T H E R M O P T I M ®

MANUEL

DE

RÉFÉRENCE

TOME IV

PILOTES EXTERNES

DIMENSIONNEMENT TECHNOLOGIQUE

NON NOMINAL

VERSION JAVA 1.7

SOMMAIRE

1 INTRODUCTION

Dans ce tome 4, nous présentons une série de développements qui ont été réalisés pour permettre de compléter le noyau de Thermoptim afin de pouvoir aborder des modélisations beaucoup plus complexes que celles qui ont été présentées dans le reste du manuel de référence, et qui permettent de :

- piloter les calculs effectués par Thermoptim en prenant la main sur le moteur de recalcul automatique
- réaliser des dimensionnements technologiques et effectuer des études de comportement d'un système en régime non-nominal.

Compte tenu de l'étendue du domaine correspondant à ces problématiques, nous ne pouvons en aucun cas prétendre leur apporter des réponses exhaustives : nous limitons notre ambition à présenter, autour de quelques exemples relativement simples, le potentiel offert par les pilotes et les outils de dimensionnement technologique et de non-nominal.

Nous ne pouvons, à ce stade de nos travaux, que montrer la voie, laissant aux utilisateurs confirmés qui auront à résoudre des problèmes précis, le soin de bâtir par eux-mêmes les modèles pertinents. Soulignons qu'il s'agit d'un sujet extrêmement intéressant quoique particulièrement difficile, mal connu et peu traité jusqu'ici, sans doute faute de disposer d'un outil approprié, mais essentiel si l'on veut vraiment comprendre comment se comportent les machines réelles. Ce document étant un manuel de référence du progiciel, nous n'avons pas pu y développer beaucoup les analyses thermodynamiques. Nous renvoyons au tome 3 du livre Systèmes Energétiques le lecteur intéressé par plus développements à ce sujet.

Jusqu'en 2008, la finesse des modèles de composants utilisés dans Thermoptim était limitée : les modèles phénoménologiques construits permettaient certes d'étudier le cycle thermodynamique de la technologie étudiée, mais pas d'en effectuer un dimensionnement technologique précis, ni d'en simuler les performances en régime de fonctionnement non-nominal, ces deux dernières problématiques étant beaucoup plus complexes que la première. Pour pouvoir traiter ces questions, nous avons été conduits à **distinguer deux niveaux dans les modèles** (leur dénomination a été choisie pour être la plus claire possible, mais elle n'est pas usuelle) :

- les **modèles phénoménologiques**, les seuls jusqu'ici mis en œuvre dans Thermoptim, donnent accès au calcul des cycles thermodynamiques, indépendamment du choix d'une technologie de composants particulière ;
- les **modèles de dimensionnement technologique et de simulation en régime non-nominal** fournissent non seulement les mêmes résultats que les précédents, mais, grâce à la prise en compte d'équations complémentaires spécifiques aux technologies choisies, permettent en plus de dimensionner précisément les divers composants et, une fois le dimensionnement technologique réalisé, d'étudier le comportement du système considéré en dehors des conditions de fonctionnement pour lesquelles il a été dimensionné.

De tels modèles sont par exemple nécessaires lorsque l'on veut évaluer les performances d'une installation existante, qui fonctionne souvent dans des conditions différentes de celles pour lesquelles elle a été dimensionnée. Cette problématique d'audit (en vue en particulier de proposer des améliorations) intéresse un nombre croissant d'organismes, industriels ou autres.

L'analyse des systèmes énergétiques en régime non-nominal pose de nombreux problèmes beaucoup plus difficiles à résoudre que ceux que l'on rencontre lorsque l'on se contente d'étudier les cycles thermodynamiques sous l'angle purement phénoménologique au point nominal, indépendamment des choix technologiques.

Précisons bien que ce que nous appelons régime non-nominal correspond au fonctionnement stabilisé d'une installation pour des conditions opératoires différentes de celles pour lesquelles elle a été dimensionnée : il ne s'agit pas d'étudier le régime transitoire rapide résultant par exemple des actuateurs d'une régulation.

Le dimensionnement technologique des composants nécessite d'affiner les modèles phénoménologiques utilisés dans le noyau de Thermoptim, en les complétant de manière appropriée.

Comme nous le verrons dans ce tome, les implémentations de ces nouvelles fonctionnalités de Thermoptim sont pour l'essentiel effectuées sous forme de classes externes, ce qui offre une grande latitude aux utilisateurs pour les personnaliser (cf. tome 3 du manuel de référence).

La version classique phénoménologique de Thermoptim reste inchangée : elle est seulement complétée par une couche logicielle permettant de prendre en compte les modèles de dimensionnement technologique et de non-nominal.

Les classes de dimensionnement technologique complètent celles du noyau en permettant la prise en compte des équations propres aux choix technologiques effectués, comme par exemple le calcul du coefficient d'échange thermique U d'un échangeur de chaleur, ou bien le débit et le rendement isentropique d'un compresseur.
Le pilote assure quant à lui la coordination systémique des calculs relatifs aux différents composants qui entrent en jeu. Il recherche un ensemble de variables de couplage cohérent avec le problème physique et technologique posé, et permet ainsi de modifier le paramétrage du projet Thermoptim.

Précisons que toutes les options de paramétrage offertes dans les écrans habituels (phénoménologiques) de Thermoptim ne sont pas nécessairement valides en régime non-nominal, et qu'il importe donc de prévoir, dans les classes externes que l'on développe, des tests destinés à vérifier leur conformité aux hypothèses des modèles retenus.

2 PILOTAGE DE THERMOPTIM

2.1 Présentation générale

Piloter Thermoptim a deux principales applications :

- d'une part faciliter la mise au point des classes externes en les testant au fur et à mesure qu'elles sont définies
- d'autre part donner accès à l'ensemble des bibliothèques non protégées pour différents usages (coordonner les calculs de différents composants couplés entre eux, effectuer des traitements externes entre deux recalculs, guider un utilisateur dans un module de formation…).

Pour rendre possible le pilotage externe, la classe principale du simulateur, Projet, peut être sous-classée, et un certain nombre de méthodes non protégées peuvent être surchargées pour définir des traitements complémentaires de ceux du noyau. Le détail de ces méthodes est donné en annexe du tome 3.

Il y a deux moyens de piloter Thermoptim : soit totalement à partir d'une application externe qui instancie le progiciel et le paramètre, soit partiellement, pour un projet donné. Un exemple relevant du premier cas est donné dans le tome 3, où il sert à créer les classes externes et les tester depuis l'environnement de développement des classes externes présenté à la section 6, qui explique comment utiliser un logiciel en freeware pour disposer d'un environnement de travail convivial. Nous n'en parlerons pas particulièrement dans ce tome.

Dans le second cas, auquel nous nous intéresserons ici, on associe une classe de pilotage spécifique à un projet donné, et cette classe est instanciée lors du chargement du projet. Elle permet alors de coordonner les recalculs de ce projet selon des règles particulières. La classe de pilotage est une extension de extThopt.ExtPilot, qui dérive de rg.thopt.PilotFrame. Pour pouvoir l'associer au projet, il faut commencer par en faire une classe externe pour qu'elle soit chargée dans Thermoptim lors de son lancement. Une fois le projet ouvert, il faut sélectionner la classe de pilotage qui lui est associée par la ligne "Pilot frame" du menu Special (cf. section 2.2.3). Si le projet est sauvé, le nom de la classe de pilotage est sauvegardé pour qu'elle puisse être instanciée lors d'un chargement ultérieur. Il est possible de sauvegarder les données du pilote au même titre que celles d'un composant externe, ce qui permet d'enregistrer des résultats de simulation qui ne peuvent l'être dans les éléments habituels de Thermoptim.

Comme nous le verrons à l'occasion des exemples qui sont donnés dans ce tome, cette fonctionnalité de Thermoptim est extrêmement puissante, car elle permet de personnaliser l'utilisation de la plupart des éléments du progiciel, ceci au prix d'un travail de programmation tout à fait raisonnable.

Dans cette section, nous présentons un pilote qui sert à calculer le bilan énergétique d'un moteur Stirling solaire. Il s'agit d'un exemple très simple qui permet d'introduire ce type de classe externe sans qu'il soit nécessaire de se plonger dans le détail d'un modèle thermodynamique complexe.

Un moteur Stirling est un moteur d'un type particulier, qui travaille en système fermé, et met en œuvre une compression refroidie et une détente réchauffée, de telle sorte que les indicateurs de performance usuels de

Thermoptim ne peuvent pas être directement utilisés : l'énergie payante est la somme de la chaleur fournie à la source chaude (dans cet exemple un concentrateur solaire) et de celle apportée pendant la détente.

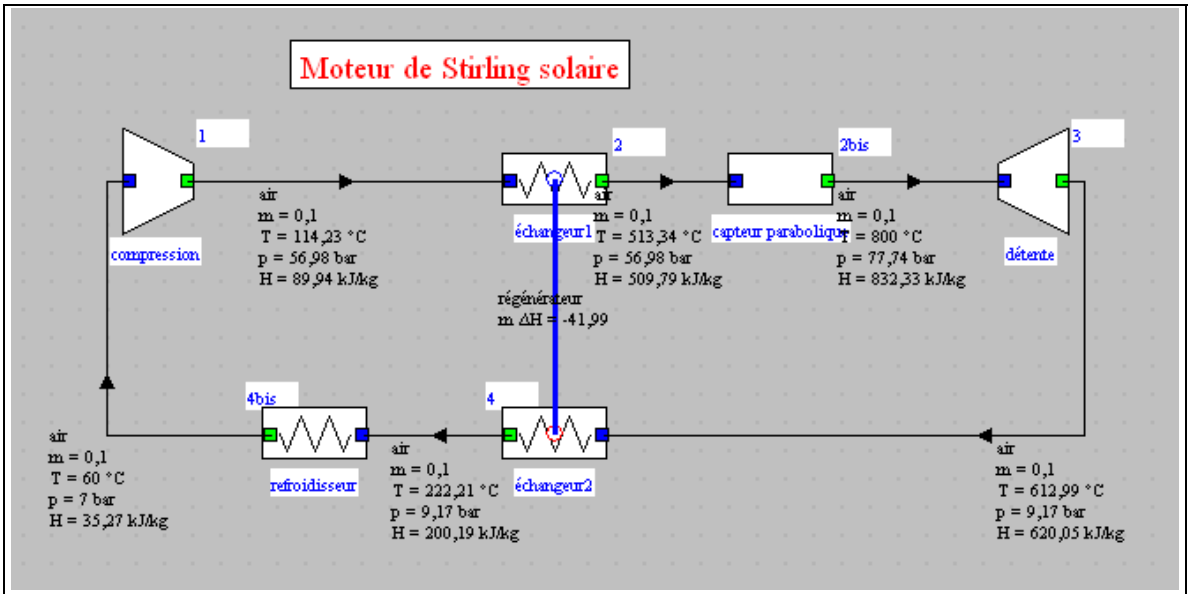

Figure 2.1 : Moteur Stirling solaire

L'objectif que nous assignons au pilote est de dresser un tableau récapitulatif des énergies mises en jeu dans le moteur, sous forme de chaleur ou de puissance mécanique, et de calculer le rendement du cycle. La figure 2.2 montre le type d'écran que l'on peut imaginer.

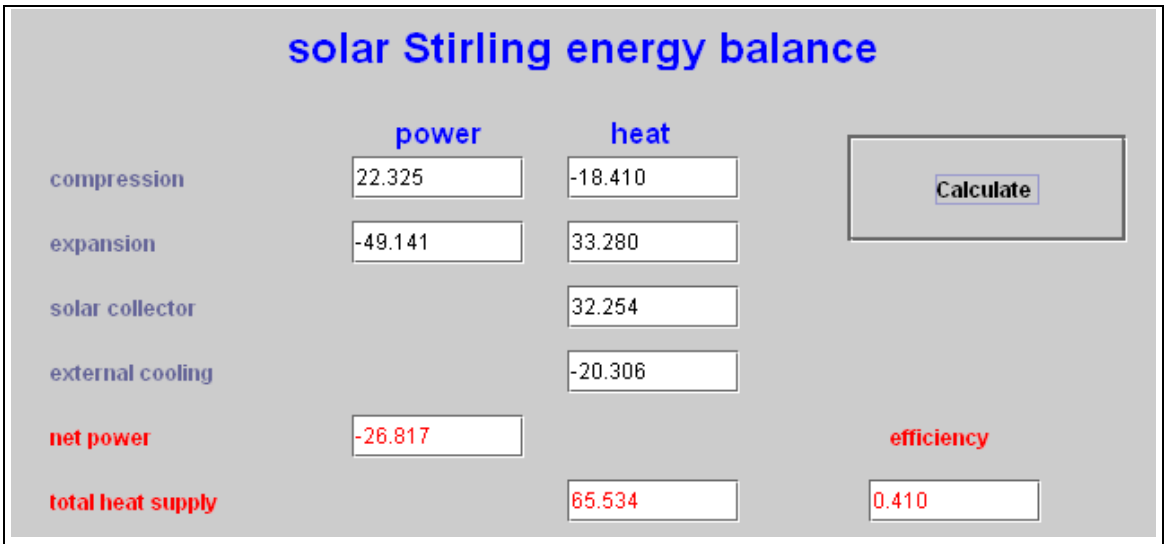

Figure 2.2 : Ecran du pilote

2.2 Implémentation informatique

2.2.1 Création de la classe, interface visuelle

Pour créer un pilote externe, il suffit de sous-classer extThopt. ExtPilot.

La réalisation de l'interface visuelle ne pose pas de problème particulier et nous ne la commenterons pas ici.
Le constructeur doit se terminer par un String précisant le code qui désignera le pilote dans la liste de ceux qui sont disponibles :

```
type="stirling";
```

Il est par ailleurs recommandé de documenter la classe :

```
public String getClassDescription(){
   return "pilot for a simple Stirling motor\n\nauthor : R. Gicquel february 2008";
}
```

2.2.2 Reconnaissance des noms des composants

Il est possible de reconnaître automatiquement les noms des divers composants constituant le modèle, en les triant par type, ce qui confère au pilote une plus grande généricité que si ces noms sont entrés sous forme de String dans le code. C'est l'objet des méthodes init() et setupProject(), qui utilisent les méthodes permettant d'accéder aux noms des composants de l'éditeur de schémas et à la liste des classes externes disponibles (pour la transfo externe représentant le capteur à concentration). Attention toutefois : si un schéma approprié n'est pas ouvert dans l'éditeur, les listes de composants sont vides et les initialisations ne peuvent se faire correctement.

```
public void init(){
isInitialized=true;
proj=getProjet();
setupProject();
//On récupère la liste et le type des composants présents dans l'éditeur de schémas
//Retrieves the list and the type of components in the diagram editor
String[]listComp=proj.getEditorComponentList();
composant=new String[listComp.length];
nomComposant=new String[listComp.length];

//on en extrait les noms des transfos de compression et de détente
//gets the names of compression et expansion processes
for(int i=0;i<listComp.length;i++){
    composant[i]=Util.extr_value(listComp[i], "type");
    nomComposant[i]=Util.extr_value(listComp[i], "name");
    if(composant[i].equals("Compression"))compressorName=nomComposant[i];
    if(composant[i].equals("Expansion"))expansionName=nomComposant[i];
}
//test de cohérence (des messages d'erreur plus précis seraient souhaitables)
//consistency test (error messages should be an improvement)
if((!expansionName.equals(""))&&(!compressorName.equals(""))&& isBuilt)isBuilt=true;
if(isBuilt)show();//on n'ouvre le pilote que si sa structure est correcte
                  //the driver cannot be open if its structure is wrong


void setupProject(){
    //on récupère ici l'instance de la transfo externe du capteur solaire
    //Retrieves the instances of the solar collector external class
    Vector vExt=proj.getExternalClassInstances();//Vector contenant les classes externes
    int j=0;
    for(int i=0;i<vExt.size();i++){
        Object[] obj=new Object[6];
        obj=(Object[])vExt.elementAt(i);
        ExtProcess ep=(ExtProcess)obj[1];
        if(ep instanceof SolarConcentratorCC){
            collector=(SolarConcentratorCC)ep;
            collectorName=collector.getName();
            j++;
        }
    }
    if(j==1)isBuilt=true;//test de cohérence du pilote par rapport au modèle
    //consistency test (error messages should be an improvement)
```

2.2.3 Calculs effectués et affichage

Une fois les noms des divers composants identifiés, on accède à leurs propriétés grâce à la méthode getProperties() de Projet, ce qui permet d'obtenir toutes les valeurs dont on a besoin.

