T H E R M O P T I M ®

NON NOMINAL

PILOTE POUR CYCLE A VAPEUR SIMPLE

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.

La turbine utilisée sera d'abord modélisée par une loi de Stodola (classe de pilotage PiloteVapeurSimple), puis (classe de pilotage PiloteVapeurCarto, avec une cartographie définie dans le fichier dataSingleStageTurbine.txt).

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).

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 :

[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.

- 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 turbines en régime non-nominal

Le modèle de turbine que nous considèrerons ici est basé sur ceux décrits section 4.5 du tome 1. Il repose sur l'hypothèse que le comportement des turbines adiabatiques peut être représenté avec une précision raisonnable par deux paramètres : le rendement isentropique classique η_s, et une grandeur K_0 appelée constante de cône ou de Stodola, qui caractérise le débit nominal.

Pour le rendement isentropique, l'allure des caractéristiques montre qu'une représentation par des courbes polynomiales fonction du rapport de détente peut permettre une bonne précision, par exemple une équation très simple à identifier, du type $y = a + b\,x + c\,x^2$ (4), ou bien une plus complexe (5) analogue à la loi de Dehausse souvent utilisée pour des compresseurs volumétriques.

$$\eta_s = K_1 + K_2 \cdot \frac{P_r}{P_a} + K_3 \cdot \left(\frac{P_r}{P_a}\right)^2 \qquad (4)$$

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

La règle dite du cône, due à Stodola s'exprime sous la forme (6) :

$$\frac{\dot{m}\sqrt{T_{in}}}{P_{in}} = K_0 \sqrt{1 - \left[\frac{P_{out}}{P_{in}}\right]^{(k+1)/k}} \qquad (6)$$

Elle n'est valable que tant que le débit reste inférieur à une valeur limite, qui est atteinte lorsque les conditions soniques s'établissent au col des tuyères du stator.

Pour un étage de turbine, le rapport critique est donné (pour l'isentropique) par l'équation (7). Pour n étages, il est élevé à la puissance n.

$$\frac{P_{in}}{P_{out}} = \left[\frac{2}{\gamma + 1}\right]^{(\gamma + 1)/2(\gamma - 1)} \qquad (7)$$

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

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

3 Choix et paramétrage des écrans technologiques

Supposons que l'on veuille dimensionner un cycle à vapeur simple (figure 1) pour qu'il fournisse une puissance de 660 MW environ pour une température de refroidissement de 8 °C, une pression maximale de cycle de 165 bars et une température de surchauffe de 560 °C, le rendement isentropique de la turbine étant voisin de 0,85.

Pour le condenseur, le débit de refroidissement est de 18 t/s.

L'efficacité de la machine vaut dans ces conditions 0,336.

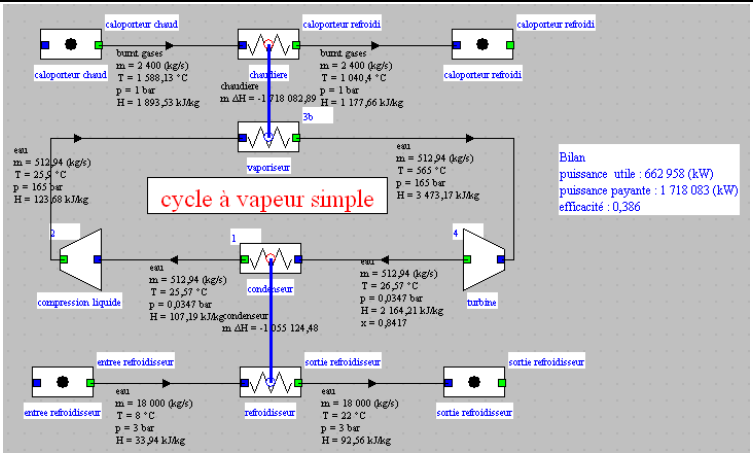

Figure 1 : Modèle de cycle de centrale à vapeur

Les pertes de charge seront ici négligées, bien qu'elles soient calculées, et les pertes par vitesse restante en sortie de turbine ne seront prises en compte que dans un second temps (cf. section 8).

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 la chaudière et le condenseur sont chacun modélisés par un seul échangeur multizone.

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.

[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.

```
//initialisations des PointThopt et des TechnoDesign
//attention : les noms des points et composants doivent être exacts, sous peine de générer une erreur empêchant le chargement des TechnoDesign
amontChaudiere=new PointThopt(proj,"2");
avalChaudiere=new PointThopt(proj,"3b");
amontCond=new PointThopt(proj,"4");
avalCond=new PointThopt(proj,"1");
hotGases=new PointThopt(proj,"caloporteur chaud");
coldGases=new PointThopt(proj,"caloporteur refroidi");
inRiver=new PointThopt(proj,"entree refroidisseur");
outRiver=new PointThopt(proj,"sortie refroidisseur");
avalCond.getProperties();
fluideThermo=(rg.corps.Corps)avalCond.lecorps;

//noms des composants
        boilerName="chaudiere";
        condenserName="condenseur";
        turbineName="turbine";
```

