est écrit dans fcn() sous la forme :

```
DeltaHcond=DeltaHevap+getTauCompr();//modifie la température de sortie compresseur
```

6.4 Bilan du condenseur

De manière analogue à ce que nous avons exposé pour l'évaporateur, la température de condensation est fixée par l'équilibre thermique du condenseur, lequel dépend essentiellement d'une part de la température et du débit de l'air de refroidissement, d'autre part du débit de frigorigène, et enfin de la température de sortie condenseur.

La température de sortie condenseur dépend ainsi du rapport de compression et du rendement isentropique du compresseur, lui-même aussi fonction de ce rapport.

$$\Delta H_{cond} = U_c \, A_c \, \Delta T_{ml\_ac} \qquad (11)$$

L'équilibrage de l'échangeur se fait en recalculant la température de condensation par la méthode du NUT. Le code a déjà été donné section 5.1.4.

6.5 Conservation du débit massique

La résolution de l'équation (7) se fait en comparant la valeur qu'elle fournit (implémentée dans le TechnoDesign du compresseur) avec celle de massFlow = x[3]. Le calcul de λ est fait dans le TechnoDesign.

$$\dot{m} = \frac{\dot{V}}{v} = \frac{\lambda \, N \, V_s}{60 \, v} \qquad (7)$$

```
double resFlow(){//équations 17.4.1 et 17.4.2
//renvoie le résidu du débit-masse

Vevap=avalEvap.V;
double rCompr=amontCond.P/avalEvap.P;
double y=technoCompr.getMassFlow(Vevap, rCompr);
double z= massFlow-y;
z= 2*z/(massFlow+y);
return z;
}
```

6.6 Conservation de la charge de frigorigène

Le calcul de la masse de frigorigène dans les échangeurs diphasiques exige des précautions particulières, car on ne connaît pas avec précision la répartition des phases liquide et vapeur.

En mode dimensionnement, on choisit un sous-refroidissement de référence (par exemple 5 K) et on en déduit la masse de réfrigérant du circuit : évaporateur, condenseur et ligne liquide (amont détendeur). On peut négliger la masse de réfrigérant contenue dans le compresseur ainsi qu'entre l'évaporateur et le compresseur.

En mode non-nominal, le sous-refroidissement est calculé pour que la masse de réfrigérant du circuit soit conservée.

Le taux de vide ε est défini par la section occupée par la vapeur par rapport à la section totale de l'échangeur. Sa connaissance est indispensable pour prédire la charge du système frigorifique. En effet la masse totale dans un échangeur peut s'exprimer en fonction de la masse volumique moyenne :

$$\rho_m = \varepsilon \rho_v + (1-\varepsilon)\rho_l \qquad (12)$$

ε, fonction d'une constante Kh et du glissement S des phases liquide et vapeur, est donné par :

$$\varepsilon = \frac{K_H}{1 + S\frac{1-x}{x}\frac{\rho_v}{\rho_l}} \qquad (13)$$

Le calcul de la masse volumique moyenne est effectué en intégrant (13) entre les titres d'entrée et de sortie de l'échangeur.

La masse contenue dans la ligne liquide est quant à elle déterminée en considérant une longueur de tube de même section que celle du condenseur, qui est entrée dans l'écran du TechnoDesign de la figure 4. Le code est donné ci-dessous :

```
//mises à jour pour la charge de frigorigène
mEvap=technoEvap.techf.getFluidLoad();
mCond=technoCond.techc.getFluidLoad();
mLigne=volLigne/avalCond.V;
mLine_value.setText(Util.aff_d(mLigne,4));
double load=mCond+mEvap+mLigne;
```

7 Utilisation du pilote

L'écran complet du pilote est donné figure 6. Il permet de modifier d'une part les surfaces des échangeurs et la longueur de la ligne liquide, et d'autre part la température d'air, la cylindrée ou la vitesse de rotation.

Entrez la température d'air ou la vitesse de rotation que vous souhaitez modifier, et, soit cliquez sur "Calculate", soit ouvrez l'écran des TechnoDesign depuis le simulateur, puis cliquez sur "Calculer le pilote". Cette deuxième manière de faire est préférable, car elle permet de suivre la convergence tout en gardant la main sur Thermoptim, pour afficher des valeurs intermédiaires ou modifier les calculs du pilote.