Comme le montre le code correspondant, la réalisation d'un pilote externe ne pose pas de problème particulier. Celui-ci est particulièrement simple et se contente d'effectuer les bilans pour lesquels le mode de calcul par défaut de Thermoptim n'est pas approprié. Il serait tout à fait possible de le compliquer légèrement, pour qu'il

mette à jour le simulateur avant d'effectuer des recalculs du modèle et de dresser le bilan recherché. Nous présenterons plus loin des exemples mettant en jeu ces fonctionnalités.

```
void bCalculate_actionPerformed(java.awt.event.ActionEvent event){
    if(!isInitialized)init();//la première fois, on initialise, car il faut un constructeur sans argument
                             //pour instancier la classe par le RMI
                             //the initialization is made during the first call, as the constructor
                             //must have no argument because the class is instanciated by the RMI

    String[] args=new String[2];
    args[0]="process";
    args[1]=compressorName;//compression
    Vector vProp=proj.getProperties(args);
    String amont=(String)vProp.elementAt(1);//point amont //inlet point
    String aval=(String)vProp.elementAt(2);//point aval //outlet point
    Double f=(Double)vProp.elementAt(4);
    double deltaUcompr=f.doubleValue();//puissance compresseur //compression power
    f=(Double)vProp.elementAt(12);
    double Qcompr=f.doubleValue();//chaleur compresseur //compressor heat

    args[1]=expansionName;//détente //expansion
    vProp=proj.getProperties(args);
    amont=(String)vProp.elementAt(1);//point amont //inlet point
    aval=(String)vProp.elementAt(2);//point aval //outlet point
    f=(Double)vProp.elementAt(4);
    double deltaUexpan=f.doubleValue();//puissance détente //expansion power
    f=(Double)vProp.elementAt(12);
    double Qexpan=f.doubleValue();//chaleur détente //expansion heat

    args[1]=collectorName;//capteur solaire //solar collector
    vProp=proj.getProperties(args);
    f=(Double)vProp.elementAt(4);
    double solarHeat=f.doubleValue();//chaleur solaire //solar heat

    //calcul des performances globales du moteur et affichages
    //calculates the overall motor energies and displays them on the driver screen
    tauExpan_value.setText(Util.aff_d(deltaUexpan, 3));
    netPower_value.setText(Util.aff_d(deltaUexpan+deltaUcompr, 3));
    tauCompr_value.setText(Util.aff_d(deltaUcompr, 3));
    Q_value.setText(Util.aff_d(Qexpan+solarHeat, 3));
    eta_value.setText(Util.aff_d((-deltaUexpan-deltaUcompr)/(Qexpan+solarHeat), 3));
    expanHeat_value.setText(Util.aff_d(Qexpan, 3));
    comprHeat_value.setText(Util.aff_d(Qcompr, 3));
    solarHeat_value.setText(Util.aff_d(solarHeat, 3));
    extCooling_value.setText(Util.aff_d(-(Qexpan+solarHeat+deltaUexpan+deltaUcompr+Qcompr), 3));
```

2.2.3 Chargement d'un pilote

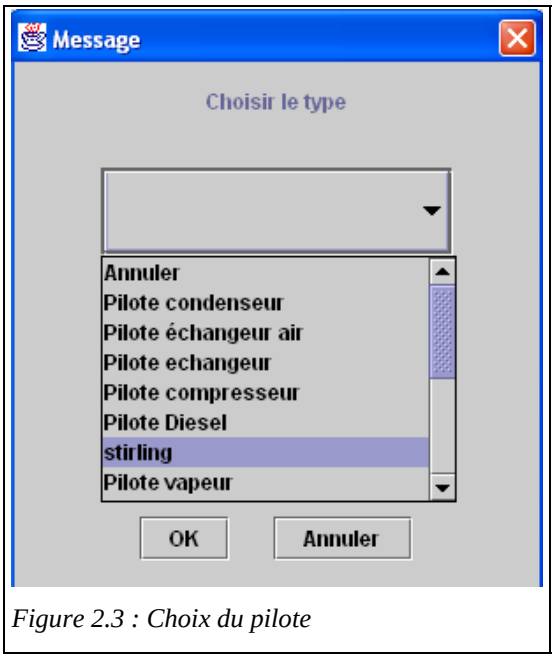

Figure 2.3 : Choix du pilote

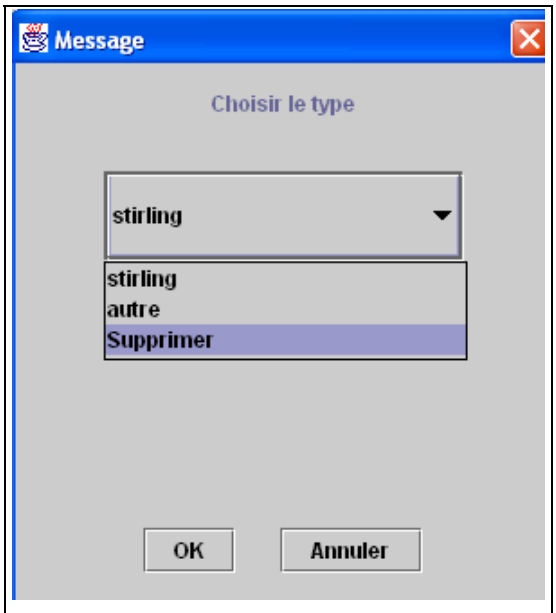

Figure 2.4 : Suppression du pilote

Le chargement d'un pilote se fait en activant la ligne "Ecran de pilotage " du menu "Spécial" du simulateur. Un menu déroulant propose alors la liste des pilotes disponibles (figure 2.3). Sélectionnez celui que vous désirez charger (ici "stirling"), puis validez.

Lorsqu'un pilote est déjà chargé, la ligne "Ecran de pilotage" vous propose l'écran de la figure 2.4, qui vous permet de revenir au pilote chargé (ligne avec son nom), d'en sélectionner un autre, ou de supprimer l'existant sans le remplacer. Un seul pilote peut être associé à un projet.

3 DIMENSIONNEMENT TECHNOLOGIQUE

Pour pouvoir effectuer des dimensionnements technologiques, nous avons introduit de nouveaux écrans, complémentaires de ceux qui permettent d'effectuer la modélisation phénoménologique. Ils peuvent être activés à partir du bouton "tech. design" placé dans les écrans habituels des différents composants.

Ces écrans permettent de définir les caractéristiques géométriques représentatives des différentes technologies utilisées, ainsi que les paramètres nécessaires pour le calcul de leurs performances. Pour un composant donné, ils dépendent bien évidemment du type de technologie retenu.

Dans la section suivante, nous présentons de manière synthétique les principes de modélisation retenus pour les principaux échangeurs, pour les compresseurs, pour les turbines et pour les détendeurs.

3.1 Dimensionnement technologique des composants

3.1.1 Echangeurs de chaleur

Les modifications les plus importantes concernent les échangeurs de chaleur, les versions antérieures de Thermoptim ne déterminant que le produit UA du coefficient global d'échange thermique U par la surface A de l'échangeur, sans que les deux termes soient évalués séparément. Pour pouvoir dimensionner un échangeur, c'est-à-dire calculer sa surface, il faut d'une part choisir sa configuration géométrique, et d'autre part calculer U, qui dépend de cette configuration, des propriétés thermophysiques des fluides, et des conditions opératoires.

Rappelons que l'approche que nous avons retenue dans Thermoptim n'est pas conventionnelle, mais qu'elle est cohérente avec d'autres approches utilisées en thermique des échangeurs, et se révèle tout à fait féconde pour l'étude de systèmes complexes : un échangeur assure le couplage entre deux transfos "échange" représentant l'une le fluide chaud qui se refroidit, et l'autre le fluide froid qui se réchauffe. Il en résulte que la définition des configurations d'écoulement et de la géométrie de l'échangeur se fait au niveau de chaque transfo, et non pas de manière globale.

3.1.1.1 Méthode de calcul

La méthode de calcul des échangeurs en régime non nominal retenue est basée sur celle du NUT. Lorsque l'on connaît ses deux températures d'entrée et ses débits, un échangeur peut être représenté par un quadripôle, dont la formulation matricielle généralisée peut s'exprimer de diverses manières, selon les températures connues (chapitre 5 du tome 1 du livre Systèmes énergétiques).

En dimensionnement, le calcul d'un échangeur se fait en trois étapes :

- connaissant les températures d'entrée et de sortie et les débits des deux fluides, on commence par déterminer l'efficacité ε, et on en déduit UA par la méthode du NUT
- on estime alors U par des corrélations dépendant de la configuration d'écoulement et de la géométrie de l'échangeur
- le calcul de la surface A s'en déduit immédiatement : A = UA/U

En régime non-nominal, si l'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 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
- détermination de l'efficacité de l'échangeur par la méthode du NUT, et calcul des températures de sortie

Le cas le plus naturel est celui où l'on connaît les températures d'entrée des deux fluides, Tce et Tfe, et leurs débits mc et mf, et où l'on cherche à savoir quelles seront les températures de sortie des deux fluides Tcs et Tfs lorsque les variables connues changent. Si l'on est capable de déterminer la valeur de U, il est possible de calculer R, puis NUT, d'en déduire ε, et de déterminer Tcs et Tfs.

Mais d'autres cas peuvent aussi être résolus : connaissant Tfe, Tfs, mc, mf et U, on peut calculer R, puis NUT, d'en déduire ε, et de déterminer les températures Tce et Tcs.

Le premier cas pouvait d'ailleurs être déjà calculé par la version phénoménologique de Thermoptim : le mode de calcul "non nominal" de l'écran de l'échangeur permet en effet de calculer, par la méthode du NUT, l'échangeur en régime non nominal si au moins deux températures sont imposées. Comme indiqué dans la section du tome 2 du manuel de référence consacrée aux échangeurs, ce mode de calcul s'applique à l'étude d'un échangeur déjà défini, dont on cherche à comprendre comment il se comporte en dehors des conditions retenues pour son dimensionnement. Thermoptim effectue une mise à jour des liens amont de l'échangeur à partir des transfos, puis effectue le calcul des températures aval et équilibre le bilan sur le plan enthalpique. Les points et les transfos associés au module sont mis à jour en fonction des résultats.

Il est donc possible d'utiliser ce mode de calcul en modifiant la valeur de UA comme l'illustrera l'exemple traité section 4.

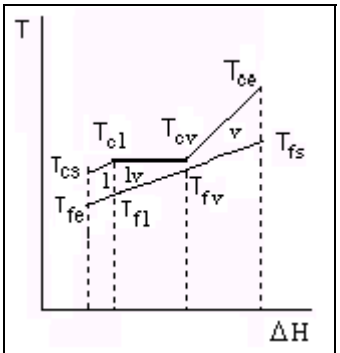

Figure 3.1 : Echangeur multizone

3.1.1.2 Echangeurs multizones

Le calcul des échangeurs multizones est plus complexe que celui des échangeurs simples, mais il repose sur le même principe. Considérons en effet le cas d'un condenseur de machine frigorifique dont le diagramme température / enthalpie est représenté figure 3.1.

Le fluide frigorigène sortant du compresseur commence par être désurchauffé (T_{ce}-T_{cv}), puis il est condensé à température constante, et enfin légèrement sous-refroidi (T_{cl}-T_{cs}). Dans ce cas, le calcul doit être mené pour chacune des zones, la surface de l'échangeur étant la somme des surfaces des zones. Bien évidemment, les coefficients d'échange thermique sont très différents dans ces trois zones.

En régime non-nominal, la surface totale reste constante, mais sa répartition entre les trois zones varie. Un calcul précis demande donc de chercher la solution qui correspond à la même surface totale et qui respecte toutes les autres contraintes thermiques et thermodynamiques.

3.1.1.3 Paramétrage géométrique

Le paramétrage géométrique des échangeurs est un sujet difficile. Nous nous contentons ici d'introduire les grandeurs utilisées, sans discuter des raisons de leur choix.

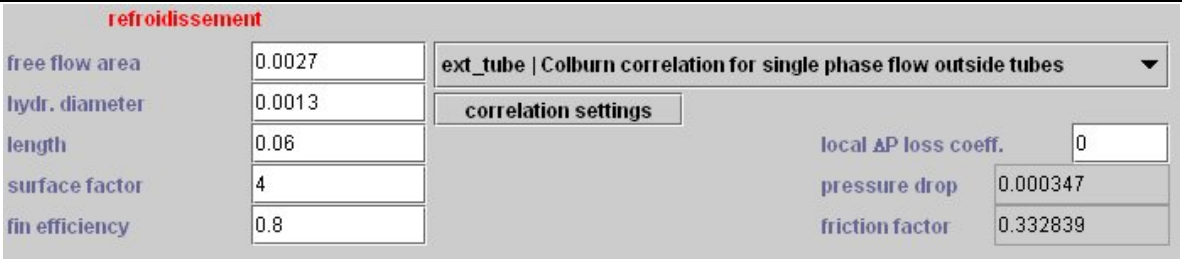

Figure 3.2 : Condensor technological design screen

L'étude du calcul des coefficients d'échange thermique et des pertes de charge montre que, outre la surface de l'échangeur A, deux grandeurs géométriques jouent un rôle particulièrement important : la section de passage dévolue au fluide $\mathbf{A_c}$, et le diamètre hydraulique $\mathbf{d_h}$. Lorsque les coefficients d'échange thermique des fluides sont très différents, on a recours à des dispositifs divers comme des ailettes pour compenser l'écart entre leurs valeurs. On parle alors de surfaces étendues, qui peuvent être caractérisées par un facteur de surface **f** et un rendement d'ailette η_0.

Ces quatre paramètres sont ceux qui ont été retenus dans Thermoptim pour caractériser les échanges thermiques au niveau de chaque fluide. On leur ajoute la longueur de l'échangeur pour certains calculs comme les pertes de charge.