Les TechnoDesign sont du type TechnoEvaporator, TechnoCondensor et TechnoSimpleTurb, trois classes spécifiquement créées pour ce type de composants. Les deux premières permettent de calculer la chaudière et le condenseur comme des échangeurs multizones, selon les équations données en annexe 1. La troisième implémente les équations (4) à (6). 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.

```
//instanciation des TechnoDesign dans les classes externes
technoChaudiere=new TechnoEvaporator(proj, boilerName, hotGases, coldGases, amontChaudiere, avalChaudiere);
addTechnoVector(technoChaudiere);
technoCond=new TechnoCondensor(proj, condenserName, amontCond, avalCond, inRiver, outRiver);
addTechnoVector(technoCond);

technoTurbine= new TechnoSimpleTurb(proj, turbineName, avalChaudiere, amontCond);
addTechnoVector(technoTurbine);

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

La dernière ligne du code ci-dessus permet de transférer ces écrans technologiques dans le noyau du progiciel.

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 constante de Stodola, la température et la pression maximales du cycle et la température du fluide de refroidissement, et possède des options de guidage de l'algorithme qui seront précisées plus loin.

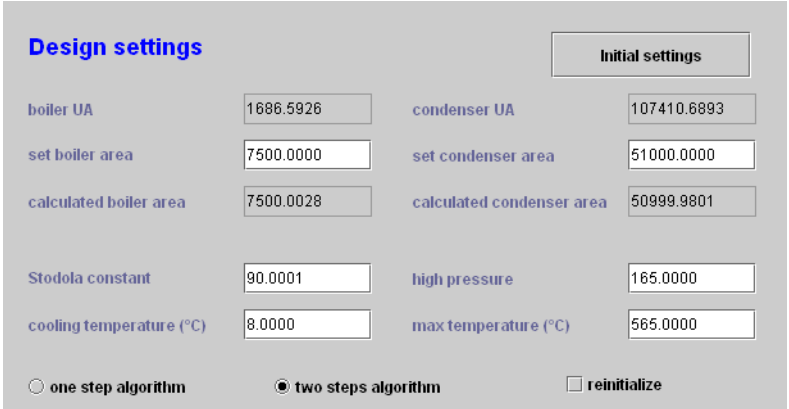

Figure 2 : Ecran du pilote (saisie des paramètres)

Commencez par cliquer sur "Initial settings" pour instancier les TechnoDesign et effectuer un premier dimensionnement technologique, en l'occurrence calculer la constante de Stodola de la turbine et 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.

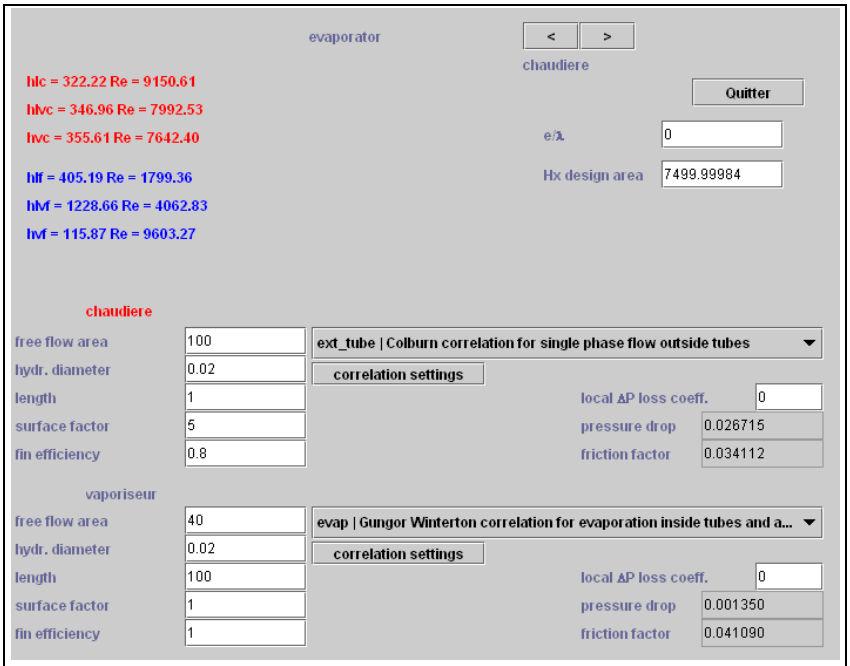

Figure 3 : Ecran de dimensionnement de la chaudière

3.3.1 Chaudière

On considère que la chaudière a côté vapeur une section de passage du fluide de 40 m^2 et un diamètre hydraulique de 2 cm avec une longueur de tubes de 100 m, et côté fumées une section de passage de 100 m^2 et un diamètre hydraulique de 2 cm, pour une longueur de tubes ailetés de 5 m, avec un facteur de surface égal à 5 et une efficacité d'ailettes de 0,8. Choisissez le type de configuration ("evap Gungor Winterton" pour évaporation à l'intérieur des tubes pour la partie "vaporiseur" (la vapeur), et "ext tube Colburn correlation" pour les fumées), et paramétrez l'écran comme indiqué figure 3.

3.3.2 Condenseur

Le condenseur a, côté vapeur, une section de passage du fluide de 123,6 m^2 et un diamètre hydraulique de 2 cm, sur une longueur de 10 cm, et côté refroidisseur une section de passage de 25,5 m^2 et un diamètre hydraulique de 1,6 cm, pour une longueur de tubes de 1 m.

Choisissez le type de configuration ("cond_ext_hor Levy correlation" pour la condensation à l'intérieur des tubes pour la partie "condenseur" (la vapeur), et "int_tube Mac Adams correlation" pour l'eau de refroidissement), et paramétrez l'écran comme indiqué figure 4.

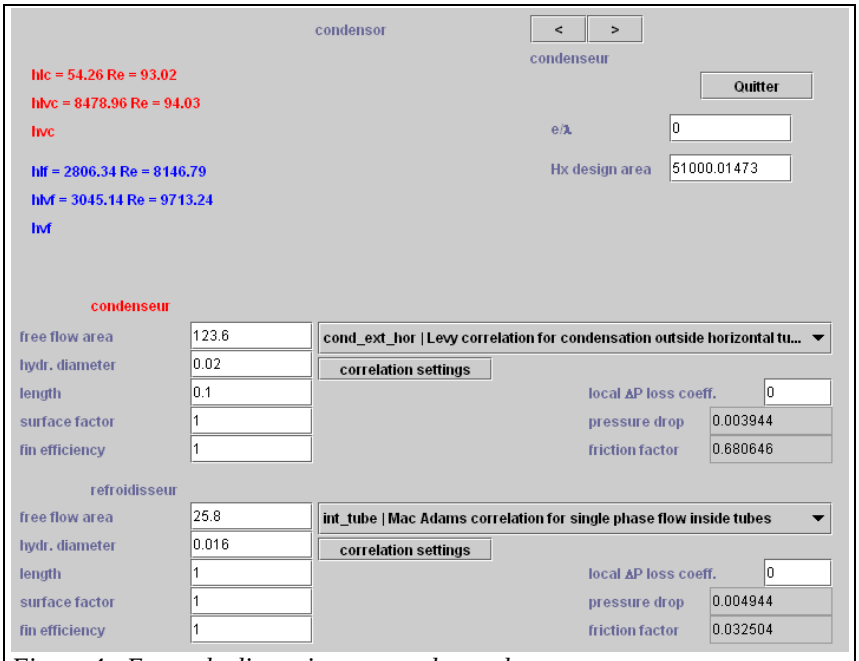

Figure 4 : Ecran de dimensionnement du condenseur

3.3.3 Turbine

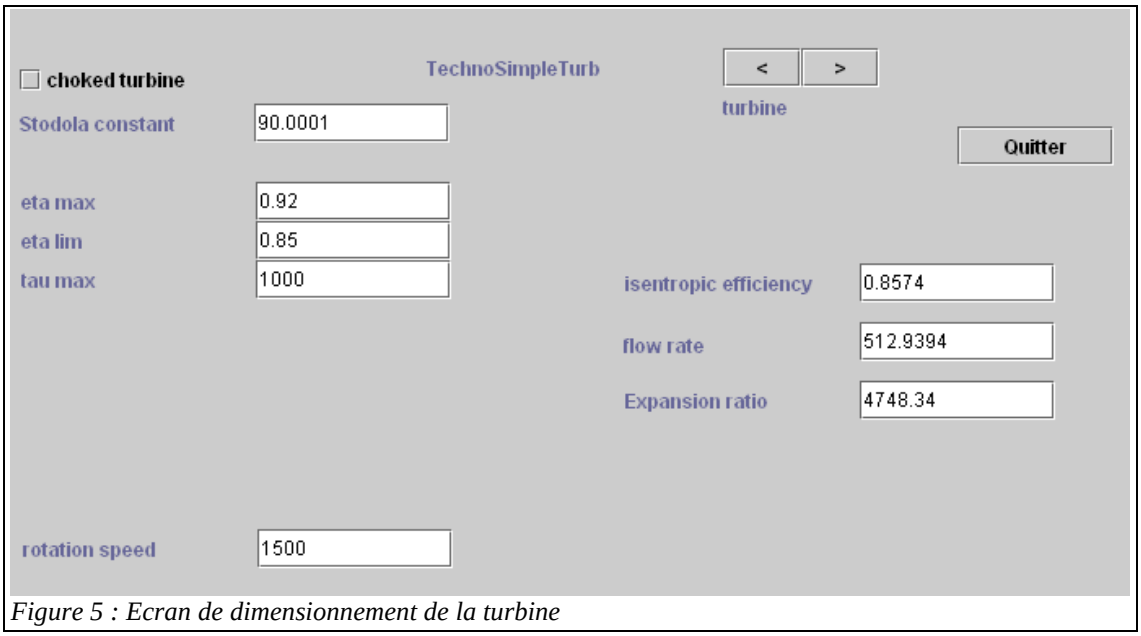

Figure 5 : Ecran de dimensionnement de la turbine

Pour la turbine (figure 5), il faut fournir les valeurs des paramètres qui interviennent dans les équations du rendement isentropique. La vitesse de rotation apparaît, mais elle n'est pas prise en compte dans ce premier modèle.

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

Les résultats sont affichés dans les écrans technologiques : surfaces d'échanges de 7 500 et 51 000 m^2 pour la chaudière et le condenseur ; constante de Stodola de 90 pour la turbine.

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 et la constante de Stodola de la turbine, 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 la chaudière 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.

```
inRiver.getProperties();
outRiver.getProperties();
Teau=inRiver.T;
Teau_value.setText(Util.aff_d(Teau-273.15,4));
```

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

```
mCpCalopCond=DeltaH/(outRiver.T-inRiver.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());
```

L'initialisation de la turbine permet de connaître le débit nominal et de déterminer la constante de Stodola correspondante, qui est affichée dans l'écran du pilote.