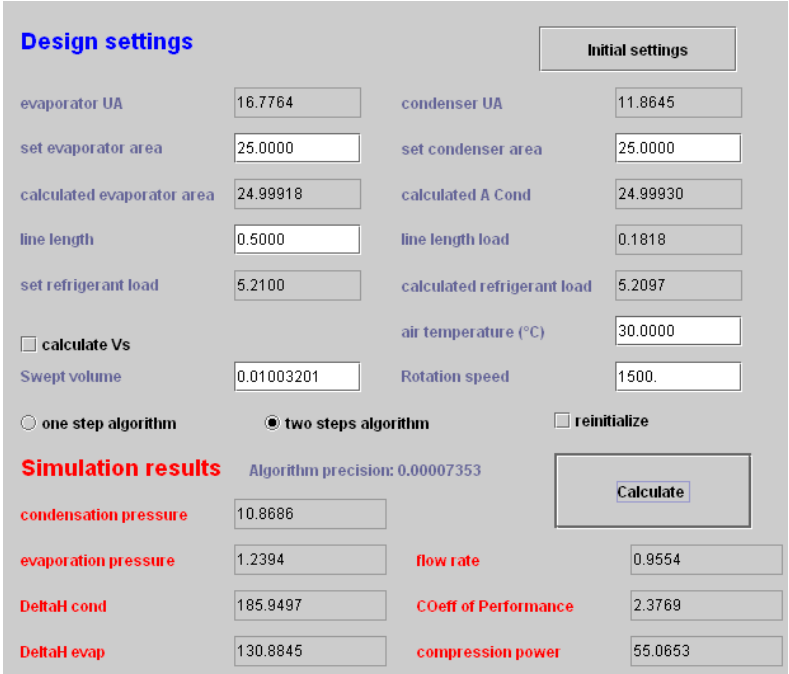

Figure 6 : Ecran du pilote

Les résultats sont affichés à l'écran une fois la convergence obtenue. Si les valeurs que vous entrez sont très différentes de celles de dimensionnement, des erreurs de calcul de Thermoptim peuvent être générées, avec messages. Si nécessaire, choisissez une valeur de recalcul plus proche de la valeur initiale.

La figure 7 montre les résultats de simulation obtenus lorsque l'on fait varier la température de l'air de refroidissement, pour le paramétrage des écrans technologiques retenu précédemment. L'influence de la vitesse de rotation est donnée figure 8.

Dans cet exemple, nous n'avons fait varier que deux variables, mais il serait très simple d'étudier l'influence des débits d'air et d'eau glycolée, ou de la température de cette dernière, soit en modifiant leurs valeurs dans les écrans du simulateur avant recalcul avec ce pilote, soit en le modifiant pour que ces valeurs apparaissent dans l'écran de pilotage et soient ensuite automatiquement mises à jour.