Dans l'écran de dimensionnement technologique des échangeurs, les conventions suivantes sont adoptées :

- le type de configuration (ici "cond") est sélectionné dans une liste déroulante
- "free flow area" représente la section de passage A_c
- "hydr. diameter" est le diamètre hydraulique habituel d_h
- "length" est la longueur de l'échangeur (utilisée pour le calcul du nombre d'ébullition Bo et pour le calcul des pertes de charge, non encore implémenté dans le cas diphasique)
- "surface factor" est le facteur de surface f pour les surfaces étendues
- "fin efficiency" est le rendement d'ailettes η_0

3.1.1.4 Corrélations pour le calcul des coefficients d'échange thermiques

TABLEAU 12.1	TYPE	CORRELATION	TYPE DE FLUIDE
"int_tube"	intérieur tubes	Mac Adams	monophasique
"ext_tube"	extérieur tubes	Colburn (Dittus Boettler)	monophasique
"cond"	condensation int. tubes	Shah/Bivens	diphasique
"cond_ext_hor"	cond. ext. tubes horiz.	B1540 T/I	diphasique
"evap"	évaporation	Gungor Winterton	diphasique
"air_coil"	batterie à air	Morisot	air
"cool_tower"	refroidissement évaporatif	Colburn	air humide
"flooded"	noyé	générique	diphasique
"plate"	à plaques	générique	monophasique

Une des principales difficultés est le calcul du coefficient d'échange global U, qui dépend des coefficients d'échange thermiques h de chaque fluide, par la formule générale ci-dessous. De nombreuses corrélations ont été proposées dans la littérature, et le choix de celle qui est le mieux adaptée n'est pas toujours évident.

$$\frac{1}{U_c A_c} = \frac{1}{A_c \eta_{0,c} h_c} + \frac{e}{A \lambda} + \frac{1}{A_f \eta_{0,f} h_f}$$

Les options offertes par défaut par Thermoptim sont données dans le tableau ci-dessus. Il récapitule les configurations proposées, en indiquant leur type, les corrélations utilisées, et le type de fluide auxquelles elles correspondent. Il est bien sûr possible d'en introduire d'autres sous forme de classes externes additionnelles.

Attention à un point important : pour que les calculs aient un sens, il faut impérativement que les unités du projet Thermoptim soient exprimées dans le système SI, c'est-à-dire que le débit soit exprimé en kg/s. Vérifiez bien ce point, en faisant de préférence un test sur l'unité dans le pilote.

3.1.2 Compresseurs

3.1.2.1 Compresseurs volumétriques

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

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

$$\lambda = a_0 - a_1 \frac{P_{ref}}{P_{asp}}$$

$$\eta_s = K_1 + K_2 \cdot \left(\frac{P_{ref}}{P_{asp}} - R_1 \right)^2 + \frac{K_3}{\frac{P_{ref}}{P_{asp}} - R_2}$$

Le calcul d'un compresseur correspond aux étapes ci-dessous :

- les rendements volumétrique et isentropique λ et η_s sont calculés à partir de leurs équations
- si l'on connaît la vitesse de rotation, la cylindrée et le rendement volumétrique, le débit volumique peut être calculé :

$$\dot{V} = \frac{\lambda \, N \, V_s}{60}$$

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

$$\dot{m} = \frac{\dot{V}}{v} = \frac{\lambda \, N \, V_s}{60 \, v}$$

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 saisi dans le champ du débit du compresseur.

En régime non-nominal, le rapport de compression détermine λ et η_s, ce qui fixe le débit et la température de sortie compresseur. La séquence des calculs est la suivante :

- 1) connaissant les conditions d'entrée et de sortie, le rapport de compression P_{ref}/P_{asp} et le volume v à l'aspiration sont mis à jour ;
- 2) les rendements volumétrique et isentropique λ et η_s et le débit réel sont calculés connaissant la vitesse de rotation N ;
- 3) le débit volumique $\dot{V}$ s'en déduit ;
- 4) le compresseur impose $\dot{M} = v \, \dot{V}$ et le propage aux composants connectés ;
- 5) le rendement isentropique η_s permet alors de calculer le travail utile mis en jeu.

3.1.2.2 Turbocompresseurs

Les turbocompresseurs peuvent être modélisés de plusieurs manières. Nous nous baserons sur une approche par similitude faisant appel à deux caractéristiques : le facteur de débit φ, et le facteur d'enthalpie ψ,

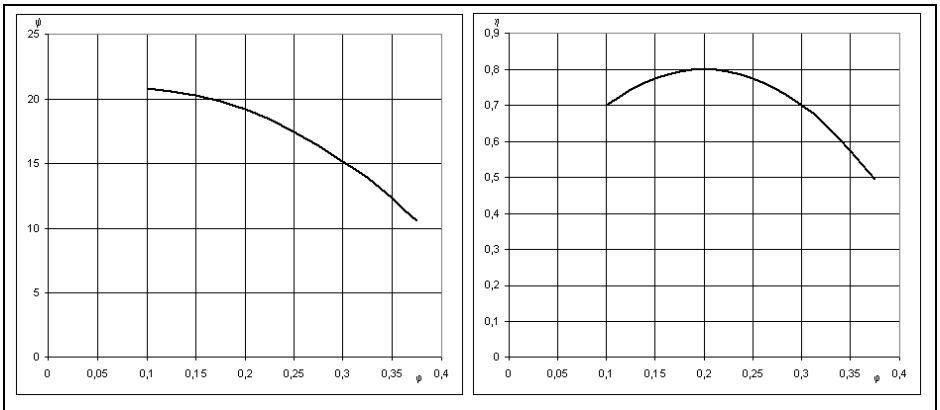

Figure 3.3 Caractéristiques réduites simplifiées d'un turbocompresseur

et nous nous limiterons ici à faire l'hypothèse simplificatrice que leurs caractéristiques réduites(ψ, φ) et (η, φ) peuvent être représentées par des courbes simples du type parabole (figure 3.3).

$$\varphi = \frac{(M_a)_c}{(M_a)_u} = \frac{240\ \dot{m}\ r\ T_a}{\pi^2\ D^3\ P_a\ N}$$

$$\psi = \frac{|\Delta h_s|}{1/2\ U^2} = \frac{7\ 200\ |\Delta h_s|}{\pi^2\ D^2\ N^2}$$

D'autres modèles sont fournis en annexe.

3.1.3 Turbines

Une turbine est définie géométriquement par une section, et ses paramètres technologiques permettent de calculer son rendement isentropique et la constante du cône, en fonction des conditions d'aspiration et de refoulement.

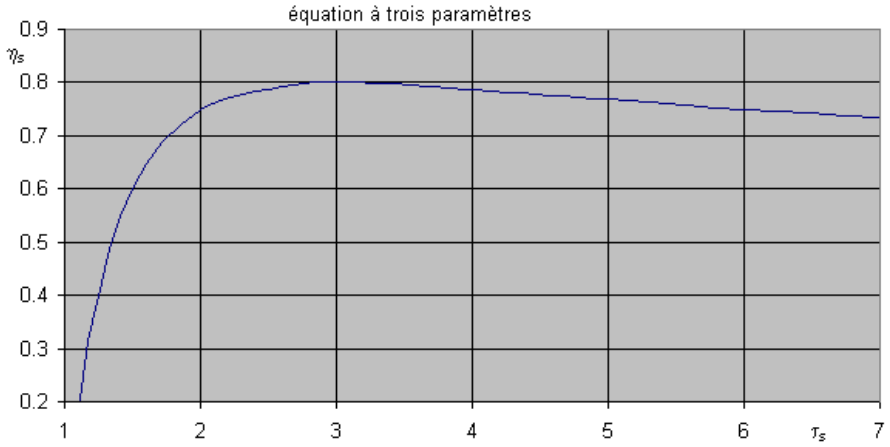

Figure 3.4 : Equation à trois paramètres

Pour le rendement isentropique, il est souvent possible de retenir une équation analogue à la loi des compresseurs volumétriques.

$$\eta_p = \eta_{lim} + (\eta_{max} - \eta_{lim}) \cdot \left(2\left[\tau_{max}\frac{P_{ref}}{P_{asp}}\right] - \left[\tau_{max}\frac{P_{ref}}{P_{asp}}\right]^2 \right)$$

On notera qu'elle s'exprime linéairement en fonction du rapport de détente et de son carré. Ses paramètres présentent l'intérêt d'avoir un sens physique.

η_{lim} est la valeur asymptotique du rendement isentropique pour les hauts rapports de détente, et η_{max} la valeur maximale du rendement, obtenue pour un rapport de détente égal à τ_{max}.

Le calcul du débit se fait en utilisant la règle de Stodola, qui peut s'exprimer sous la forme suivante, K_0 étant la constante du cône :

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

Généralement, on prendra k=1, ce qui correspond à une forme quadratique. Si le débit est choqué, cette relation se simplifie sous la forme $\frac{\dot{m}\sqrt{T_{in}}}{P_{in}}$ = Cste.

D'autres modèles sont fournis en annexe.

3.1.4 Détendeurs

Le dimensionnement technologique des détendeurs n'est pris en compte que de manière sommaire, car on considère que leur dynamique est beaucoup plus rapide que celle des autres composants du cycle, et qu'en conséquence ils se comportent comme des organes de régulation.
Le seul paramètre introduit à ce niveau est la valeur de la surchauffe.

3.2 Implémentation informatique

Les structures des classes externes permettant de traiter le dimensionnement technologique et le non nominal pour les divers composants de Thermoptim présentent un certain nombre de similitudes avec celles qui ont été exposées au tome 3, mais aussi quelques différences qui s'expliquent par leurs caractéristiques propres, et que nous présenterons en temps utile.

Les principaux développements ayant porté sur les compresseurs et les échangeurs de chaleur, c'est eux qui seront présentés ici.

3.2.1 Principes généraux

Tout d'abord, il faut garantir la cohérence avec les classes "phénoménologiques" de Thermoptim, qui doivent pouvoir effectuer leurs calculs de la même manière, que les écrans technologiques soient présents ou non.

Ceci implique en particulier que l'affichage de l'écran technologique doit être distinct des écrans "phénoménologiques".

Ensuite, nous avons décidé d'externaliser le plus possible l'implémentation informatique afin que les utilisateurs puissent personnaliser les calculs autant qu'ils le souhaitent. Il est en effet quasiment impossible d'une part de trouver des formulations complètement génériques compte tenu de la multiplicité des solutions technologiques envisageables, et d'autre part de mettre au point des algorithmes suffisamment robustes pour résoudre les jeux d'équations qui peuvent en découler. En externalisant ces classes, les utilisateurs peuvent les adapter relativement aisément aux problèmes spécifiques auxquels ils sont confrontés.

Nous nous sommes donc contentés de mettre à leur disposition une infrastructure de travail leur facilitant significativement la tâche.

Par principe donc, nous considérons que les calculs de dimensionnement technologique et de non nominal doivent être effectués dans des classes externes, de telle sorte que celles que nous avons intégrées dans le noyau servent essentiellement à permettre la communication entre ce dernier et les classes externes.

En effet, les objets du noyau n'étant pas directement accessibles depuis l'extérieur, les échanges d'informations demandent à être faits en suivant des règles un peu contraignantes, que nous avons cherché à rendre aussi souples et conviviales que possibles, davantage encore que ne le permettaient les méthodes de communication présentées dans le tome 3. Les méthodes que nous avons implémentées à cet effet sont documentées en annexe.

3.2.2 Structure retenue

Dans les écrans de composants, le bouton "tech. design" donne accès à l'écran de paramétrage technologique. Les écrans technologiques peuvent être gérés de manière centralisée, à partir de la ligne de menu "Technological design screens" du menu "Technological Design" du simulateur.

Une classe TechnoDesign a été créée comme classe mère des écrans de paramétrage technologique. Elle est sous-classée par les écrans spécifiques de chaque type de composant, selon le principe présenté section 2.3 du tome 3.

Le dimensionnement technologique est effectué dans tous les TechnoDesign par la méthode makeDesign().

Dans les TechnoDesign, des classes dérivées de JPanel permettent de modifier les paramètres en fonction du choix technologique effectué.

Des classes externes mères représentatives des principaux types de composants ont été préparées. Elles sont destinées à être sous-classées en fonction des besoins des utilisateurs.

La seule manière d'instancier les TechnoDesign destinés à être reconnus par le noyau est de le faire dans un pilote externe, puis de les charger dans le Vector vTechno de ce pilote. Dès lors, ils deviennent visibles depuis le noyau et peuvent être associés à leurs composants. Ce choix signifie que **les pilotes externes jouent un rôle prépondérant pour le dimensionnement technologique et le non nominal**.

La sauvegarde des paramétrages technologiques est effectuée à la fin des fichiers de projet, à partir du pilote. La dernière ligne doit se terminer par un retour à la ligne ("\n"). Certains TechnoDesign peuvent occuper plusieurs lignes (c'est le cas pour les échangeurs, qui en utilisent une pour les résultats d'ensemble, et une pour chaque fluide, et pour les thermocoupleurs qui en utilisent deux).

Toutes les classes de dimensionnement technologique héritent de la classe extThopt. ExtTechnoDesign qui possède un JPanel appelé JPanel1, dans lequel est conçue l'interface graphique.

Elle dispose de deux constructeurs :

```
public ExtTechnoDesign(Projet proj, String componentName, PointThopt amont, PointThopt aval){
}

public ExtTechnoDesign(Projet proj, String componentName){
}
```

Le premier suppose que l'utilisateur définisse ses propres PointThopt amont et aval, tandis que le second les instancie automatiquement, à partir des propriétés du projet, ce qui lui donne une plus grande généricité.

Les PointThopt présentés en annexe, qui sont en quelque sorte des clones externes des points du noyau, lui donnent accès aux états amont et aval. Le constructeur instancie un CorpsThopt (voir annexe) pour permettre d'effectuer certains calculs thermodynamiques.

La méthode updateDeltaH() sert à mettre à jour l'enthalpie mise en jeu dans le composant. Cette classe possède aussi une méthode getFlow(), qui permet d'actualiser le débit à partir de celui du noyau.

Nous commentons ici deux exemples d'implémentation, relatives aux compresseurs volumétriques et aux échangeurs, qui seront utilisés dans les exemples de ce tome. Les autres sont construits de manière analogue.

3.2.2.1 Compresseurs volumétriques

La classe VolumCompr comporte :

- une méthode, setupPanel(), qui construit le JPanel pour l'interface graphique de l'écran technologique, qui n'est pas détaillé ici. Cette interface permet d'effectuer les paramétrages nécessaires.
- des méthodes génériques qui permettent de calculer le rendement isentropique et le débit, et de les afficher à l'écran.

Les constructeurs sont les suivants :

```
public VolumCompr(Projet proj, String compressorName, PointThopt amont, PointThopt aval){
                super(proj, compressorName, amont, aval);
        this.compressorName=componentName;
        setupPanel();
}

public VolumCompr(Projet proj, String compressorName){
                super(proj, compressorName);
        this.compressorName=componentName;
        setupPanel();
}
```

Ils font appel aux constructeurs d'ExtTechnoDesign, puis, pour pouvoir utiliser le nom plus spécifique de compressorName, ils l'identifient à componentName, et enfin, ils construisent le JPanel.