```
if(!turbineName.equals("")){//initialisation de la turbine H2O
args=new String[2];
args[0]="process";
args[1]=turbineName;
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();
f=(Double)vProp.elementAt(11);
eta_is=f.doubleValue();

Kh=technoTurbine.getStodolaConstant();
Kh_value.setText(Util.aff_d(Kh,4));
PVR=0;
}
```

5 Résolution du système d'équations de la machine de centrale à vapeur 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 couple (débit de vapeur, température de condensation), les variables liées étant les pressions du fluide et les puissances thermiques et mécaniques mises en jeu. A ces deux variables "naturelles" , il faut ajouter les deux variables intermédiaires UAevap et UAcond (cf. section 2.1).

La recherche de ce quadruplet 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 la valeur de la température de condensation en fonction de celle du fluide de refroidissement, pour éviter tout croisement de température dans le condenseur.

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 :

```
x[1] = Tcond;
x[2] = massFlow;
```

[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[3] = UAevap;
x[4] = UAcond;
```

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 la chaudière et du condenseur et la constante de Stodola pour la turbine, ainsi que la température du fluide de refroidissement. Les autres valeurs sont celles des écrans du simulateur.

```
 //lecture à l'écran du pilote des paramètres et variables, éventuellement modifiés après initialisation
 AdesignChaudiere=Util.lit_d(AdesignChaudiere_value.getText());
 AdesignCond=Util.lit_d(AdesignCond_value.getText());
 Kh=Util.lit_d(Kh_value.getText());
 technoTurbine.setKh(Kh);
 UAevap=Util.lit_d(UAevap_value.getText());
 UAcond=Util.lit_d(UAcond_value.getText());

 Pevap=Util.lit_d(Pevap_value.getText());
 avalChaudiere.P=Pevap;
 avalChaudiere.T=Util.lit_d(Tmax_value.getText())+273.15;
 avalChaudiere.update(UPDATE_T,UPDATE_P,!UPDATE_X);
 avalChaudiere.getProperties();

 amontChaudiere.P=Pevap;
 amontChaudiere.update(!UPDATE_T,UPDATE_P,!UPDATE_X);
 amontChaudiere.getProperties();
 Teau=Util.lit_d(Teau_value.getText())+273.15;

 inRiver.T=Teau;
 inRiver.update(UPDATE_T,!UPDATE_P,!UPDATE_X);
 inRiver.getProperties();

 amontCond.getProperties();
 avalCond.getProperties();
 DTsurch=avalChaudiere.DTsat;
 DTssrefr=avalCond.DTsat;

algorithmResults.setText("");
```

Comme indiqué plus haut, selon que l'on coche ou non l'option "reinitialize", les températures de changement d'état sont celles du simulateur ou déterminées à partir des températures du fluide de refroidissement. 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= avalCond.T+Teau-inRiver.T;
}
else{
 Tcond=fluideThermo.getSatTemperature(avalCond.P, 1);
}
```

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

```
int m = 4;
int n = 4;

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] = massFlow;
x[3] = UAevap;
x[4] = UAcond;

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 quatre méthodes recCond(), resFlow(), resAevap(), resAcond()définies ci-dessous.

L'appel aux deux 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];
 massFlow=x[2];
 UAevap=x[3];
 UAcond=x[4];

 Vector vProp;
 Double f;
```

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.

```
//mise à jour du point aval du condenseur (avec modification du sous-refroidissement)
Pcond=fluideThermo.getSatPressure(Tcond, 1);
avalCond.T=Tcond+DTssrefr;// DTssrefr est négatif
avalCond.P=Pcond;
avalCond.T=fluideThermo.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();

double Hl=avalCond.H;
DHcompr=avalCond.V*(avalChaudiere.P-avalCond.P)*100;//prise en compte de la compression liquide
Hl=Hl+DHcompr;
DHcompr=DHcompr*massFlow;

amontCond.P=Pcond;
amontCond.update(!UPDATE_T,UPDATE_P,!UPDATE_X);
amontCond.getProperties();

//nouveau DeltaHevap
DeltaHevap=massFlow*(avalChaudiere.H-Hl);
mCpRefrigChaudiere=DeltaHevap/(avalChaudiere.T-avalCond.T);
double[]res=Util.epsi_NUT(mCpRefrigChaudiere,mCpCalopChaudiere,UAevap);
epsilon=res[0];
mCpmin=res[1];
Tf=amontChaudiere.T+DeltaHevap/epsilon/mCpmin;

hotGases.T=Tf;
hotGases.update(UPDATE_T,!UPDATE_P,!UPDATE_X);
coldGases.T=Tf-DeltaHevap/gasFlow;
coldGases.update(UPDATE_T,!UPDATE_P,!UPDATE_X);

hotGases.getProperties();
coldGases.getProperties();

//calcul de la turbine
DeltaHcond=DeltaHevap+getTurbinePower();
amontCond.getProperties();
```

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

```
 //calcul des premiers résidus
fvec[1] = resCond();
fvec[2] = 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(boilerName, RECALCULATE, !UPDATE_UA, 0, !UPDATE_EPSI, 0, !UPDATE_DTMIN, 0);
 updateHx(condenserName, RECALCULATE, !UPDATE_UA, 0, !UPDATE_EPSI, 0, !UPDATE_DTMIN, 0);
 updateHx(boilerName, RECALCULATE, !UPDATE_UA, 0, !UPDATE_EPSI, 0, !UPDATE_DTMIN, 0);
 updateHx(condenserName, RECALCULATE, !UPDATE_UA, 0, !UPDATE_EPSI, 0, !UPDATE_DTMIN, 0);
 updateHx(boilerName, 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 :

```
amontChaudiere.getProperties();
avalChaudiere.getProperties();
amontCond.getProperties();
avalCond.getProperties();
hotGases.getProperties();
coldGases.getProperties();
inRiver.getProperties();
outRiver.getProperties();
```

Les TechnoDesign des échangeurs sont alors recalculés, ce qui met à jour les pertes de charge (non prises en compte dans ce modèle) et permet d'estimer les surfaces d'échange correspondant à ce nouvel état :

```
technoChaudiere.makeDesign();
AcalculatedChaudiere_value.setText(technoChaudiere.ADesign_value.getText());

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

Enfin, les deux derniers résidus sont estimés :

```
//calcul des derniers résidus après recalcul des TechnoDesign (adimensionnés)
fvec[3] = resAevap();
fvec[4] = resAcond();

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

double Tsortie_refrig=0;
mCpRefrigCond=DeltaHcond/(amontCond.T-(Tcond+DTssrefr));
double[] res=Util.epsi_NUT(mCpRefrigCond,mCpCalopCond,UAcond);
epsilon=res[0];
mCpmin=res[1];
double Tentree_refrig=Teau+DeltaHcond/epsilon/mCpmin;
Tsortie_refrig=Tentree_refrig-DeltaHcond/mCpRefrigCond;

//le résidu est l'écart entre les deux températures de condensation
double z= Tcond-(Tsortie_refrig-DTssrefr);
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));

eff_value.setText(Util.aff_d(-tauTurb/DeltaHevap,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));
tauTurb_value.setText(Util.aff_d(tauTurb,4));
DeltaHcompr_value.setText(Util.aff_d(DHcompr,4));
```

6 Equations du cycle à vapeur en non-nominal

6.1 Bilan de la chaudière

Les températures du caloporteur sont fixées par l'équilibre thermique de la chaudière, lequel dépend essentiellement d'une part de la température d'évaporation et du débit du caloporteur (fumées dans cet exemple), et d'autre part du débit de vapeur.

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

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

```
//nouveau DeltaHevap
DeltaHevap=massFlow*(avalChaudiere.H-Hl);
mCpRefrigChaudiere=DeltaHevap/(avalChaudiere.T-avalCond.T);
double[]res=Util.epsi_NUT(mCpRefrigChaudiere,mCpCalopChaudiere,UAevap);
epsilon=res[0];
mCpmin=res[1];
Tf=amontChaudiere.T+DeltaHevap/epsilon/mCpmin;
```

6.2 Bilan de la turbine

Le bilan de la turbine exprime que le travail de détente est égal au produit du débit par le travail massique de détente isentropique et par le rendement isentropique. Il dépend lui aussi de plusieurs grandeurs et est donné par :

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

Il est calculé par :