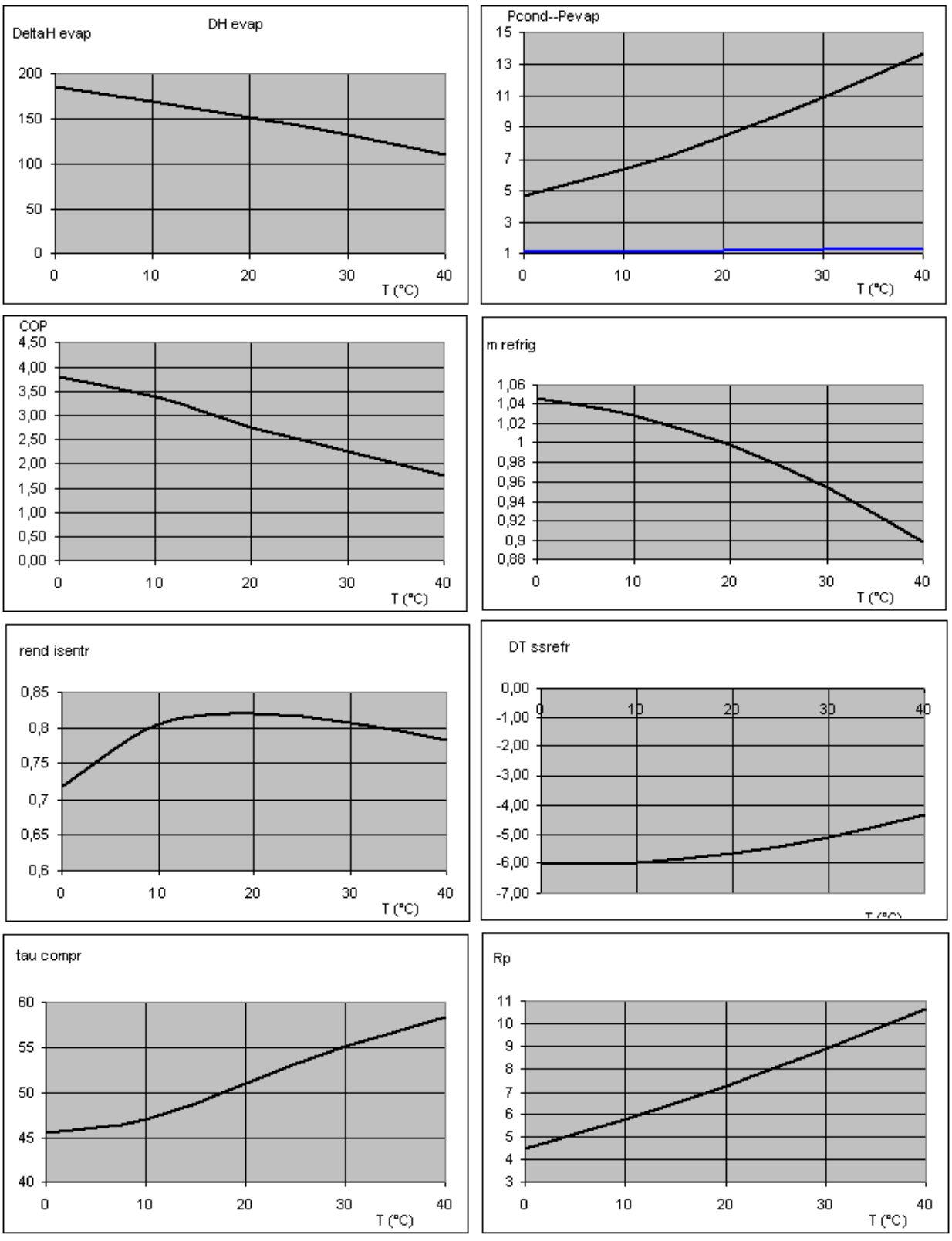

Figure 7 : Influence de la température de l'air extérieur (compresseur volumétrique, N = 1500 tr/mn)