Cette classe définit en particulier les deux méthodes de calcul des rendements isentropique et volumique:

```
double getRisentr(){
        K1=Util.lit_d(K1_value.getText());
        K2=Util.lit_d(K2_value.getText());
        K3=Util.lit_d(K3_value.getText());
        R1=Util.lit_d(R1_value.getText());
        R2=Util.lit_d(R2_value.getText());

        double r_compr=aval.P/amont.P;
        double risentr=0.5;
        if(R1*R1+R2*R2==0)risentr=K1+K2/r_compr+K3/r_compr/r_compr;//3 parameter equation
        else risentr=K1+K2*(r_compr-R1)*(r_compr-R1)+K3/(r_compr-R2);
        risentr_value.setText(Util.aff_d(risentr,7));
        return risentr;
}

double getLambdaVol(){
        a0=Util.lit_d(a0_value.getText());
        alpha=Util.lit_d(alpha_value.getText());

        double r_compr=aval.P/amont.P;
        double lambda= a0-alpha*r_compr;
        display_value.setText(Util.aff_d(lambda,7));
        return lambda;
}
```

La sauvegarde des valeurs des paramètres se fait grâce à la méthode saveCompParameters(), qui, comme nous l'avons dit plus haut, doit se terminer par un caractère de retour à la ligne.

```
public String saveCompParameters(){
        String h="alpha_value="+alpha_value.getText()+tab
        +"K1_value="+K1_value.getText()+tab
        +"K2_value="+K2_value.getText()+tab
        +"K3_value="+K3_value.getText()+tab
        +"R1_value="+R1_value.getText()+tab
        +"R2_value="+R2_value.getText()+tab
        +"N_value="+N_value.getText()+tab
        +"Vs_value="+Vs_value.getText()+tab
        +"Device_number="+deviceNumber_value.getText()
        +"\n";//retour à la ligne impératif
   return h;
}
```

3.2.2.2 Echangeurs de chaleur

Une différence notable entre un échangeur et un compresseur est que le second est un composant unique, alors que le premier représente un couplage thermique entre deux transfos échange. En revanche, les équations de couplage thermique sont les mêmes pour tous les types d'échangeurs que Thermoptim peut calculer, avec une exception : les échangeurs multizones se présentent souvent comme des batteries d'échange dont les zones liquide, diphasique et vapeur ont des limites évolutives dans le temps. Il est donc impossible d'affecter à chacune d'entre elles une surface déterminée : il faut les recalculer en permanence, même si la surface totale est connue.

Le TechnoDesign des échangeurs s'appelle TechnoHx. En mode design, il calcule le U et propose donc une valeur de A, surface de l'échangeur. Il effectue ses calculs en estimant l'ensemble des propriétés thermophysiques des fluides. Le chapitre 13 du tome 3 du livre Systèmes Energétiques précise comment celles-ci sont déterminées.

Le TechnoDesign pour les transfos de type Exchange est TechnoExch, qui permettra aussi de calculer les pertes de charge lorsque les méthodes seront implémentées. Pour garantir un traitement homogène entre les échangeurs et les deux transfos qu'il connecte, TechnoHx fait directement appel aux deux TechnoExch associés, qu'il charge sur son écran. Les TechnoExch des transfos d'un échangeur ne sont donc pas instanciées indépendamment de son TechnoHx.

Différentes configurations d'écoulement ont été introduites : elles héritent toutes de FlowConfig et sont sélectionnées au niveau des TechnoExch. Ce sont elles qui assurent le calcul des propriétés thermophysiques et du nombre de Nüsselt, notamment pour les échangeurs multizones.

En résumé :
- TechnoHx introduit la surface de l'échangeur, en fait le dimensionnement (makeDesign()), gère les affichages des calculs des propriétés thermophysiques et instancie les TechnoExch froid et chaud
- les TechnoExch froid et chaud font les calculs des propriétés thermophysiques et gèrent les FlowConfig grâce à des ComboBox. Les JPanel des TechnoHexch apparaissent dans la fenêtre du TechnoHx.
- les FlowConfig implémentent les corrélations qui permettent de calculer les coefficients d'échange et de pertes de charge
- le SettingsFrame d'un FlowConfig peut permettre de modifier ses paramètres.

Le constructeur de la classe TechnoHx est le suivant. Il diffère du précédent du fait de la structure particulière de la classe. Il commence par récupérer les noms des transfos chaude et froide qu'il apparie, instancie quatre PointThopt pour connaître les états amont et val des deux fluides, et construit leurs écrans technologiques. Il indique ensuite à chacun d'eux quel est son homologue. Enfin, il configure son écran.

```
public TechnoHx(Projet proj, String hXname, PointThopt amontHot, PointThopt avalHot, PointThopt
amontCold, PointThopt avalCold){
                        this();
                        this.proj=proj;
                        this.hXname=hXname;
        String[] args=new String[2];
        args[0]="heatEx";
        args[1]=hXname;
        Vector vProp=proj.getProperties(args);
        hotFluid=(String)vProp.elementAt(0);
        coldFluid=(String)vProp.elementAt(1);

        this.amontHot=amontHot;
        this.amontCold=amontCold;
        this.avalHot=avalHot;
        this.avalCold=avalCold;

        techc=new TechnoExch(proj, this, hotFluid, amontHot, avalHot);
        techf=new TechnoExch(proj, this, coldFluid, amontCold, avalCold);

        techc.setOtherTechnoExch();
        techf.setOtherTechnoExch();

             JLabCompName.setText(hXname);
                hotName.setText(hotFluid);
                coldName.setText(coldFluid);
                setupPanel();
        setupTechHx();
}
```

La classe implémente ensuite toute une série de méthodes permettant d'effectuer les calculs thermodynamiques.

Le constructeur des TechnoDesign des transfos "échange" est analogue à celui des compresseurs. Il en existe en fait trois variantes, leur signature différant du fait de la nécessité pour eux de savoir s'ils sont rattachés à un échangeur (TechnoHx) , à un thermocoupleur (TechnoT) ou ils sont isolés.

```
public TechnoExch(Projet proj, TechnoHx tHx, String exchangeName, PointThopt amont, PointThopt aval){
                        this(proj, exchangeName, amont, aval);
                        this.tHx=tHx;
}
```

```
public TechnoExch(Projet proj, String exchangeName, PointThopt amont, PointThopt aval){
                    super(proj, exchangeName, amont, aval);
                    this.exchangeName=componentName;
            setupPanel();
                    JLabCompName.setText(exchangeName);
      fc=new IntTubeConfig(this);
      setupListeConf();
}
```

TechnoExch étant la classe de dimensionnement technologique des transfos Echange, elle joue un rôle essentiel dans l'étude des échangeurs.

Les variables globales qu'elle définit sont les suivantes :

```
   boolean diph;
   TechnoHx tHx;
   TechnoExch otherTe;
   TechnoTC tTc;
   public FlowConfig fc;
   SettingsFrame sf;
   String[]listeConf;
```

diph permet d'indiquer que le calcul des coefficients d'échange doit être fait en diphasique.
tHx est la classe TechnoHx appelante lorsque le TechnoExch est un élément d'un échangeur.
otherTe est le second TechnoDesign d'un échangeur dans ce cas.
tTc est la classe TechnoTC appelante lorsque le TechnoExch est un élément d'un thermocoupleur.
fc est le FlowConfig définissant la configuration d'écoulement.
sf est l'écran de paramétrage des coefficients des corrélations du FlowConfig
listeConf est la liste des configurations d'écoulement disponibles qui permet à l'utilisateur de spécifier le type de configuration d'écoulement dans lequel circule le fluide, et donc les équations à retenir pour calculer les coefficients d'échange thermiques.

La méthode setupListeConf() construit dynamiquement la liste des FlowConfig lors du chargement de Thermoptim, en fonction des classes externes qui les définissent. Les codes définissant ces configurations d'écoulement et leur descriptif sont chargés dans une liste déroulante pour que l'utilisateur puisse faire son choix. En cas de doute, si la liste est trop étroite, il peut afficher l'écran de paramétrage correspondant en cliquant sur "correlation settings", qui lui permet de lire l'ensemble du descriptif (cf. figure 3.11).

Le constructeur de la classe mère est donné ci-dessous. Il associe au FlowConfig le TechnoExch qui fait appel à lui, et initialise certaines grandeurs nécessaires au calcul du nombre de Nüsselt.

```
public FlowConfig (TechnoExch te){
      this.te=te;
      C1=0.023;a_Re=0.8;b_Pr=0.4;c_Visc=0.14;RR=0.001;
      te.JLabel13.setText("length");
      otherTe=te.otherTe;
 }
```

Pour rajouter une autre configuration, il suffit de sous-classer FlowConfig et d'attribuer un code qui n'existe pas déjà, puis de placer la classe dans extThopt.zip ou extUser.zip. Elle sera automatiquement incluse dans listeConf.

Les pertes de charge sont calculées dans le FlowConfig de la manière suivante.

En régime laminaire, les pertes de charge sont proportionnelles au débit et à la viscosité du fluide : $f = 64/Re$.
Pour des tubes lisses, f est donné par la formule de Blasius ($f = 0{,}316\ Re^{-0{,}25}$) si $Re < 30\ 000$, ou bien par la formule suivante ($0{,}0032 + 0{,}221\ Re^{-0{,}237}$) pour des valeurs de Re supérieures.

Pour les tubes rugueux en régime turbulent, ($Re > 2100$) le coefficient de frottement est donné par l'équation de Colebrook :

$$\frac{1}{\sqrt{f}} = -0{,}868589 \ln\left[\frac{RR}{3{,}71} + \frac{2{,}51}{Re\sqrt{f}}\right]$$

$$RR = \frac{\varepsilon}{D}$$ ε = rugosité absolue du tube

On peut montrer que cette équation implicite en f peut être approximée par :

$$\frac{1}{\sqrt{f}} = -0{,}78173 \ln\left[\left[\frac{RR}{3{,}71}\right]^{1{,}1} + \frac{6{,}9}{Re}\right]$$

Les paramétrages du SettingsFrame permettent d'opter pour un calcul à partir de la formule de Colebrook si cette option est sélectionnée, ou de type Blasius autrement.

En régime diphasique, le calcul des pertes de charge peut se révéler très complexe, notamment du fait qu'il faut commencer par déterminer quelles sont les longueurs des différentes zones de l'échangeur.

Pour la zone d'équilibre liquide-vapeur, nous avons utilisé la formulation approchée pour les GV nucléaires proposée par Herre et Gallori, modifiée pour tenir compte d'un titre d'entrée non nul.

$$\Delta P_{diff} = \varphi \, \Delta P_{liq}$$

$$\varphi = \left[1 + |x_s - x_e| \left(\frac{\rho_l}{\rho_v} - 1\right)\right]\left[1 + |x_s - x_e| \left(\frac{\mu_l}{\mu_v} - 1\right)\right]^{-0{,}2}$$

La perte de charge est ainsi calculée en utilisant ce facteur de renforcement par rapport à l'état saturé liquide.

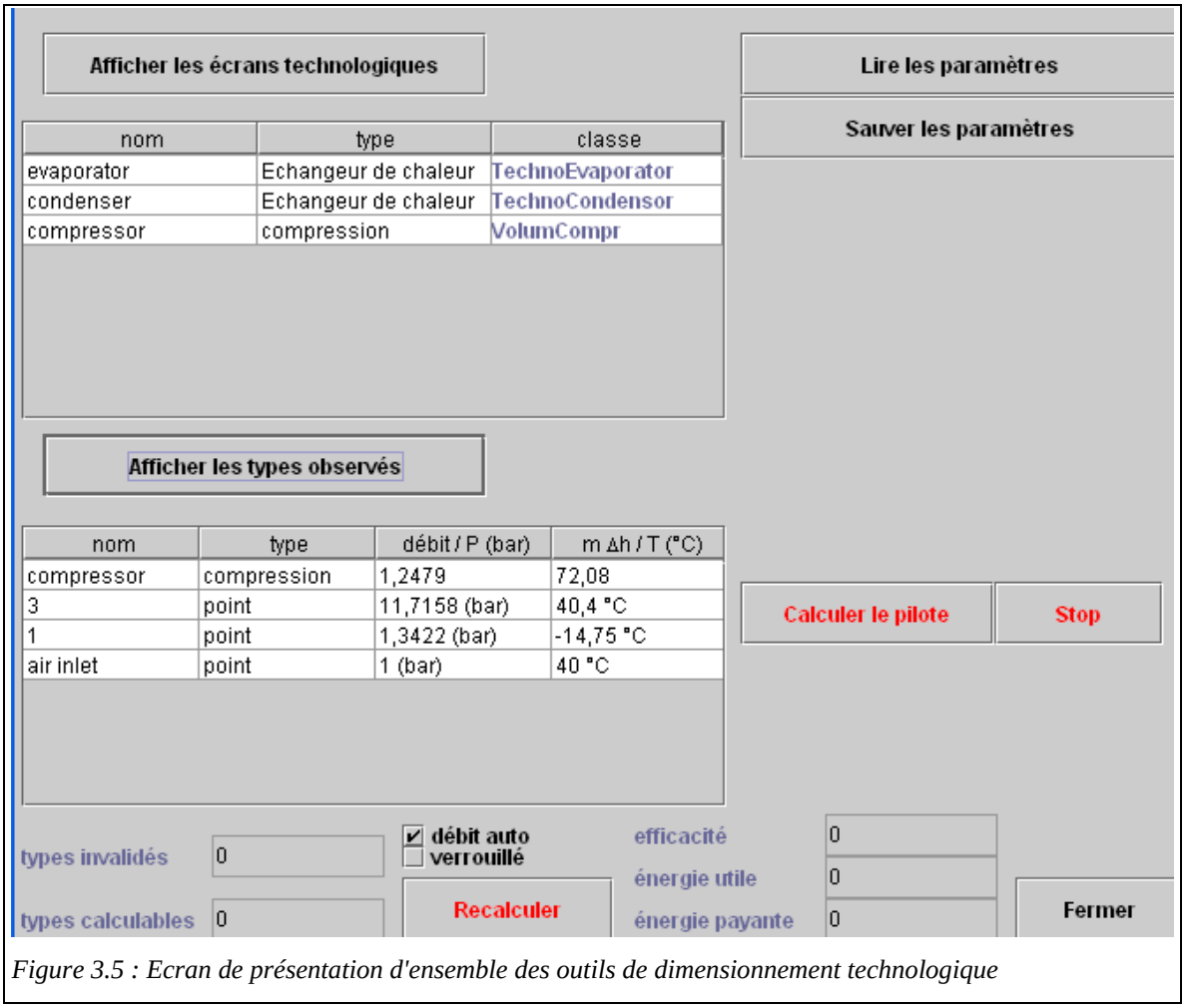

Figure 3.5 : Ecran de présentation d'ensemble des outils de dimensionnement technologique

3.2.3 Ecran d'affichage technologique et de simulation

Un écran d'affichage technologique et de simulation a été ajouté au simulateur. Il est accessible depuis la ligne "Ecrans de dimensionnement technologique" du menu "Dimensionnement technologique" (figure 3.5) du simulateur, ou en tapant Ctrl T. Il comporte deux tables principales placées dans sa partie gauche. Celle du haut permet d'accéder à l'ensemble des TechnoDesign instanciés par le pilote et chargés dans le Vector vTechno. Un double-clic sur une des lignes ouvre le TechnoDesign sélectionné. Celle en-dessous permet d'avoir un accès rapide à certains points ou transfos de Thermoptim, que l'on qualifie d'observées parce que l'on désire observer leur état thermodynamique.

Les boutons et affichages du bas de l'écran facilitent le recalcul automatique, qui peut être déclenché sans revenir à celui du simulateur.

Les deux boutons "Calculer le pilote" et "Stop" permettent de lancer et d'arrêter les calculs du pilote sans pour autant bloquer l'accès à Thermoptim (on dit qu'il s'exécute dans un thread séparé), ce qui présente l'avantage qu'il est possible d'accéder à toutes les fonctions du progiciel pendant qu'il effectue les opérations programmées dans le pilote. Le bouton "Stop" permet d'interrompre ces calculs en cas de besoin.