```
double getTurbinePower(){
Vector vProp;
Double f;
```

```
//recalcul de la turbine
double eta_is=technoTurbine.getRisentr();
updateprocess(turbineName, "Expansion",RECALCULATE,IS_SET_FLOW, UPDATE_FLOW, massFlow,
UPDATE_ETA, eta_is);

//ce qui permet de connaître la puissance consommée
String[] args=new String[2];
args[0]="process";
args[1]=turbineName;
vProp=proj.getProperties(args);
f=(Double)vProp.elementAt(4);
tauTurb=f.doubleValue();
double tauIs=tauTurb/eta_is;
return tauTurb;
}
```

6.3 Premier principe

τ_c étant le travail de compression réalisé par la pompe, le premier principe s'écrit :

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

Il est écrit dans fcn() sous la forme :

```
DeltaHcond=DeltaHevap+getTurbinePower();
```

6.4 Bilan du condenseur

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

La température de sortie condenseur dépend ainsi du rapport de détente et du rendement isentropique de la turbine, 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 de la turbine) avec celle de massFlow = x[2].

```
double resFlow(){
//renvoie le résidu du débit-masse
//recalcul du débit d'eau
double y=technoTurbine.getMassFlow();
double z= massFlow-y;
z= 2*z/(massFlow+y);
return z;
}
```

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 d'autre part la constante de Stodola, la température et la pression maximales du cycle et la température du fluide de refroidissement.

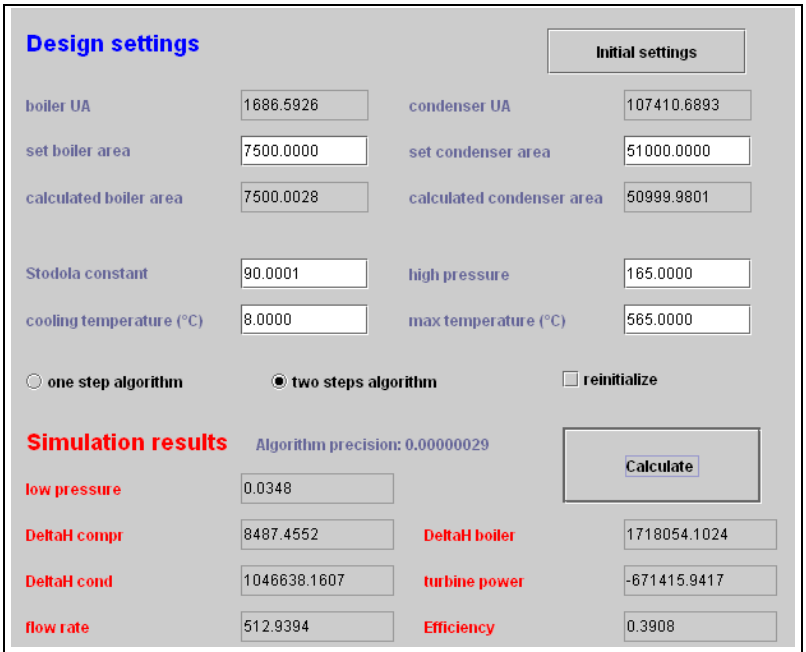

Figure 6 : Ecran du pilote

Entrez les valeurs 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.

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 refroidissement, pour le paramétrage des écrans technologiques retenu précédemment. L'influence de la pression maximale du cycle 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 autres, 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.

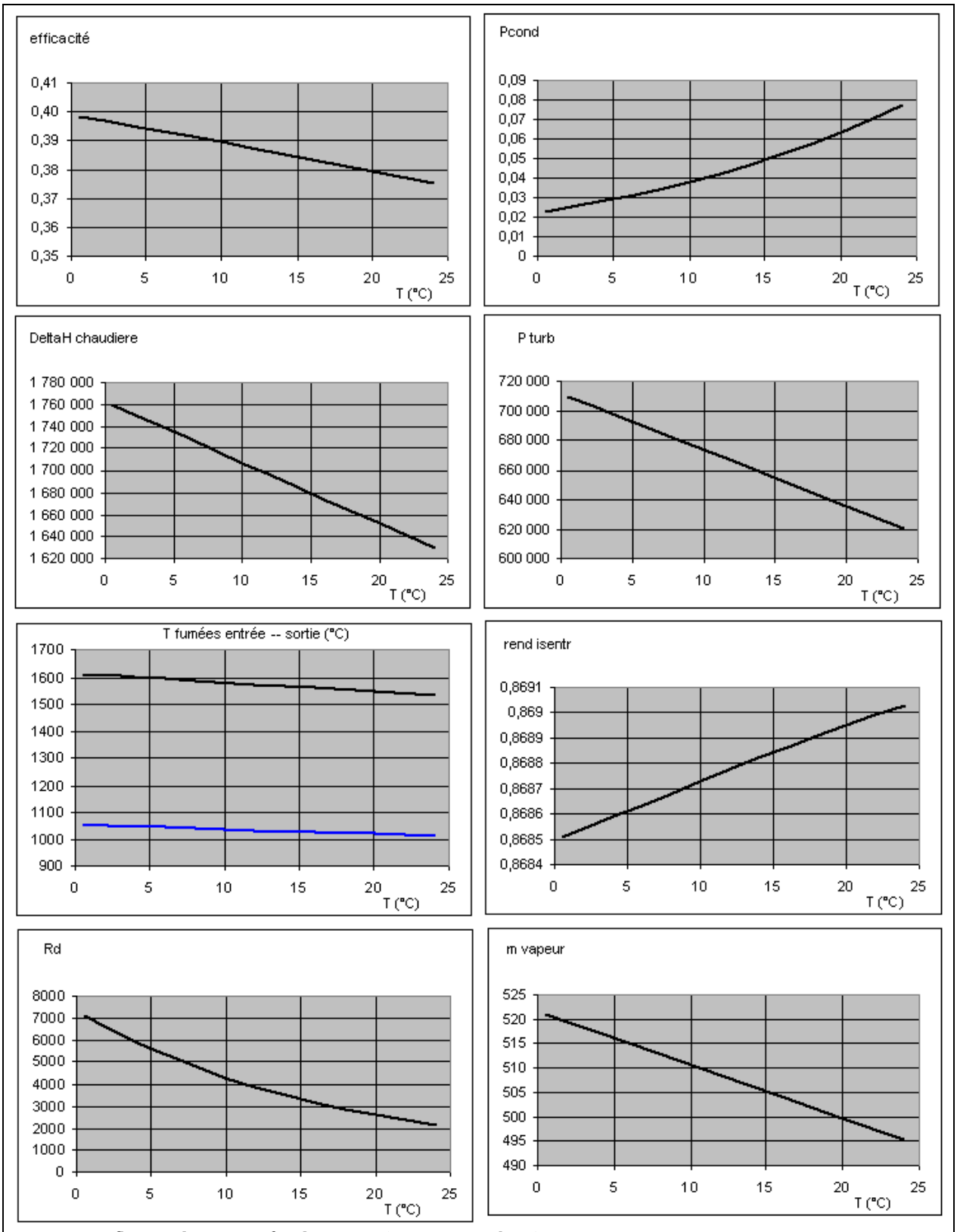

Figure 7 : Influence de T eau refroidissement (Pevap = 165 bars)

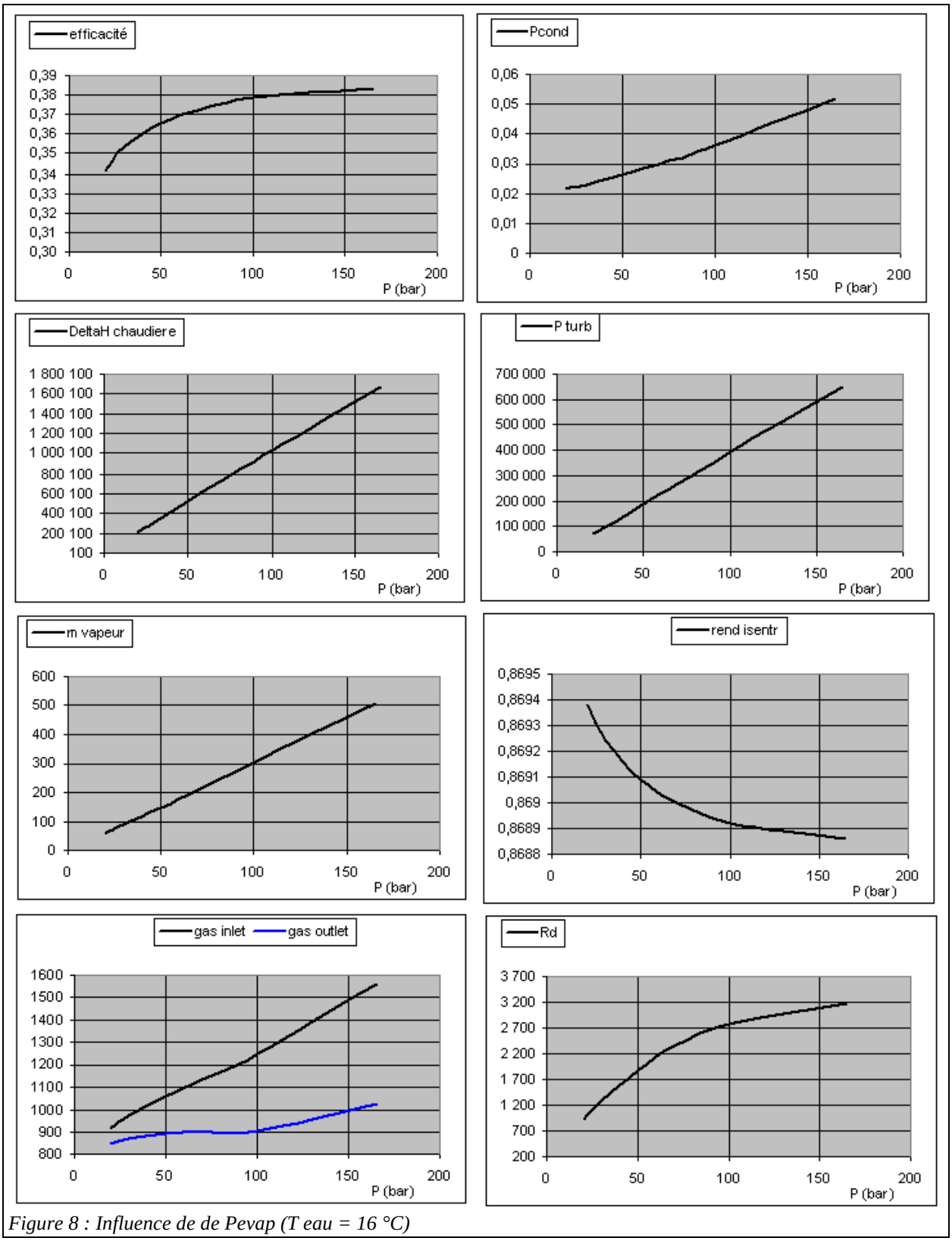

Figure 8 : Influence de de Pevap (T eau = 16 °C)

8 Cycle avec prise en compte des pertes par vitesse restante

Lorsque la pression aval baisse, pour une section de passage donnée, la vitesse débitante augmente, au point que l'énergie cinétique en sortie de turbine peut ne plus être négligeable. Physiquement, cela signifie qu'une partie de la puissance disponible n'est pas récupérée sur l'arbre de la turbine, et se transforme à l'aval en chaleur.

L'évaluation des pertes correspondantes peut être effectuée sur la base du triangle des vitesses, ce qui suppose de connaître la section de passage aval, la vitesse circonférentielle U et l'angle β de sortie. Dans l'équation ci-dessous, C_2 représente la vitesse débitante Cf avec nos notations habituelles.

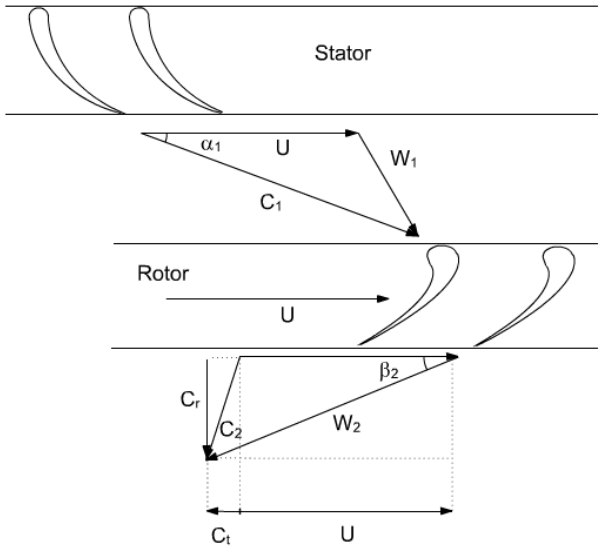

Figure 9 : Triangle des vitesses d'une turbine

Ces pertes sont égales à $C_2^2/2$, avec (figure 9) :

$C_t = Cr \cot g\ \beta - U$

$C_2^2 = Cr^2 + Ct^2 = Cr^2 + (Cr \cot g\ \beta - U)^2$

Sur le plan pratique, le calcul des pertes par vitesse restantes (PVR) peut être effectué de la manière suivante.

- on commence par déterminer l'état de sortie de la turbine en l'absence de PVR ;
- la connaissance du volume massique et de la sectrion d'échappement permet de connaître Cr et donc les PVR ;
- le travail fourni par la turbine est égal à celui en l'absence des PVR moins les PVR. Connaissant le travail isentropique en l'absence de PVR, on en déduit une nouvelle valeur du rendement isentropique ;
- un recalcul de la turbine avec cette valeur du rendement isentropique donne l'état du point de sortie en tenant compte des PVR.

8.1 Modification du code

La classe technologique de la turbine s'appelle maintenant TechnoTurb, et le code de getTurbinePower() est simplement modifié comme suit :