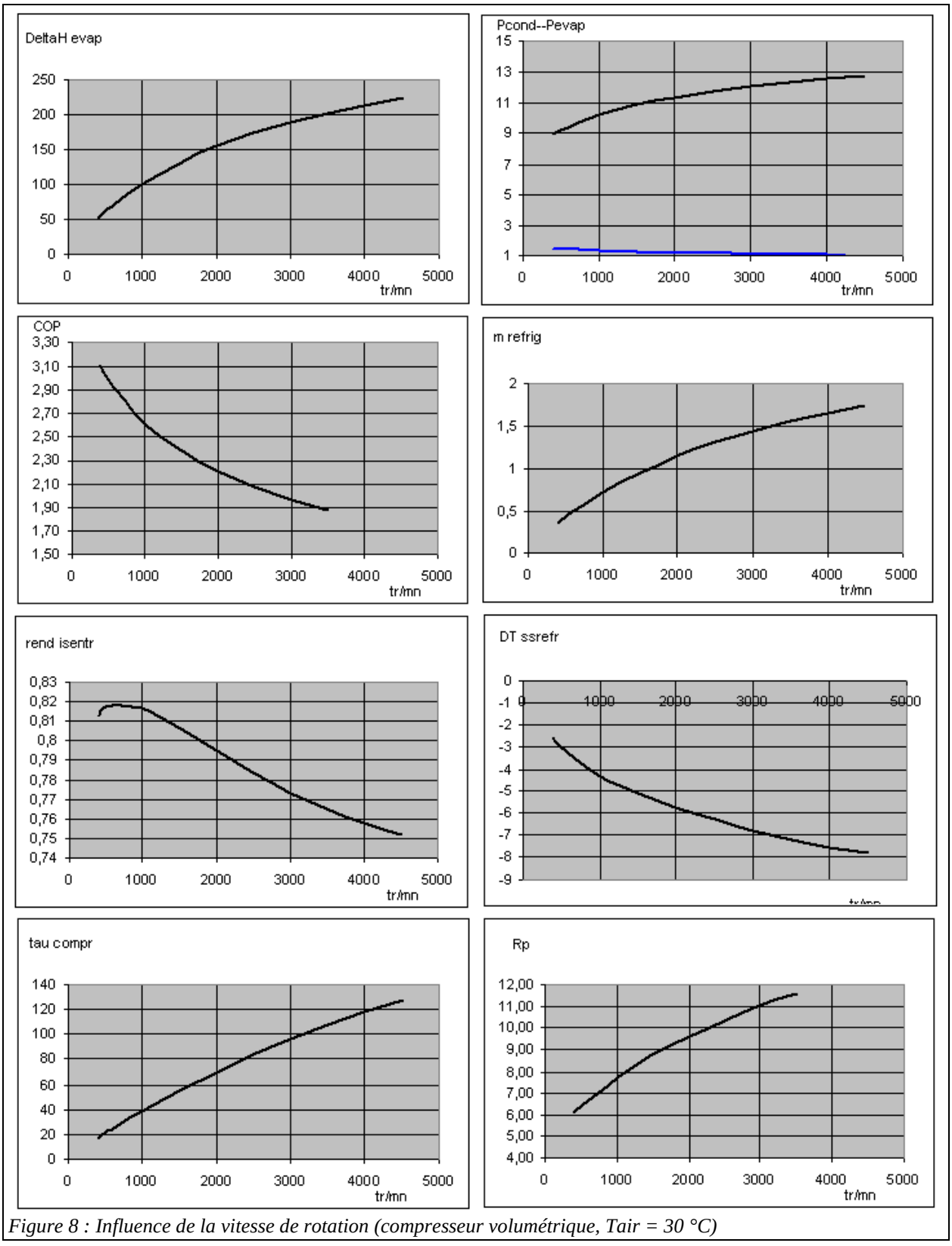

Figure 8 : Influence de la vitesse de rotation (compresseur volumétrique, Tair = 30 °C)

8 Cycle avec compresseur centrifuge

8.1 Calcul des turbocompresseurs en régime non-nominal

Lorsque le compresseur utilisé n'est plus volumétrique mais centrifuge, les équations (4) à (7) ne sont plus valables, et doivent être remplacées par une représentation aussi précise de la cartographie du turbocompresseur. Les caractéristiques du turbocompresseur sont données par les équations (14) et (15) :

$$Rp = \frac{P_r}{P_a} = f(\dot{m}_c) \qquad (14)$$

$$\eta_s = f(\dot{m}_c) \qquad (15)$$

Ces équations sont mises sous forme adimensionnelle et exprimées numériquement comme expliqué dans la note ModelisationSimplifieeTurboCompresseurAdim.doc et dans le tome 4 du manuel de référence de Thermoptim (cf. figures 9 et 10).

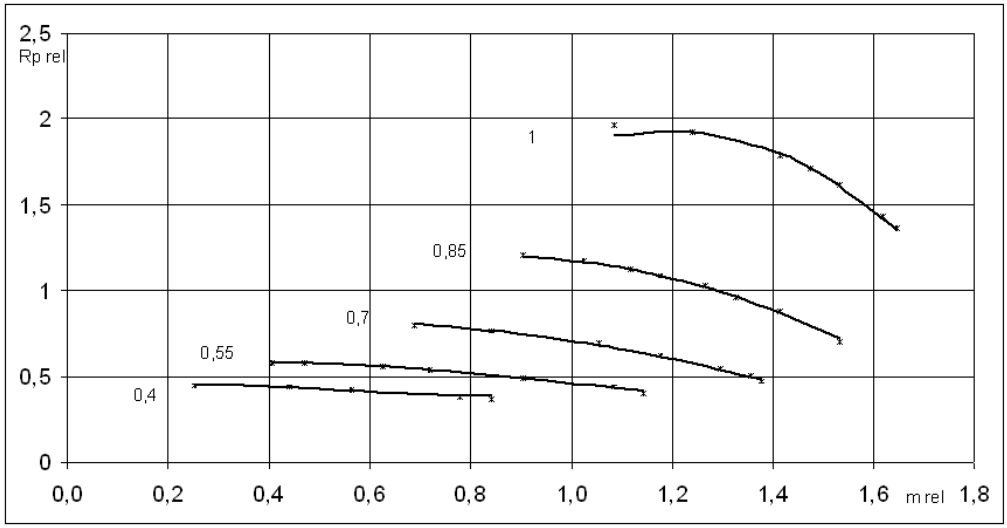

Figure 9 : Caractéristiques (rapport de compression, débit) adimensionnelles

Le débit massique dépend donc de l'état thermodynamique à l'aspiration et du rapport de compression.