Pour pouvoir utiliser cette fonctionnalité, il faut que la méthode de calcul du pilote s'appelle calculatePilot(). Bien évidemment, il est recommandé de ne pas effectuer de modifications du paramétrage pendant que le pilote s'exécute, pour éviter toute interférence.

Les deux boutons situés en haut à droite, "Sauver les paramètres" et "Lire les paramètres", vous permettent de sauvegarder un paramétrage technologique réalisé avec un pilote, et de le relire avec un autre semblable, ce qui évite d'avoir à ré-entrer toutes les valeurs chaque fois que l'on change de pilote.

3.2.4 Pilote générique pour la création des écrans technologiques

Nous montrerons dans la section 4 comment créer des écrans technologiques en construisant des pilotes, ce qui suppose un minimum de programmation. Ce travail est impératif lorsqu'on veut effectuer des calculs en régime non-nominal, car il est alors nécessaire de prendre la main sur le moteur de recalcul de Thermoptim.

En revanche, lorsque l'on se contente de vouloir effectuer le dimensionnement technologique d'un projet qui ne met en œuvre que des composants du noyau de Thermoptim, il est possible de créer automatiquement les écrans technologiques en utilisant le pilote générique que nous allons brièvement présenter, ce qui évite d'avoir à en programmer un.

Ouvrez le projet Thermoptim, et chargez le pilote générique en choisissant, dans la liste des pilotes, celui dont l'intitulé est "generic techno design pilot", puis cliquez sur le bouton "Set the technological design screens" (figure 3.6). La liste des composants pour lesquels existent des écrans technologiques apparaît dans la table, avec leur nom, leur type et le nom de la classe d'écran technologique instanciée par défaut : VolumCompr pour le compresseur, et TechnoHx pour les échangeurs de l'exemple de la figure 3.6.

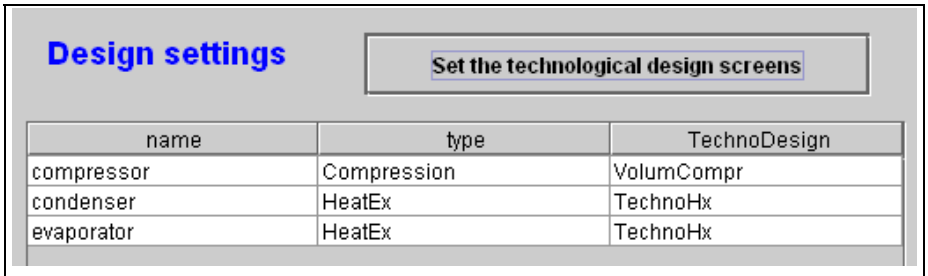

Figure 3.6 : Ecran du pilote générique (initialisation par défaut)

Dans cet exemple, cette initialisation convient pour le compresseur, mais pas pour les échangeurs, le condenseur devant être modélisé par la classe TechnoCondensor, et l'évaporateur par TechnoEvaporator.

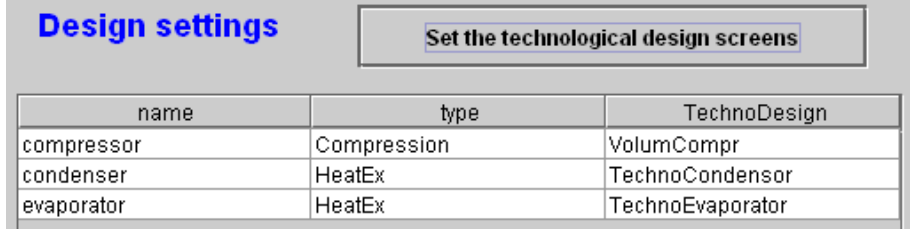

Figure 3.7 : Ecran du pilote générique après sélection

Pour changer la classe du TechnoDesign d'un composant, sélectionnez sa ligne, puis double-cliquez. Un message vous demande de confirmer votre intention, puis vous propose la liste des classes disponibles. Choisissez celle que vous désirez et validez votre choix. Après avoir modifié les classes des deux échangeurs, vous obtenez l'écran de la figure 3.7.

Vous pouvez alors accéder aux écrans technologiques de ces composants soit depuis leurs propres écrans (bouton "tech. Design"), soit à partir de l'écran d'affichage technologique et de simulation présenté section précédente, qui s'ouvre depuis l'écran du simulateur (figure 3.8).

3.3 Exemple de dimensionnement d'échangeur

3.3.1 Présentation et résultats obtenus

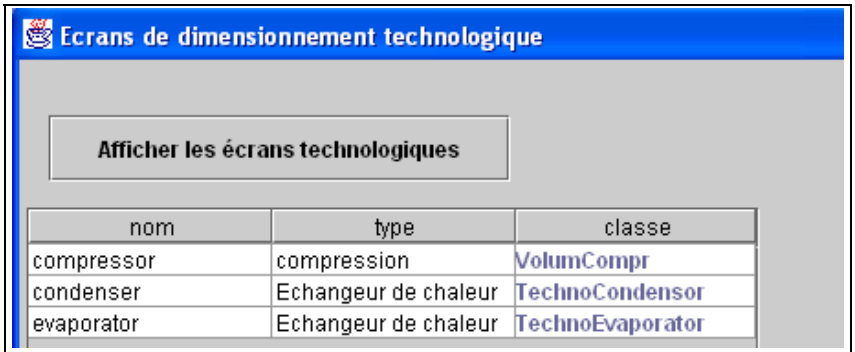

Figure 3.8 : Fenêtre des écrans technologiques

Considérons un échangeur à tubes et ailettes destiné à refroidir environ 0,66 kg/mn d'air sortant d'un compresseur à 5 bars et 275 °C grâce à un débit de 1,17 kg/mn d'eau froide passant dans un serpentin de deux tubes en parallèle. Le diamètre des tubes est de 15 mm, et leur épaisseur de 1,5 mm. L'espacement entre les ailettes est de 3 mm.

L'échangeur peut être facilement modélisé dans Thermoptim.

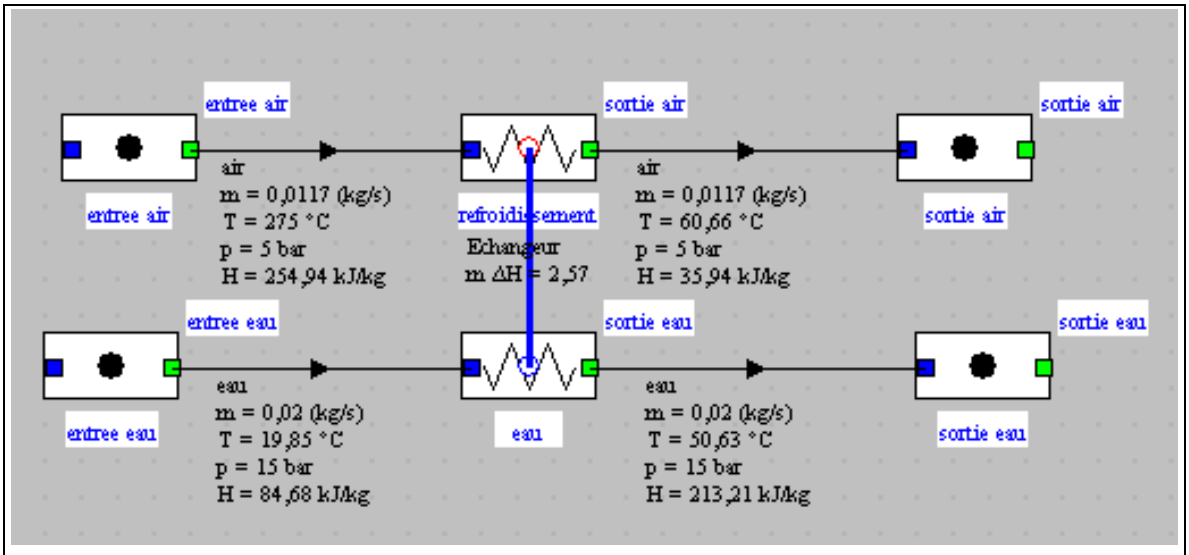

Figure 3.9 : Synoptique de l'exemple

Pour dimensionner l'échangeur, on se fixe une efficacité de 0,84.

En utilisant les écrans de dimensionnement technologique de Thermoptim, il est possible de calculer le coefficient d'échange thermique global et d'en déduire la surface nécessaire pour transmettre la puissance souhaitée.

Il faut pour cela déterminer les diamètres hydrauliques dh et les sections de passage Ac des deux fluides.

A l'intérieur des tubes, dh est donné, et le calcul de Ac est très simple : il est égal au produit du nombre de tubes par la section unitaire. A l'extérieur des tubes, les calculs sont un peu plus compliqués. dh est égal à 4 fois la section de passage par le périmètre mouillé. Quant à Ac, c'est la surface transversale offerte à l'air.

On a supposé que les ailettes multipliaient par 4 la surface d'échange, avec une efficacité égale à 0,8.

Ces valeurs doivent être entrées dans l'écran de dimensionnement technologique de l'échangeur. Après quelques tâtonnements, on obtient un paramétrage proche de celui de la figure 3.11.

Le pilote peut alors calculer la surface nécessaire, qui est ici égale à 0,067 m^2, les coefficients d'échange valant 669.3 $W/m^2/K$ côté air, et 759.1 côté eau, le tout conduisant donc à un coefficient global égal à 355.7 $W/m^2/K$.

L'échangeur sera donc constitué de deux tubes de 90 cm disposés en serpentin sur 3 niveaux et traversant des plaques de tôle d'une surface étendue totale de 0,31 m^2, séparées les unes des autres de 3 mm.

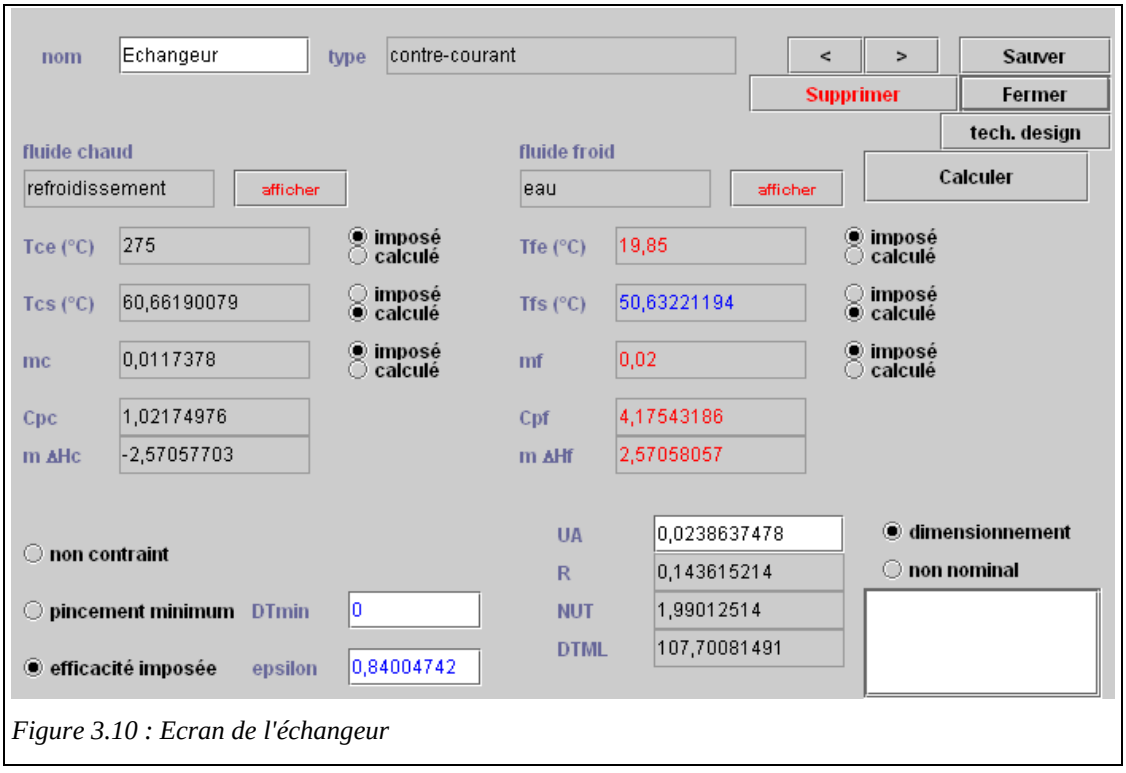

Figure 3.10 : Ecran de l'échangeur

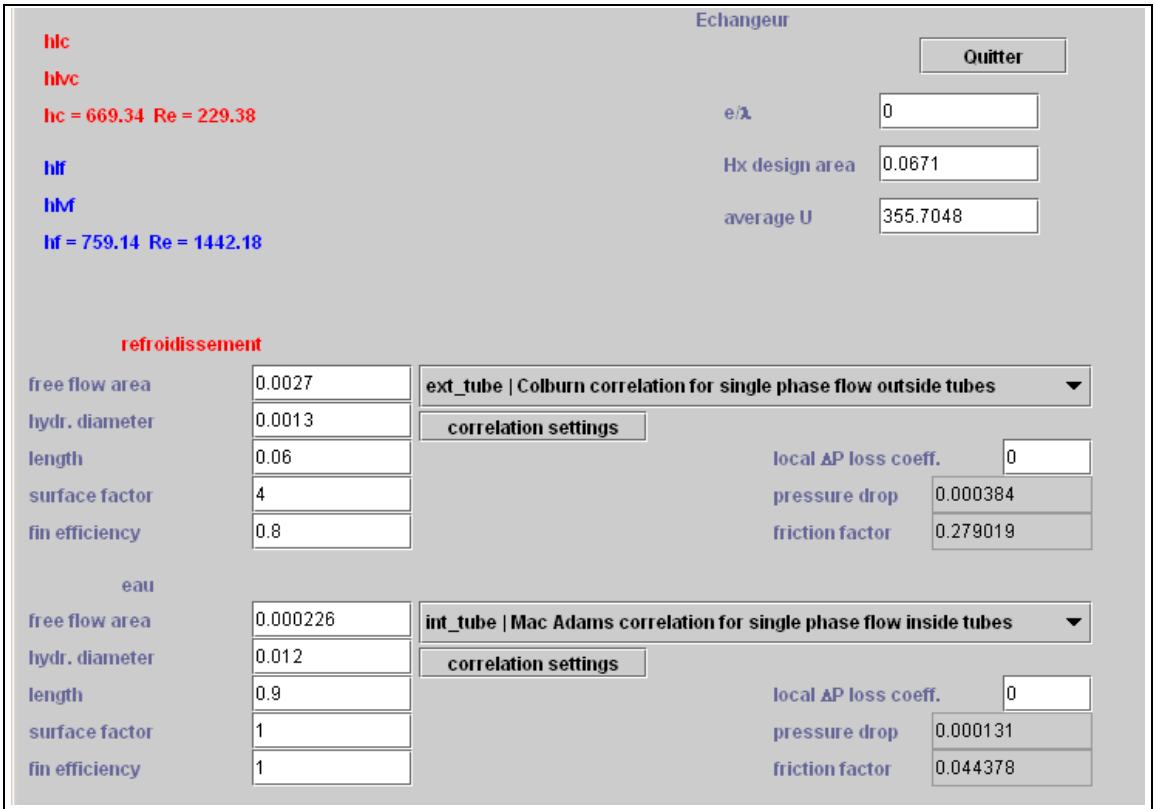

Figure 3.11 : Ecran de dimensionnement technologique

Comme nous l'avons dit dans la section précédente, il serait ici possible d'utiliser le pilote générique pour construire cet écran technologique. Toutefois, cet exemple simple se prêtant bien à une première réalisation, nous présentons ci-dessous la manière d'en créer un.