```
double tauIs=tauTurb/eta_is;
PVR=0;

if(technoTurbine.isPVR()){//s'il y a prise en compte des pertes par vitesse restante
amontCond.getProperties();
double v=amontCond.V;
double volFlow=massFlow*v;
PVR=technoTurbine.getPVR(volFlow);
tauTurb=tauTurb+PVR*massFlow;
eta_is=tauTurb/tauIs;
if(eta_is<0.2)eta_is=0.2;//pour éviter des valeurs négatives éventuelles
technoTurbine.risentr_value.setText(Util.aff_d(eta_is,4));
updateprocess(turbineName, "Expansion",RECALCULATE,IS_SET_FLOW, UPDATE_FLOW, massFlow,
UPDATE_ETA, eta_is);
}
```

return tauTurb;

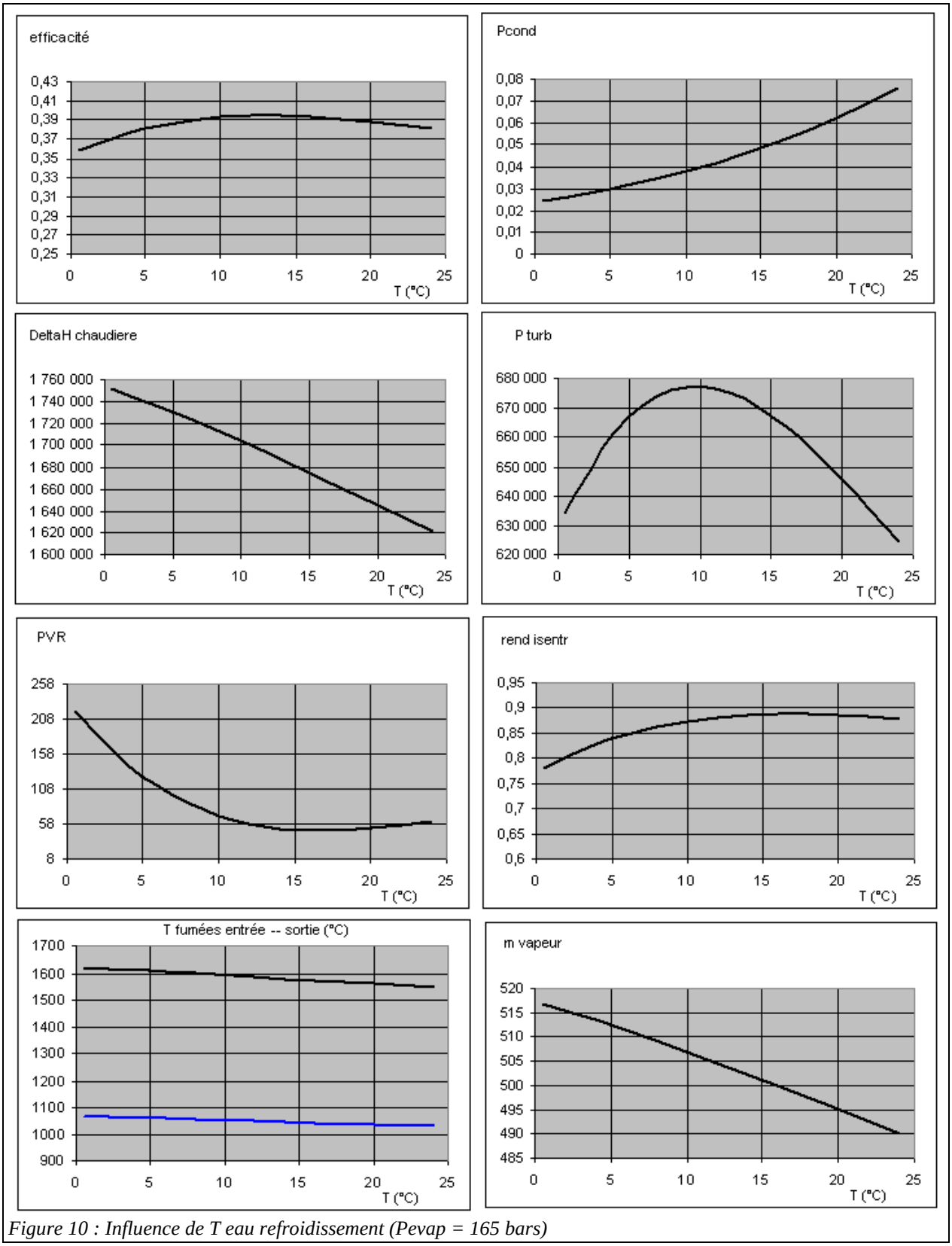

Figure 10 : Influence de T eau refroidissement (Pevap = 165 bars)

8.1 Nouveaux résultats

Avec ces hypothèses, l'influence de la température de refroidissement est assez différente du cas précédent, essentiellement parce que les pertes par vitesse restante viennent pénaliser le cycle lorsque la température de refroidissement est basse (figure 10). C'est moins sensible sur l'influence de la pression (figure 11).

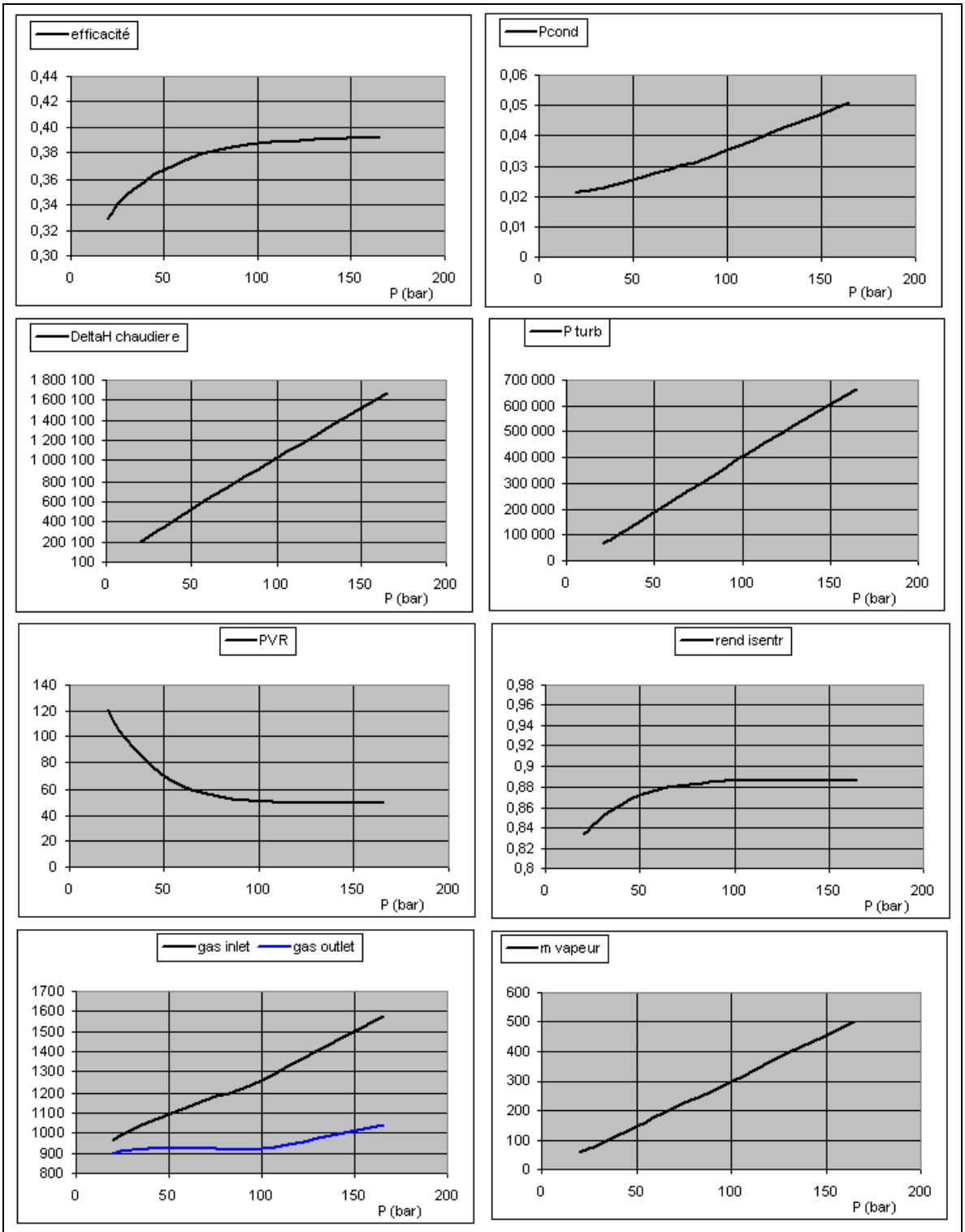

Figure 11 : Résultats du modèle : influence de Pevap (T eau = 16 °C)

9 Modélisation avec cartographie de turbine

9.1 Cartographie des turbines

Dans le cas d'une turbine, bien qu'il soit aussi possible de retenir le même repère que pour les turbocompresseurs, c'est généralement le rapport de pression qui est utilisé en abscisse. En ordonnée, on trouve le débit massique corrigé ou l'efficacité isentropique de la machine.

Le paramètre des courbes est encore la vitesse de rotation corrigée, mais elle ne joue ici qu'un rôle secondaire : ce n'est que lorsqu'on veut très fortement réduire le rapport de pression ou la vitesse de rotation que les performances tendent à se dégrader. Cette souplesse tient à la stabilité d'écoulement dans les aubages liée au gradient de pression qui y règne.