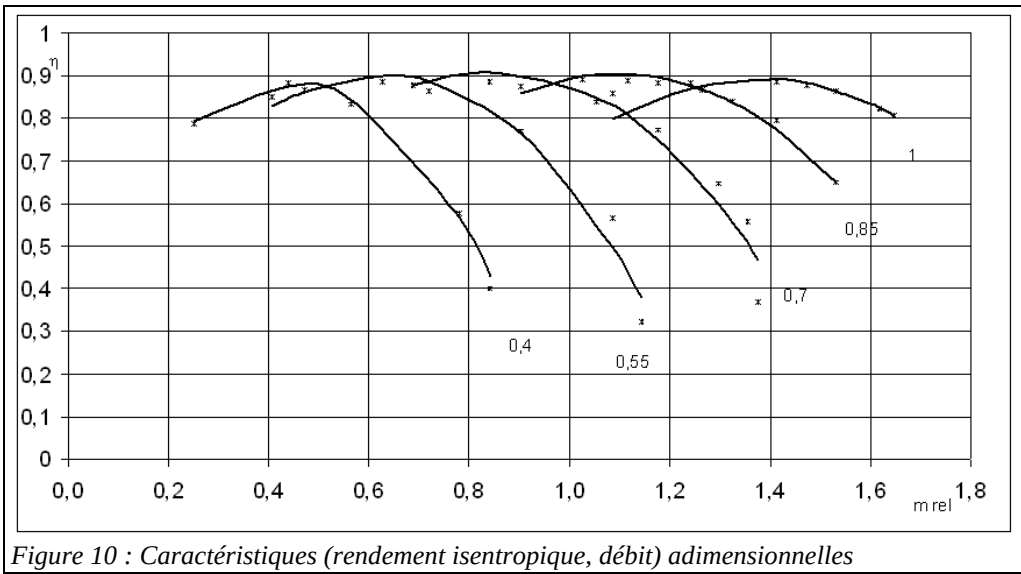

Figure 10 : Caractéristiques (rendement isentropique, débit) adimensionnelles

8.2 Modifications du code

Les modifications à apporter au code du pilote sont tout à fait mineures, la plupart des méthodes étant implémentées dans les classes externes MappedTurboCompr (TechnoDesign) et TurboComprMapDataFrame (cartographie). L'instanciation du TechnoDesign est changée en conséquence.

technoCompr=new MappedTurboCompr(proj, compressorName, avalEvap, amontCond);

8.2.1 Initialisations de dimensionnement du compresseur

Lors de l'initialisation du compresseur, on charge la cartographie appropriée, puis on recherche la vitesse de rotation conduisant au rapport de compression nominal, compte tenu des conditions d'entrée :

```
technoCompr.setDataFile(dataFile);
technoCompr.makeDesign();

//recherche de la vitesse de rotation correspondant à Rp et massFlow
double Ninit=technoCompr.getNfromRpAndFlow(Rp,
massFlow)/technoCompr.racT0*Math.pow(avalEvap.T,0.5)*technoCompr.Nref;
if(Ninit!=0){//si on est en dehors de la cartographie, on ne fait rien en dehors du message d'information
technoCompr.setN(Ninit/technoCompr.Nref);
technoCompr.setNparameters();//calcule les paramètres qui dépendent de N
N_value=Ninit;
Nref_value.setText(Util.aff_d(Ninit,4));
}
```

8.2.2 Initialisations des calculs en non-nominal

Avant de lancer les calculs, on lit à l'écran la valeur de la vitesse de rotation choisie, et on met à jour le TechnoDesign en l'adimensionnant:

```
N_value=Util.lit_d(Nref_value.getText())/technoCompr.Nref;
technoCompr.setDesignValues();
technoCompr.setN(N_value);
technoCompr.setNparameters();
```

8.2.3 Actualisation de la vitesse de rotation réduite

Pendant les mises à jour qui prennent place dans la fonction fcn(),la vitesse de rotation réduite est actualisée une fois les nouvelles conditions d'aspiration connues :

```
//mise à jour de la vitesse de rotation corrigée du compresseur
technoCompr.updateN();
technoCompr.setNparameters();
```

8.2.4 Vérification de l'adaptation à la cartographie de la vitesse de rotation réduite

Une fois les calculs terminés, on vérifie si la vitesse de rotation réduite reste située dans les limites admissibles pour la cartographie choisie. Sinon l'utilisateur est averti.

```
technoCompr.checkMapValidity();
```

8.3 Résultats du modèle

Les résultats du modèle sont tout à fait cohérents avec ceux obtenus avec le compresseur volumétrique pour ce qui concerne l'influence de la température extérieure, les seules différences étant dues au changement de caractéristiques (cf. figure 11).