3.3.2 Conception du pilote

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

3.3.2.1 Initialisations

Pour connaître le nom de l'échangeur, on pourrait utiliser la méthode getHxList() de Projet, mais elle impose que le schéma soit chargé dans l'éditeur. Si l'on veut éviter cette contrainte, il suffit de le déclarer directement dans le code.

Pour les autres initialisations, on utilise des instances de PointThopt, qui sont des sortes de clones des points du noyau de Thermoptim, et permettent d'avoir un accès aisé à leurs valeurs. Quatre de ces objets sont instanciés, représentant les entrées et sorties des deux fluides. Ici, nous avons entré leurs noms à la main, mais il serait possible de les reconnaître automatiquement.

```
//initialisations pour la simulation
//attention : les noms des points et composants doivent être exacts, sous
hxName="Echangeur";
amontChaud=new PointThopt(proj,"entree air");
avalChaud=new PointThopt(proj,"sortie air");
amontFroid=new PointThopt(proj,"entree eau");
avalFroid=new PointThopt(proj,"sortie eau");
```

L'écran technologique est ensuite instancié et le Vector vTechno construit :

```
technoEchangeur=new TechnoHx(proj, hxName, amontChaud, avalChaud, amontFroid, a
addTechnoVector(technoEchangeur);
setupTechnoDesigns(vTechno);
```

La dernière ligne permet d'initialiser le TechnoDesign lors de l'ouverture d'un fichier de projet existant ou lors du chargement initial du pilote. Sans elle, ce dernier n'apparaît pas dans la liste des TechnoDesign de l'écran d'affichage technologique et de simulation de la figure 3.5.

3.3.2.2 Calculs

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

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

    //initialisations du TechnoDesign
    technoEchangeur.UA=UAech;
    technoEchangeur.makeDesign();

    //affichages
    U_value.setText(Util.aff_d(technoEchangeur.getU(),4));
    UAech_value.setText(Util.aff_d(UAech,4));
    AechReel=Util.lit_d(technoEchangeur.ADesign_value.getText());
    AcalculatedEch_value.setText(Util.aff_d(AechReel,4));
}
```

Les calculs sont ici très simples : en utilisant la méthode getProperties() de Projet, le pilote récupère les valeurs de UA et ΔH. L'échangeur est ensuite dimensionné par la méthode makeDesign() et les résultats sont affichés à l'écran.

3.3.2.3 Sauvegardes

Dans cet exemple, il n'y a pas de sauvegardes particulières à effectuer si l'on considère constantes les résistances thermiques d'encrassement. Les méthodes standard de sauvegarde des paramètres technologiques font l'affaire.

Deux méthodes d'ExtPilot, transferTechnoData() et setupTechnoDesigns(), permettent d'initialiser le pilote et les écrans technologiques lors de l'ouverture d'un fichier de projet existant.

La première méthode regroupe toutes les lignes du fichier de projet relatives aux écrans technologiques et les charge dans le pilote en attendant qu'il soit construit, et la seconde décode ces lignes pour effectuer ses initialisations, en passant le relais aux méthodes standard des TechnoExch, qui font quant à elles appel aux FlowConfig.

La dernière ligne de setupTechnoDesigns(), proj.bindTechnoDesigns(vSettings), sert à relier les TechnoDesign utilisés par le pilote avec les composants du noyau, afin qu'ils puissent être affichés à partir de leurs écrans et des tables générales de l'écran d'affichage technologique et de simulation.

4 NON-NOMINAL

La structure informatique développée pour pouvoir effectuer des dimensionnements technologiques permet aussi d'analyser le comportement en régime non-nominal des modèles créés avec Thermoptim.

Ce type d'analyse présente cependant des difficultés complémentaires : il faut que le modélisateur soit capable de poser le jeu d'équations décrivant les couplages existants entre les différents composants du système étudié, et qu'il trouve un algorithme de résolution approprié.

Les cas que nous avons traités nous ont montré qu'il existe deux grandes difficultés complémentaires :

- la première concerne la mise en équation des phénomènes physiques. Il faut analyser avec soin et modéliser correctement le comportement des différents composants, ce qui peut être loin d'être simple. Nous proposons dans les exemples mis en ligne sur le portail Thermoptim-Unit un certain nombre de modèles, mais ils ne sont pas les seuls possibles, tout comme nous n'avons pas traité tous les cas intéressants ;
- la seconde correspond à la résolution, généralement numérique, des équations du modèle, ce qui demande d'une part de trouver un algorithme adéquat, et d'autre part de l'initialiser correctement.

Notre propos étant principalement centré sur la thermodynamique appliquée, nous nous sommes essentiellement concentrés sur la première difficulté. Les outils de résolution numérique que nous proposons utilisent des bibliothèques soit propres à Thermoptim, pour effectuer de simples recherches par dichotomie, soit génériques, comme Minpack, qui est un ensemble de classes permettant de faire de l'optimisation non linéaire, et en particulier de chercher la solution de systèmes d'équations, qui sera présenté dans la section 4.3.

Nous commencerons dans un premier temps par traiter un exemple simple qui ne pose pas de difficulté de résolution numérique et d'expliquer le code de la classe du pilote.

4.1 Exemple

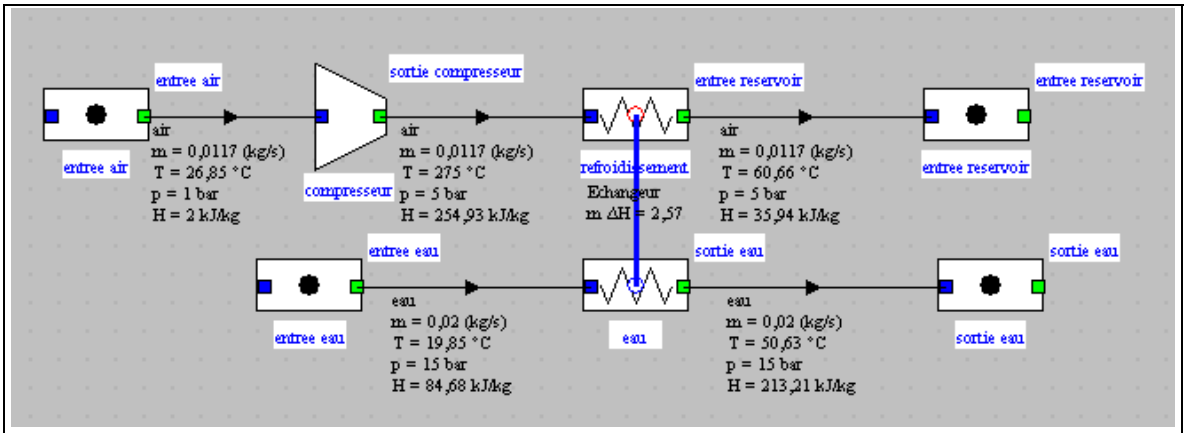

Figure 4.1 : Exemple de compresseur refroidi

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

Le système peut être facilement modélisé dans Thermoptim et conduit à un schéma du type de la figure 4.1, qui ne se distingue de celui étudié lors de l'exemple précédent que par l'ajout du compresseur.

L'écran de dimensionnement technologique du compresseur est donné figure 4.2. Celui de l'échangeur est similaire au précédent. Ils correspondent aux modèles physiques présentés section 3.1 de ce manuel.

Notez, car nous ne l'avons pas encore dit, qu'un double-clic sur le nom du composant (ici "compresseur" situé sous les petites flèches en haut à droite) permet d'accéder à son écran thermodynamique.

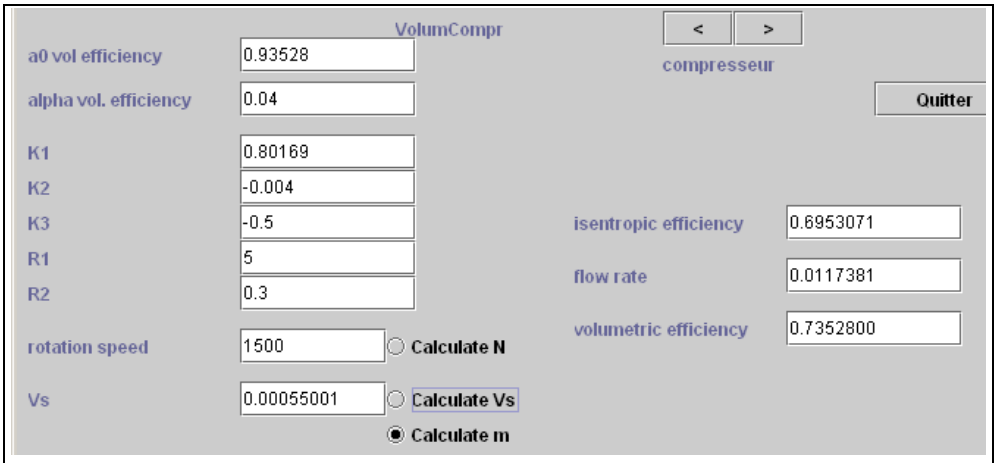

Figure 4.2 : Ecran du compresseur

4.1.1 Résultats obtenus

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

La figure 4.3 montre, en fonction de la pression du stockage, les évolutions du rendement isentropique du compresseur et du travail consommé, le débit d'air aspiré, la température de l'air entrant dans le réservoir, la charge de l'échangeur et la valeur de U.

4.1.2 Conception du pilote

La classe du pilote est PiloteCompresseurVs, et son type est "Pilote compresseur Vs". Son interface graphique étant simple et classique, nous ne la détaillerons pas ici.

L'écran du pilote est présenté figure 4.4.

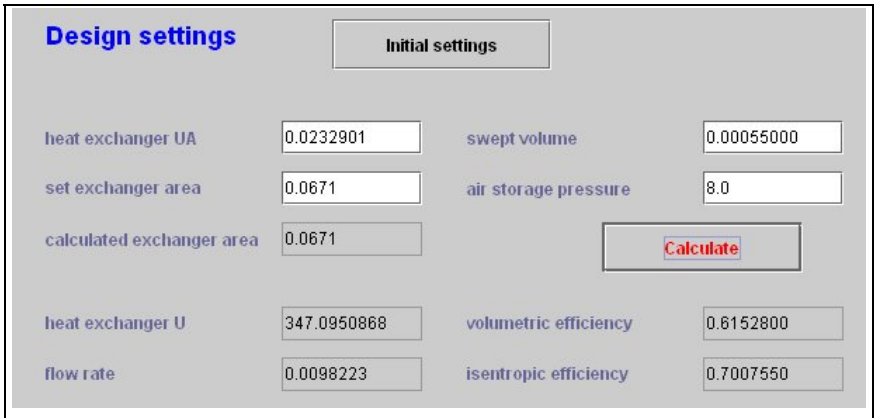

Figure 4.4 : Ecran du pilote

Pour réaliser la classe correspondante, on reprend celle du pilote de l'échangeur, et on la complète sur deux points :

- d'une part en instanciant et en initialisant l'écran technologique du compresseur, de manière analogue à celui de l'échangeur ;
- d'autre part en introduisant un bouton "Calculate", et en définissant les calculs et modifications du paramétrage du simulateur qu'il convient d'effectuer lorsqu'on fait varier la pression aval.

4.1.2.1 Initialisations

Lors de l'initialisation, après que les PointThopt et les deux TechnoDesign aient été construits, la cylindrée du compresseur permettant d'obtenir le débit souhaité est déterminée, sur la base du paramétrage de l'écran technologique.

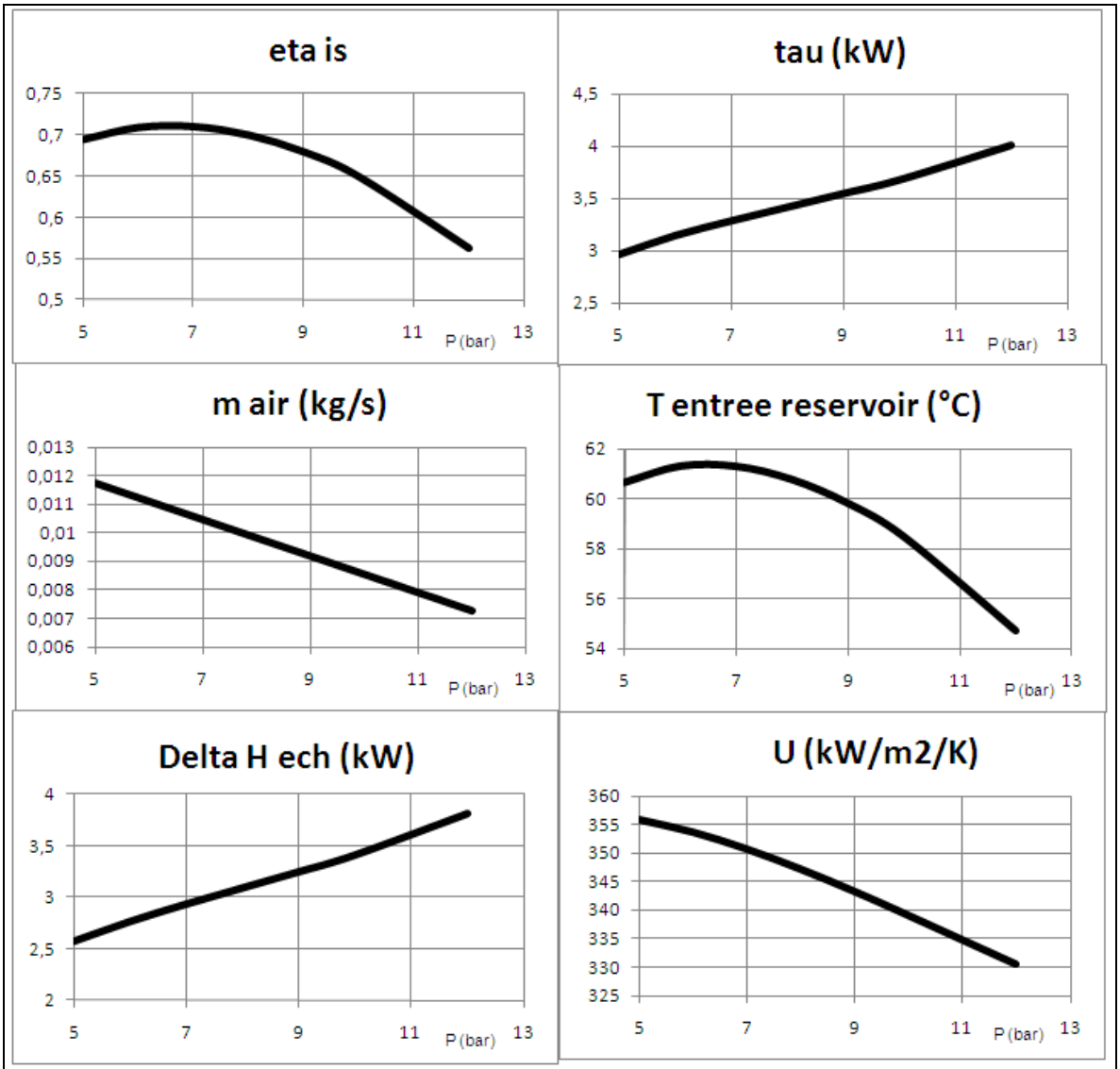

Figure 4.3 : Résultats de simulation

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

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

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

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

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

4.1.2.2 Calculs

Les calculs à effectuer sont les suivants ;

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

//mise à jour du simulateur et des affichages
for(int j=0;j<3;j++)proj.calcThopt();
flow_value.setText(Util.aff_d(massFlow,4));
eta_is_value.setText(Util.aff_d(eta_is,4));
lambdaVol_value.setText(Util.aff_d(lambdaVol,4));
AcalculatedEch_value.setText(technoEchangeur.ADesign_value.getText());
```

4.1.2.3 Sauvegardes

Les sauvegardes et mises à jour lors des lectures de fichier de projet sont analogues à celles de la classe PiloteEchangeur. La méthode setupPilot() a été complétée pour tenir compte de l'existence du compresseur.