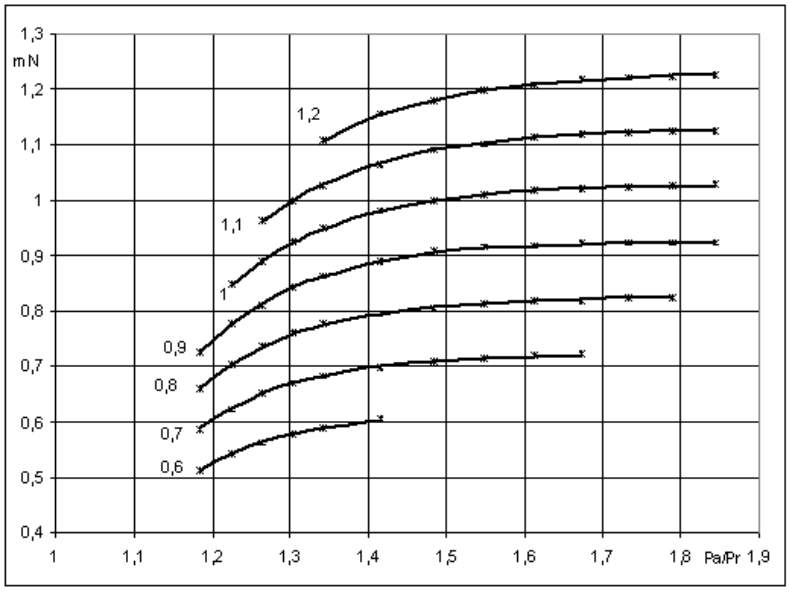

Figure 12 : Caractéristiques (rapport de détente, débit)

Compte tenu du regroupement des courbes, les caractéristiques (rapport de détente, débit) sont peu lisibles, et il est intéressant de changer d'ordonnée pour avoir une meilleure visibilité. On peut en particulier multiplier le débit par la vitesse de rotation réduite, ce qui conduit à la représentation de la figure 12.

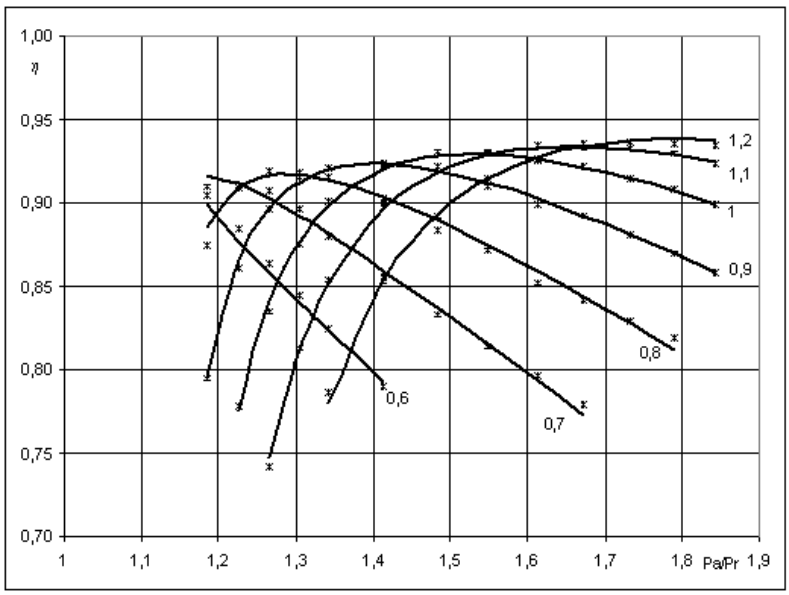

Figure 13 : Caractéristiques (rendement isentropique, débit)

Ces équations sont mises sous forme adimensionnelle et exprimées numériquement comme expliqué dans la note ModelisationSimplifieeTurbomachines.doc[4] et dans le tome 4 du manuel de référence de Thermoptim

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

9.2 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 MultiStageMappedTurbine (TechnoDesign) et TurbineMapDataFrame (cartographie). L'instanciation du TechnoDesign est changée en conséquence.

technoTurbine= new MultiStageMappedTurbine (proj, turbineName, avalChaudiere, amontCond);

[4] http://www.thermoptim.org/SE/seances/C01/ModelisationSimplifieeTurbomachines.pdf

9.2.1 Initialisations de dimensionnement de la turbine

Lors de l'initialisation de la turbine, on charge la cartographie appropriée, puis on recherche, compte tenu des conditions d'entrée, soit la vitesse de rotation conduisant au rapport de détente nominal, soit le nombre d'étages compatible avec la vitesse de rotation choisie :