Celle de la vitesse de rotation est quant à elle assez différente de celle obtenue avec le compresseur volumétrique (cf. figure 12).

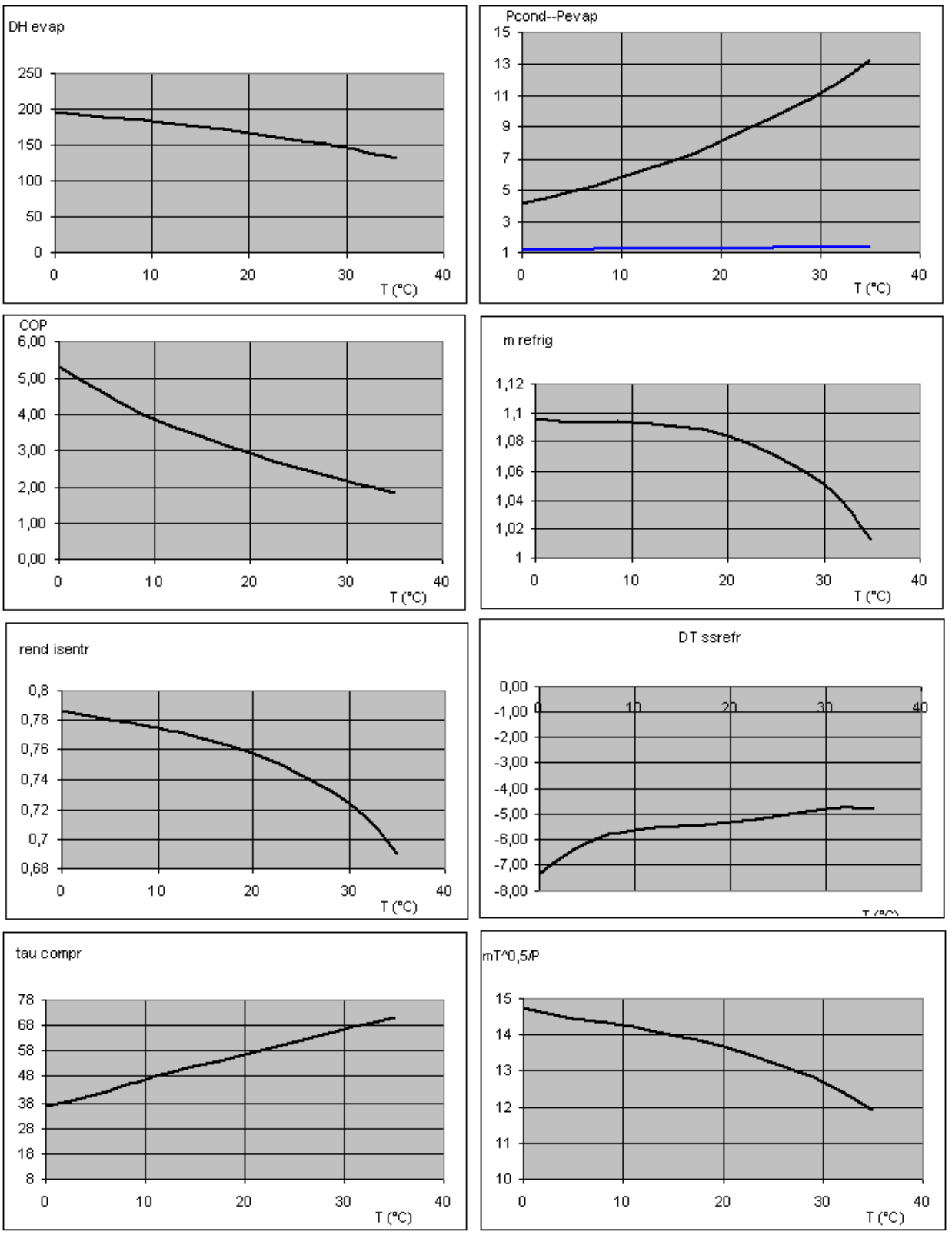

Figure 11 : Influence de la température de l'air extérieur (compresseur centrifuge, N = 19 000 tr/mn)