4.2 Utilisation du modèle pour simuler le remplissage d'un stockage d'air comprimé

Les résultats obtenus peuvent être utilisés pour simuler le remplissage d'un stockage d'air comprimé. Pour cette simulation, nous développons un second modèle, résolu sous Excel, avec les équations suivantes :

- la masse M d'air contenue dans le stockage est égale à la masse initiale plus l'intégrale du débit
- l'énergie interne U est égale à l'énergie interne initiale plus l'intégrale du produit du débit par l'enthalpie de l'air sortant du refroidisseur, moins l'intégrale des pertes par convection avec l'air ambiant
- la température du stockage est déduite de son énergie interne
- la pression se calcule alors par la loi des gaz idéaux.

Pour connaître le débit, le travail de compression et la température de l'air en entrée de stockage, il suffit de faire des régressions polynomiales à partir des simulations effectuées précédemment avec Thermoptim. Les équations obtenues sont, en fonction du rapport de compression r :

$\dot{m} = - 0{,}0006\ r + 0{,}0149$

$\tau = 0{,}0019\ r^3 - 0{,}0484\ r^2 + 0{,}5322\ r + 1{,}2806$

$Tair = 0{,}01\ r^3 - 0{,}4735\ r^2 + 4{,}9142\ r + 46{,}699$

L'allure des résultats est donnée ci-dessous, pour un stockage d'un demi m3 (diamètre 80 cm, longueur 1 m), alimenté par un compresseur de cylindrée Vs = 0,525 l. La pression et la température dans le stockage, sa masse et le travail à fournir au compresseur sont affichés figure 4.5.

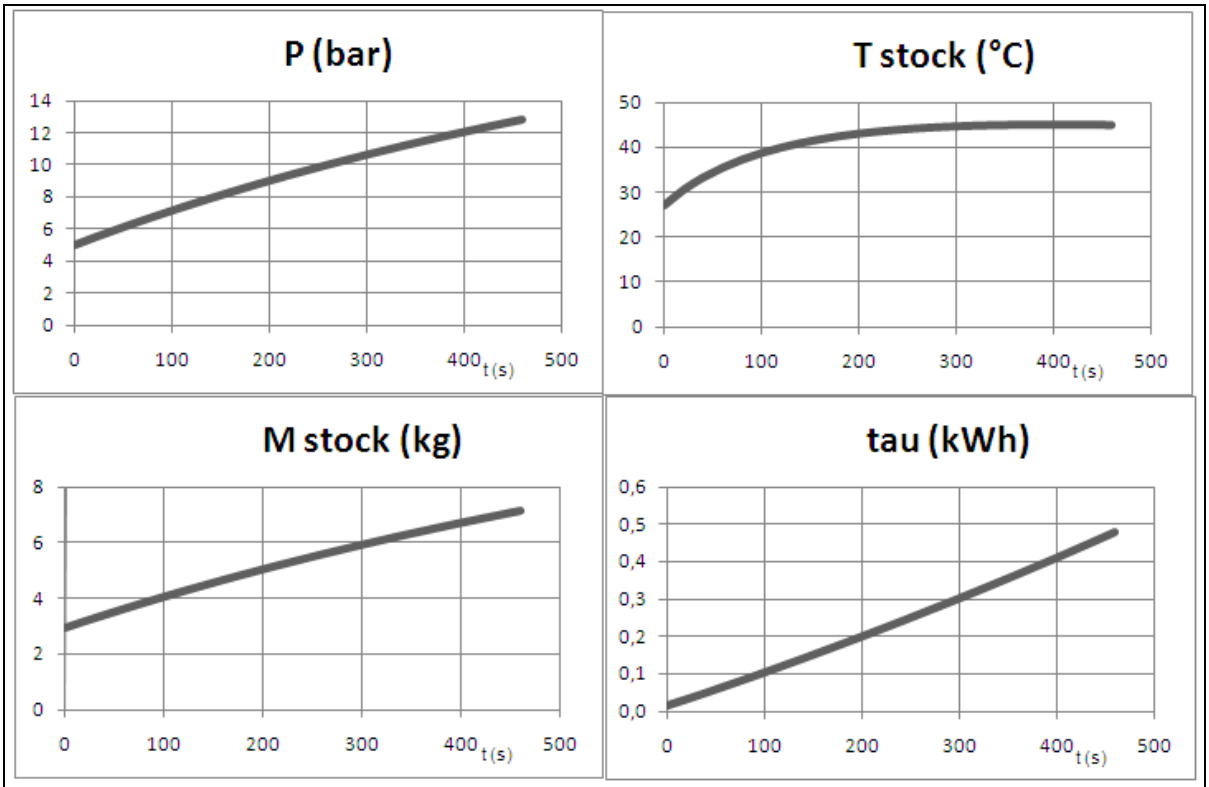

Figure 4.5 : Remplissage du stockage d'air comprimé

Cet exemple vous a permis de faire l'apprentissage de l'écriture d'un pilote destiné à simuler le comportement en régime non-nominal d'un système thermodynamique relativement simple. Pour approfondir la question, nous vous conseillons d'étudier la notice de programmation d'un pilote de machine frigorifique disponible sur le portail Thermoptim-UNIT[1], qui, compte tenu du nombre d'équations couplées qu'il met en jeu, nécessite l'utilisation d'un solveur.

4.3 Utilisation du solveur minPack pour résoudre des systèmes d'équations non-linéaires

Dans de nombreux cas, les équations décrivant le comportement non-nominal d'un système énergétique sont non-linéaires, de telle sorte que leur résolution constitue un problème difficile, surtout lorsque le nombre d'inconnues est élevé.

4.3.1 Présentation de minPack

Basé sur la méthode de Levendberg-Marquardt, minPack est un ensemble d'algorithmes mis au point en Fortran, puis 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.

Précisons que minPack est en fait un ensemble d'algorithmes de minimisation qui peut être utilisé pour trouver le minimum de la norme L2 des résidus ||F(X)|| d'un ensemble de m équations non-linéaires à n inconnues F(X).

[1] http://www.thermoptim.org/sections/logiciels/thermoptim/documentation/pilote-pour-cycle

Lorsque n est égal à m, minPack peut être utilisé pour résoudre le système d'équations F(X) = 0, ou tout au moins chercher un vecteur X qui soit la plus proche possible de la solution.

On peut aussi utiliser minPack pour identifier un jeu de n paramètres permettant d'ajuster une équation non linéaire sur un ensemble de m données.

4.3.2 Mise en œuvre de minPack

L'ensemble des classes nécessaire est inclus dans la bibliothèque extBib.zip qui doit être déclarée dans le classPath.

L'implémentation sous Java de minPack se fait en utilisant une interface appelée optimization.Lmdif_fcn, qui contraint les classes appelantes à disposer d'une fonction appelée fcn().

Cette fonction fcn() reçoit comme principaux arguments un vecteur (tableau x[n])[2] contenant les variables et un vecteur (tableau fvec[m]) renvoyant les résidus des fonctions que l'on cherche à annuler. Comme nous l'avons indiqué, leur nombre peut excéder celui des variables, mais il doit être le même si on cherche la solution d'un système d'équations.

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.

4.3.3 Exemple

Pour illustrer l'emploi de minPack, nous avons réalisé une classe de test, appelée TestMinPack, qui résout le système d'équations correspondant à l'intersection d'un cercle et d'une droite. Le code se présente comme suit :

```
int info[] = new int[2];
int m = 2;//système de deux équations
int n = 2;//à deux inconnues
int iflag[] = new int[2];
double fvec[] = new double[m+1];//vecteur des résidus
double x[] = new double[n+1];
double residu0;//norme L2 initiale
double residu1;//norme L2 finale

System.out.print("\n\n\n\n\n problem dimensions:  " + n + "  " + m + "\n");

//initialisation des inconnues
x[1] = 1;
x[2] = 0;

iflag[1] = 0;

testMinPack.fcn(m,n,x,fvec,iflag);//premier appel pour calcul norme L2 initiale
//norme L2 initiale
residu0 = optimization.Minpack_f77.enorm_f77(m,fvec);

testMinPack.nfev = 0;
testMinPack.njev = 0;

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

[2] 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

```
//appel à minPack
optimization.Minpack_f77.lmdif2_f77(testMinPack, m, n, x, fvec, epsi, epsfcn, info);

//norme L2 finale
residu1 = optimization.Minpack_f77.enorm_f77(m,fvec);

//affichage des résultats
System.out.println("\n Initial L2 norm of the residuals: " + residu0 +
      "\n Final L2 norm of the residuals: " + residu1 +
      "\n Number of function evaluations: " + testMinPack.nfev +
      "\n Number of Jacobian evaluations: " + testMinPack.njev +
      "\n Info value: " + info[1] );
System.out.println();
System.out.println("precision: \t" + residu1);
for(int i=1;i<=n;i++){//affichage de la solution
   System.out.println("x["+i+"] \t" + x[i]);
}
```

La fonction fcn est définie comme suit :

```
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++;

          fvec[1] =x[1]*x[1]+x[2]*x[2]-1;
          fvec[2] =3*x[1]+2*x[2]-1;
}
```

Les résultats que l'on obtient sont les suivants, pour l'initialisation x[1] = 1, x[2] = 0 :

```
problem dimensions:  2  2

 Initial L2 norm of the residuals: 2.0
 Final L2 norm of the residuals: 2.39665398638067E-10
 Number of function evaluations: 6
 Number of Jacobian evaluations: 10
 Info value: 2

precision:          2.39665398638067E-10
x[1]      0.7637079407212384
x[2]      -0.6455619110818576
```

pour l'initialisation x[1] = 0, x[2] = 0 :

```
problem dimensions:  2  2

 Initial L2 norm of the residuals: 1.4142135623730951
 Final L2 norm of the residuals: 3.853333208070353E-10
 Number of function evaluations: 9
 Number of Jacobian evaluations: 12
 Info value: 2

precision:          3.853333208070353E-10
x[1]      -0.3021694793631984
x[2]      0.9532542190447976
```

Cet exemple illustre d'une part comment minPack peut être utilisé, mais aussi l'importance qu'il y a à bien initialiser le problème soumis. Dans notre cas, le problème a deux solutions, mais l'algorithme se satisfait de la première qu'il trouve, la plus proche de la valeur de départ.

ANNEXE : Classes ExtPilot, PointThopt, CorpsThopt

Dans cette annexe, nous donnons quelques indications sur les trois classes ExtPilot, PointThopt et CorpsThopt qui ont été définies pour faciliter la création des classes externes avancées, et notamment des pilotes.

Classe ExtPilot

C'est la classe mère de tous les pilotes. Afin de sécuriser le paramétrage des différentes méthodes de mise à jour des objets du noyau de Thermoptim qu'elle propose, elle définit un certain nombre de booléens sous forme directement compréhensible :

boolean RECALCULATE=true, UPDATE_T=true, UPDATE_P=true, UPDATE_X=true, UPDATE_ETA=true, UPDATE_UA=true, UPDATE_EPSI=true, UPDATE_FLOW=true, IS_SET_FLOW=true, UPDATE_DTMIN=true;

De cette manière, le code suivant d'une mise à jour d'un point :

avalFroid.update(UPDATE_T,!UPDATE_P,!UPDATE_X);

se lit mettre à jour la seule température du point avalFroid, ce qui est beaucoup plus lisible que :

avalFroid.update(true, false , false);

public void **updateprocess**(String name, String type, boolean recalculate, boolean isSetFlow, boolean updateFlow, double flow,
boolean updateParam1, double param1, boolean updateParam2, double param2)

Disponible avec plusieurs signatures, cette méthode met à jour et recalcule une transfo. Si recalculate vaut true, elle est recalculée, si isSetFlow vaut true, son débit est imposé, si updateFlow vaut true, son débit est modifié et vaut flow, si updateParam vaut true, la valeur param est imposée. Le sens de param dépend du type et peut être retrouvé en se référant à la méthode updateProcess() de Projet, telle que définie dans le tome 3 manuel de référence.

Par exemple, la ligne suivante met à jour la transfo de nom compressorName, de type Compression, demande le recalcul, impose le débit en modifiant sa valeur, et modifie le rendement isentropique :

updateprocess(compressorName, "Compression",RECALCULATE,IS_SET_FLOW, UPDATE_FLOW, massFlow, UPDATE_ETA, eta_is);

De manière analogue, existent des méthodes de mise à jour et recalcul pour les échangeurs et les nœuds :

public void **updateHx**(String name, boolean recalculate, boolean updateUA, double UA,
boolean updateEpsi, double epsi, boolean updateDTmin, double DTmin)

public void **updateNode**(String name, boolean recalculate, boolean updateEfficiency, double epsi)

Les deux méthodes transferTechnoData() et setupTechnoDesigns() permettent d'initialiser le pilote et les écrans technologiques lors de l'ouverture d'un fichier de projet existant.

La première méthode regroupe toutes les lignes du fichier de projet relatives aux écrans technologiques et les charge dans le pilote en attendant qu'il soit construit, et la seconde décode ces lignes pour effectuer ses initialisations, en passant le relais aux méthodes standard des TechnoExch, qui font quant à elles appel aux FlowConfig.

La dernière ligne de setupTechnoDesigns(), proj.bindTechnoDesigns(vSettings), sert à relier les TechnoDesign utilisés par le pilote avec les composants du noyau, afin qu'ils puissent être affichés à partir de leurs écrans et des tables générales de l'écran d'affichage technologique et de simulation.

```
void setupTechnoDesigns(Vector vTechnoDesign){
    Vector vSettings=new Vector();
    if(technoDesignData.equals(""))setupTechnoDesigns();
    else{//cas où il faut relire les données
        StringTokenizer st = new StringTokenizer(technoDesignData,"\n");
        while(st.hasMoreTokens()){
            String ligne=st.nextToken();
            String type=Util.extr_value(ligne,"compType");
            String comp=Util.extr_value(ligne,"component");
            String value="";
            if(type.equals("HeatEx")){//échangeurs : 3 lignes dans ce cas
                value=Util.extr_value(ligne,"ADesign_value");
                for(int j=0;j<vTechnoDesign.size();j++){
                    TechnoHx etd;
                    Object[]obj1=new Object[5];
                    obj1=(Object[])vTechno.elementAt(j);
                    etd=(TechnoHx)obj1[1];
                    String etdType=(String)obj1[4];
                    String etdComp=(String)obj1[0];
                    if((comp.equals(etdComp))&&(type.equals(etdType))){
                        if(!(value==null)){
                            etd.ADesign_value.setText(value);
                        }
                        ligne=st.nextToken();
                        etd.techc.readCompParameters(ligne);
                        ligne=st.nextToken();
                        etd.techf.readCompParameters(ligne);
                        Object[]obj=new Object[3];
                        obj[0]=comp;
                        obj[1]=type;
                        obj[2]=etd;
                        vSettings.addElement(obj);
                        break;
                    }
                }
            }
        else{//autres TechnoDesign
            for(int j=0;j<vTechnoDesign.size();j++){
                ExtTechnoDesign etd;
                Object[]obj1=new Object[5];
                obj1=(Object[])vTechno.elementAt(j);
                etd=(ExtTechnoDesign)obj1[1];
                String etdType=(String)obj1[4];
                String etdComp=(String)obj1[0];
                if((comp.equals(etdComp))&&(type.equals(etdType))){
                    etd.readCompParameters(ligne);
                    Object[]obj=new Object[3];
                    obj[0]=comp;
                    obj[1]=type;
                    obj[2]=etd;
                    vSettings.addElement(obj);
                    break;
                }
            }
        }
    }
}
proj.bindTechnoDesigns(vSettings);
```

Classe PointThopt

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 manuel de référence. Elle possède des double dans lesquels sont stockés les équivalents des variables d'état du noyau :

```
double W,Epsi,QPrime,Tprime,Tr,VPrime,Cond, M_sec, corrFactor;
 CorpsThopt corps;
```

Lors de la construction, on spécifie la référence au projet et le nom du point dans le simulateur, qui permet à Thermoptim de savoir quel point est concerné. Attention, s'il y a une erreur à ce niveau, le PointThopt ne peut être correctement instancié.