```
technoTurbine.setDataFile(technoTurbine.getDataFile());

if(JCheckRotationSpeed.isSelected()){
//recherche de la vitesse de rotation correspondant à Rp et massFlow
double Ninit=technoTurbine.getNfromRpAndFlow(Rd,
massFlow)/technoTurbine.racT0*Math.pow(avalChaudiere.T,0.5)*technoTurbine.Nref;
if(Ninit!=0){//si on est en dehors de la cartographie, on ne fait rien en dehors du message d'information
technoTurbine.setN(Ninit/technoTurbine.Nref);
technoTurbine.setNparameters();//calcule les paramètres qui dépendent de N
N_value=Ninit;
Nref_value.setText(Util.aff_d(Ninit,4));
}
}
else{
N_value=Util.lit_d(Nref_value.getText())/technoTurbine.Nref;
technoTurbine.setN(N_value);
int nStage=technoTurbine.getStageNumber(N_value,massFlow);
//prévoir test sur valeur de j
nStage_value.setText(Util.aff_i(nStage));
technoTurbine.setStageNumber(nStage);
}
double v=amontCond.V;
double volFlow=massFlow*v;
technoTurbine.getPVR(volFlow);
```

9.2.2 Initialisations des calculs en non-nominal

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

```
Kh=Util.lit_d(nStage_value.getText());
technoTurbine.setKh(Kh);
N_value=Util.lit_d(Nref_value.getText())/technoTurbine.Nref;
technoTurbine.setDesignValues();
technoTurbine.setN(N_value);
technoTurbine.setNparameters();
technoTurbine.setStageNumber();
```

9.2.3 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 de la turbine
technoTurbine.updateN();
technoTurbine.setNparameters();
```

9.2.4 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.

```
technoTurbine.checkMapValidity();
```

9.3 Résultats du modèle

Les résultats du modèle sont tout à fait cohérents avec ceux obtenus précédemment pour ce qui concerne l'influence de la température de refroidissement, les seules différences étant dues au changement de caractéristiques (cf. figure 14).