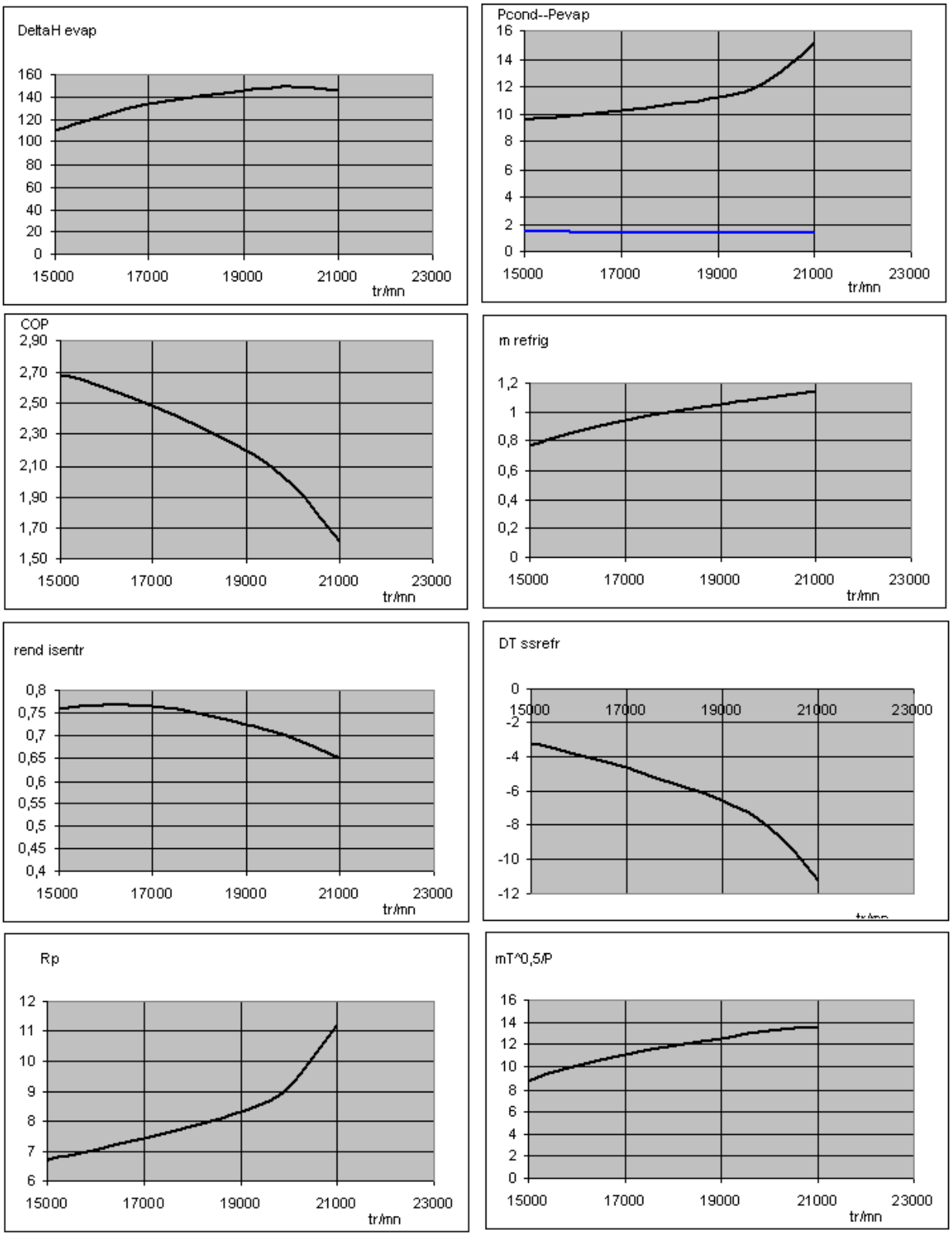

Figure 12 : Influence de la vitesse de rotation (compresseur centrifuge, Tair = 30 °C)

Annexe 1 : Principe de calcul des échangeurs multizones

Evaporateurs : fluide froid diphasique et fluide chaud en sensible

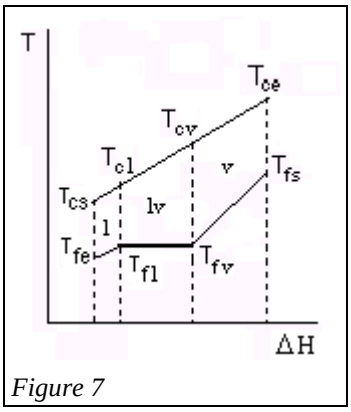

Figure 7

Les équations sont les suivantes (figure 7) :