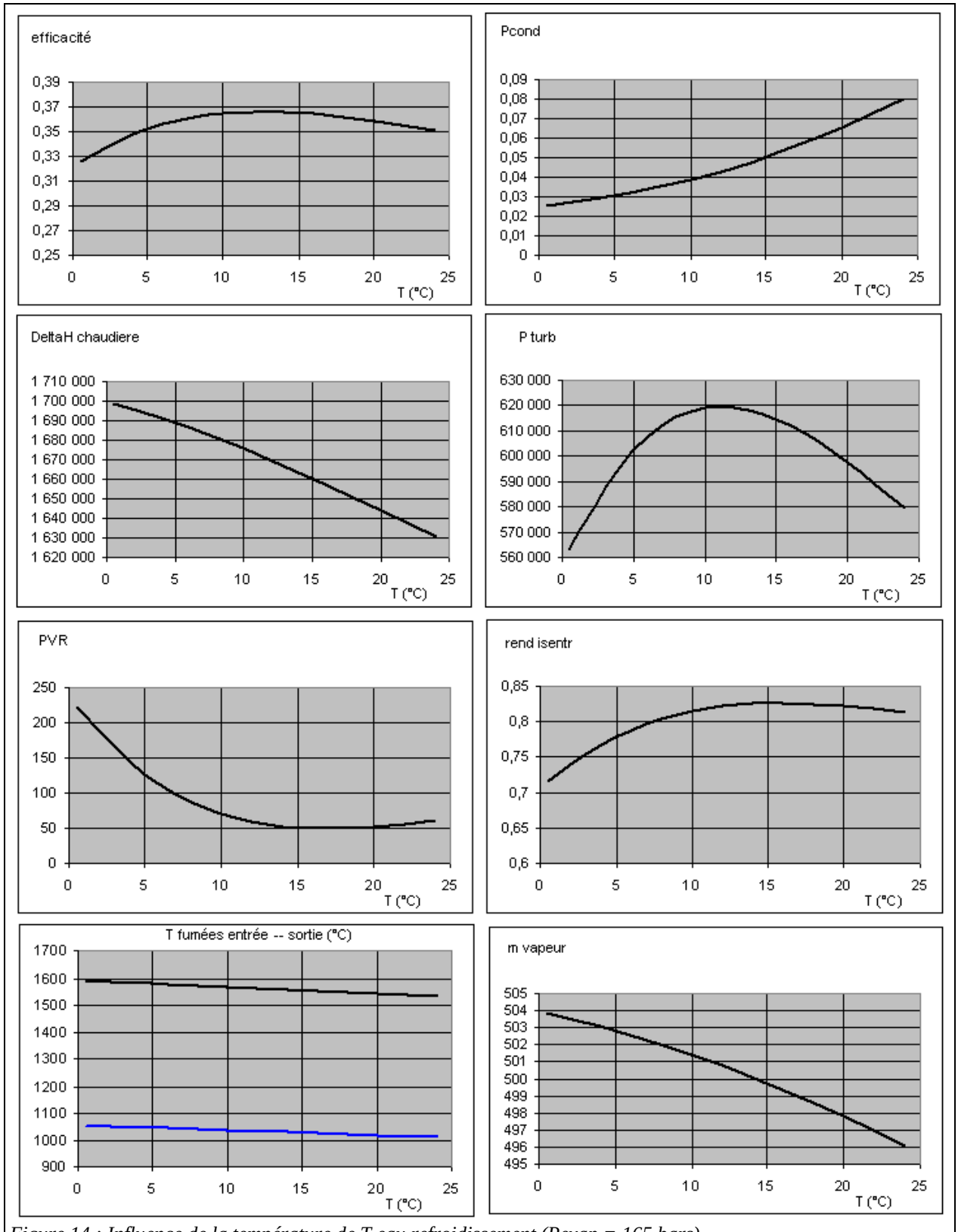

Figure 14 : Influence de la température de T eau refroidissement (Pevap = 165 bars)

Celle de la pression maximale du cycle est donnée figure 15.

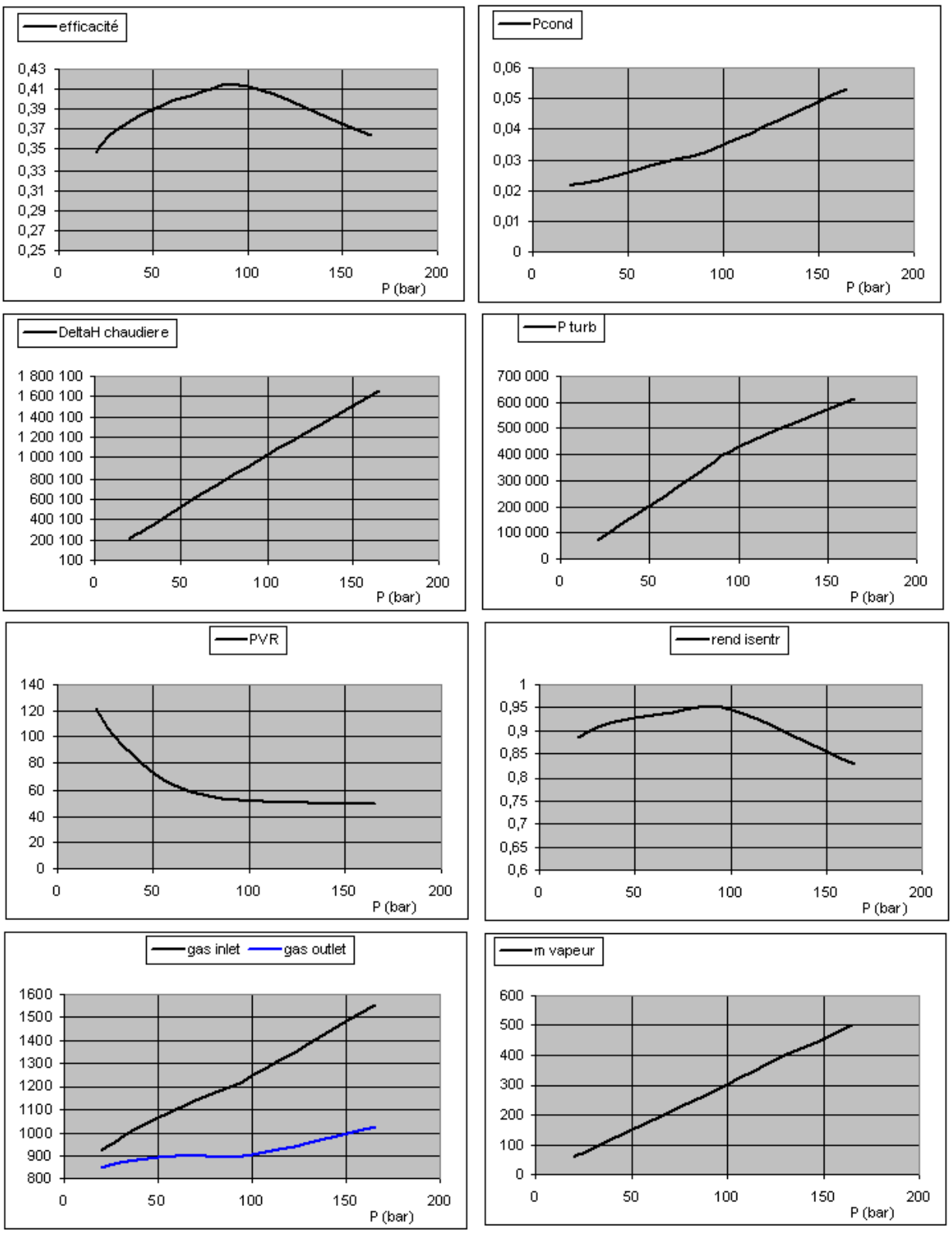

Figure 15 : Influence de la vitesse de de Pevap (T eau = 16 °C)

Annexe 1 : Principe de calcul des échangeurs multizones

Evaporateurs : fluide froid diphasique et fluide chaud en sensible

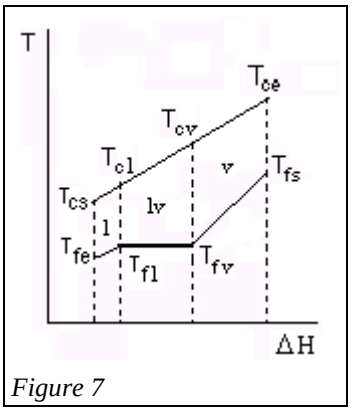

Figure 7

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

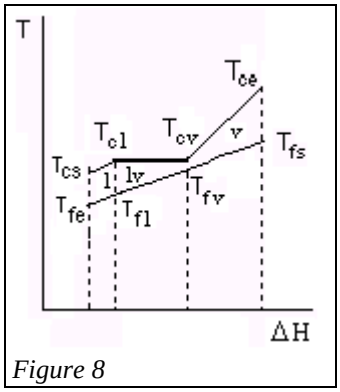

Figure 8

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

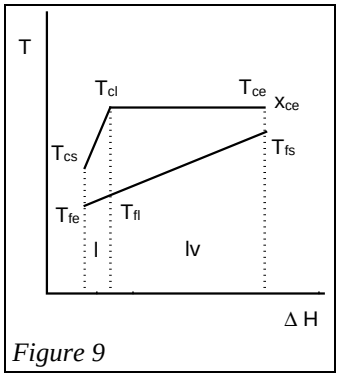

Figure 9

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 :

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

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