$m_c Cp_c (T_{ce} - T_{cv}) = m_f Cp_{fv} (T_{fs} - T_{fv})$
$m_c Cp_c (T_{cv} - T_{cl}) = m_f Cp_{flv} (T_{fv} - T_{fl}) = m_f L_f$
$m_c Cp_c (T_{cl} - T_{cs}) = m_f Cp_{fl} (T_{fl} - T_{fe})$

Les relations donnant epsilon et R sont alors :

$$\varepsilon_v = \frac{T_{fs} - T_{fv}}{T_{ce} - T_{fv}} \quad R_v = \frac{m_f Cp_{fv}}{m_c Cp_c}$$

$$\varepsilon_{lv} = \frac{T_{cv} - T_{cl}}{T_{cv} - T_{fe}} \quad R_{lv} = \frac{m_c Cp_c}{m_f Cp_{flv}}$$

$$\varepsilon_l = \frac{T_{fl} - T_{fe}}{T_{cl} - T_{fe}} \quad R_l = \frac{m_f Cp_{fl}}{m_c Cp_c}$$

Si le fluide entre à l'état diphasique dans l'évaporateur, les équations sont légèrement différentes.

Condenseurs à entrée vapeur : fluide chaud diphasique et fluide froid en sensible

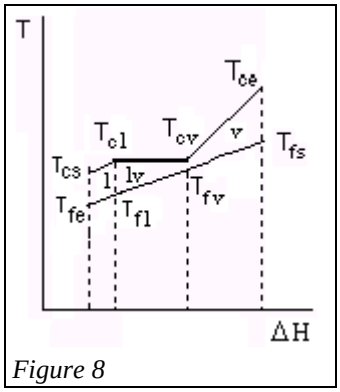

Figure 8

Les équations sont les suivantes (figure 8) :

$m_c Cp_{cv} (T_{ce} - T_{cv}) = m_f Cp_f (T_{fs} - T_{fv})$
$m_c Cp_{clv} (T_{cv} - T_{cl}) = m_f Cp_f (T_{fv} - T_{fl}) = m_c L_c$
$m_c Cp_{cl} (T_{cl} - T_{cs}) = m_f Cp_f (T_{fl} - T_{fe})$

Les relations donnant epsilon et R sont alors :

$$\varepsilon_v = \frac{T_{ce} - T_{cv}}{T_{ce} - T_{fv}} \quad R_v = \frac{m_c Cp_{cv}}{m_f Cp_f}$$

$$\varepsilon_{lv} = \frac{T_{fv} - T_{fl}}{T_{cv} - T_{fl}} \quad R_{lv} = \frac{m_f Cp_f}{m_c Cp_{clv}}$$

$$\varepsilon_l = \frac{T_{cv} - T_{cs}}{T_{cv} - T_{fe}} \quad R_l = \frac{m_c Cp_{cl}}{m_f Cp_f}$$

Condenseurs à entrée diphasique : fluide chaud diphasique et fluide froid en sensible

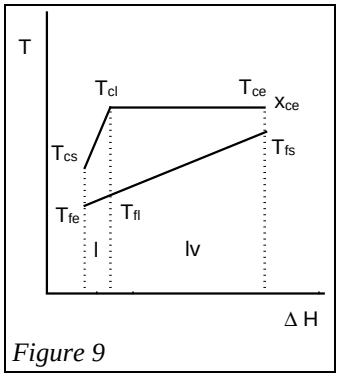

Figure 9

Les équations sont les suivantes (figure 9) :

$m_c Cp_{clv} (T_{ce} - T_{cl}) = m_f Cp_f (T_{fs} - T_{fl}) = m_c L_c x_{ce}$
$m_c Cp_{cl} (T_{cl} - T_{cs}) = m_f Cp_f (T_{fl} - T_{fe})$

Les relations donnant epsilon et R sont alors :

$$\varepsilon_{lv} = \frac{T_{fs} - T_{fl}}{T_{ce} - T_{fl}} \quad R_{lv} = \frac{m_f Cp_f}{m_c Cp_{clv}}$$

$$\varepsilon_{l} = \frac{T_{ce} - T_{cs}}{T_{ce} - T_{fe}} \quad R_{l} = \frac{m_c Cp_{cl}}{m_f Cp_f}$$