```
public PointThopt(Projet proj, String name){
            this.proj=proj;
            this.name=name;
            try{
               getProperties();
            }
            catch(Exception e){
               String msg = "Error constructing the PointCorps name: "+ name;
               JOptionPane.showMessageDialog(new JFrame(), msg);
               e.printStackTrace();
            }
            corps=new CorpsThopt(proj, nomCorps,lecorps);
}
```

PointThopt propose lui aussi des méthodes de mise à jour de son équivalent du simulateur, fonctionnant sur le même principe que celles de ExtPilot, avec des signatures variables selon le nombre de valeurs à modifier.

```
   public void update(boolean updateT, boolean updateP, boolean updateX, boolean updateCorrFactor, boolean
melHum, String task, double value)
```

Une version allégée la plus courante a servi d'exemple à l'utilisation des booléens explicites d'ExtPilot :

```
   public void update(boolean updateT, boolean updateP, boolean updateX){
      update(updateT, updateP, updateX, false, false, "", 0.);
   }
```

Classe CorpsThopt

Cette classe permet de créer dans une classe externe des sortes de clones des corps 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 updateSubstance() de Projet, documentées dans le tome 3 manuel de référence. Elle possède des double dans lesquels sont stockés les équivalents des variables d'état du noyau :

```
public double T,P,X,V,U,H,S,M,M_sec,TC,PC,VC;
```

Lors de la construction, on spécifie la référence au projet et le nom du corps dans le simulateur, qui permet à Thermoptim de savoir quel corps est concerné. Attention, s'il y a une erreur à ce niveau, le CorpsThopt ne peut être correctement instancié.

```
         public CorpsThopt(Projet proj, String name, Corps lecorps){
            this.proj=proj;
            this.name=name;
            this.lecorps=lecorps;
```

```
}
```

Un exemple d'instanciation est donné dans le constructeur de PointThopt.

La méthode **getSubstProperties**() permet de mettre à jour les valeurs après un calcul effectué sur la variable lecorps. L'exemple ci-dessous montre l'utilisation d'un CorpsThopt faisant appel au PointThopt "amont" :

```
CorpsThopt corps=new CorpsThopt(proj,amont.nomCorps, amont.lecorps);
corps.lecorps.CalcPropCorps(amont.T,amont.P,1.);
corps.getSubstProperties();
```

Les fonctions usuelles sont alors directement accessibles : par exemple, l'enthalpie vaut corps.H.

CLASSES DE DIMENSIONNEMENT TECHNOLOGIQUE

Les classes relatives aux compresseurs volumétriques et aux échangeurs ayant été présentées section 3.2, nous ne montrerons ici que les autres. Pour introduire de nouvelles classes, il suffit soit de sous-classer celles qui existent, soit d'en créer sur le même modèle et de les placer dans extThopt.zip ou extUser.zip.

Classes représentatives des turbines

La modélisation en non-nominal des turbines est un problème qui peut être abordé à des niveaux de difficulté variables. C'est pourquoi nous avons implémenté une série de modèles différents, permettant de graduer la finesse d'approche en fonction d'une part des données disponibles et d'autre part du niveau de l'utilisateur :

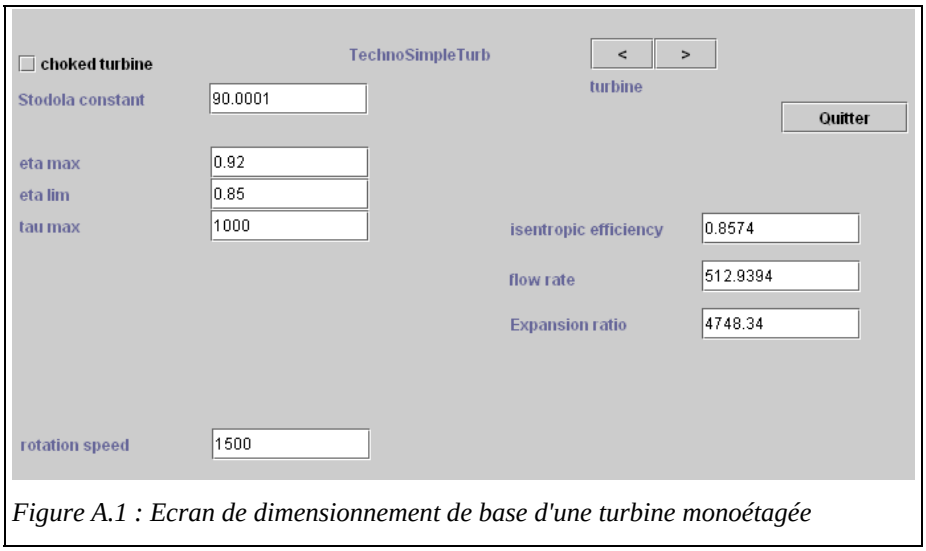

Figure A.1 : Ecran de dimensionnement de base d'une turbine monoétagée

- la classe de base s'appelle TechnoSimpleTurb. Elle est basée sur la règle du cône et ne s'applique en toute rigueur qu'à une turbine mono-étagée ;
- la classe TechnoMultiStageTurb étend la précédente en considérant que la règle du cône de la turbine multi-étagée est la même que celle d'une turbine mono-étagée, le rapport de détente à prendre en compte étant la racine nième du rapport de détente total, n étant le nombre d'étages. Si n=1, le comportement est le même que la classe précédente ;
- la classe TechnoTurb étend la précédente, en permettant de prendre en compte d'une part l'existence de pertes par vitesse résiduelle (PVR) en sortie de turbine, et d'autre part une détérioration du rendement isentropique en zone humide, selon la règle de Baumann ;
- la classe MultiStageMappedTurbine étend la précédente, la règle du cône étant cette fois-ci remplacée par une cartographie représentée par un ajustement numérique à une vingtaine de paramètres.

TechnoSimpleTurb

Cette classe définit les paramètres de dimensionnement technologique pour les turbines (figureA.1).

$$\frac{\dot{m}\sqrt{T_{in}}}{P_{in}} = Cste \qquad (A.1)$$

$$Tin\frac{\dot{m}^2}{K^2} = P_{in}^2 - P_{out}^2 \qquad (A.2)$$

La signification des champs est la suivante :

- Stodola constant est la valeur du coefficient de Stodola. Si l'option "choked turbine" est cochée, le débit est considéré comme choqué, et la formule retenue est (A.1). Sinon, c'est (A.2) ;
- lorsque η est représenté par une équation à 3 paramètres, eta max est la valeur maximale, eta lim la limite pour les taux de détente très élevés, et tau max l'abscisse du maximum. Si les deux premières valeurs sont égales, le rendement isentropique est constant ;
- rotation speed est la vitesse de rotation ;

TechnoMultiStageTurb

Son écran se distingue du précédent par l'apparition d'un nouveau champ, représentant le nombre d'étages de la turbine (figure A2).

La formule de Stodola prise en compte est plus générale que (A2), et prend en compte le coefficient polytropique de la détente.

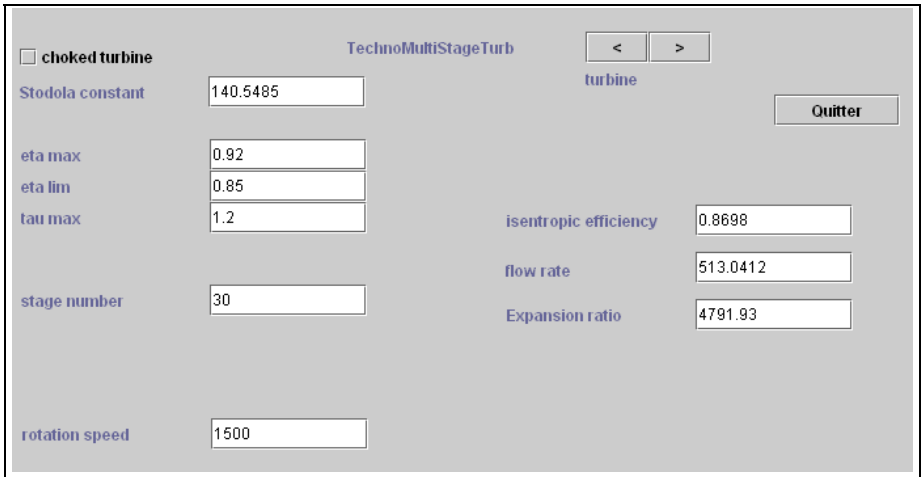

Figure A.2 : Ecran de dimensionnement de base d'une turbine multiétagée

TechnoTurb

Son écran (figure A3) se distingue du précédent par l'apparition de plusieurs nouveaux champs, permettant comme indiqué plus haut de prendre en compte d'une part l'existence de pertes par vitesse résiduelle (PVR) en sortie de turbine, et d'autre part une détérioration du rendement isentropique en zone humide, selon la règle de Baumann (A3).

Les valeurs des pertes par vitesse résiduelle apparaissent sur la gauche de l'écran.

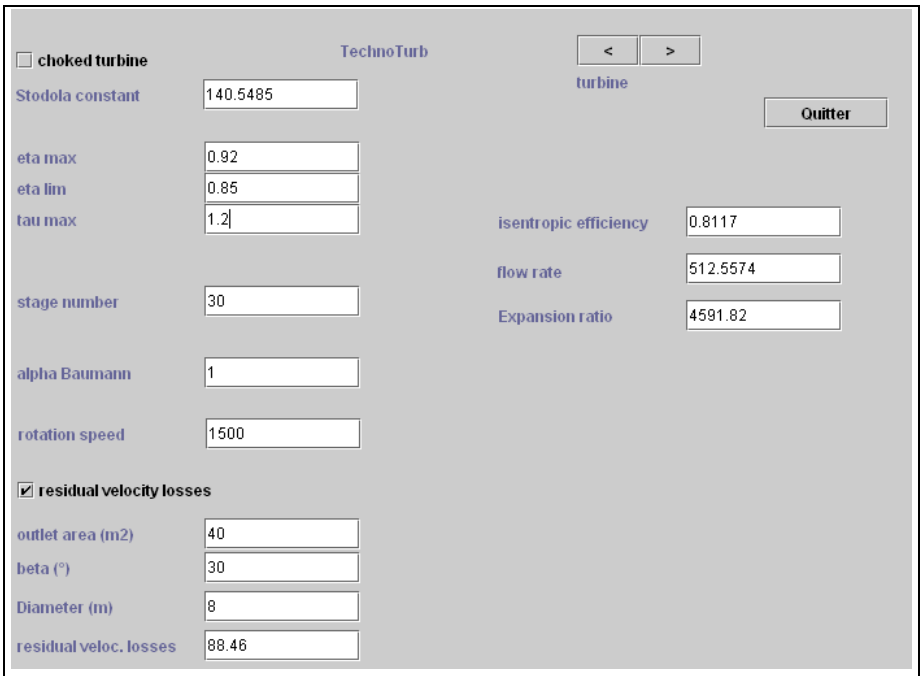

Figure A.3 : Ecran de dimensionnement d'une turbine multiétagée

$$\eta_{hum} = \eta_{sec} (1 - \alpha (1 - x)) \qquad (A.3)$$

La signification des champs est la suivante :

- alpha Baumann est la valeur du coefficient α de l'équation (A.3) ;
- pour le calcul des pertes par vitesse résiduelle, il faut de plus fournir la section de sortie et le diamètre de la turbine, ainsi que l'angle β de sortie.

Classe MultiStageMappedTurbine

Cette classe, qui hérite de TechnoTurb, définit les paramètres de dimensionnement technologique pour les turbines représentées par une cartographie détaillée, telle que définie dans une note de modélisation[3].

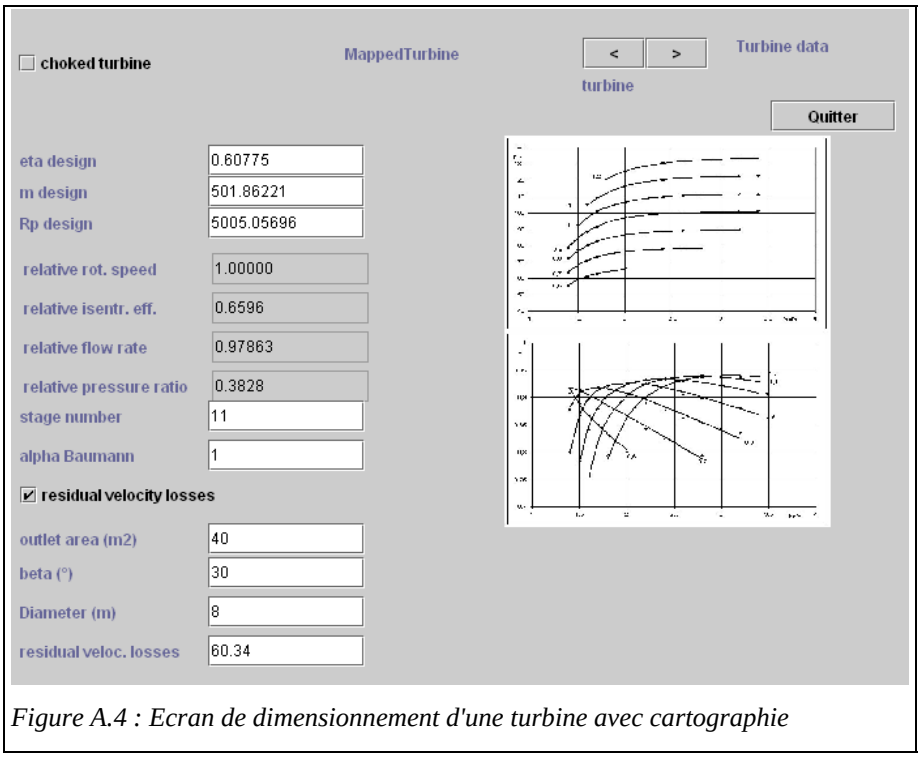

Figure A.4 : Ecran de dimensionnement d'une turbine avec cartographie

L'écran technologique (figureA.4) possède dans sa partie supérieure droite un libellé ("Turbine data") qui donne accès par double-clic à l'écran permettant d'afficher la série de coefficients décrivant les caractéristiques de débit et de rendement isentropique (classe TurbineMapDataFrame, figureA.5).

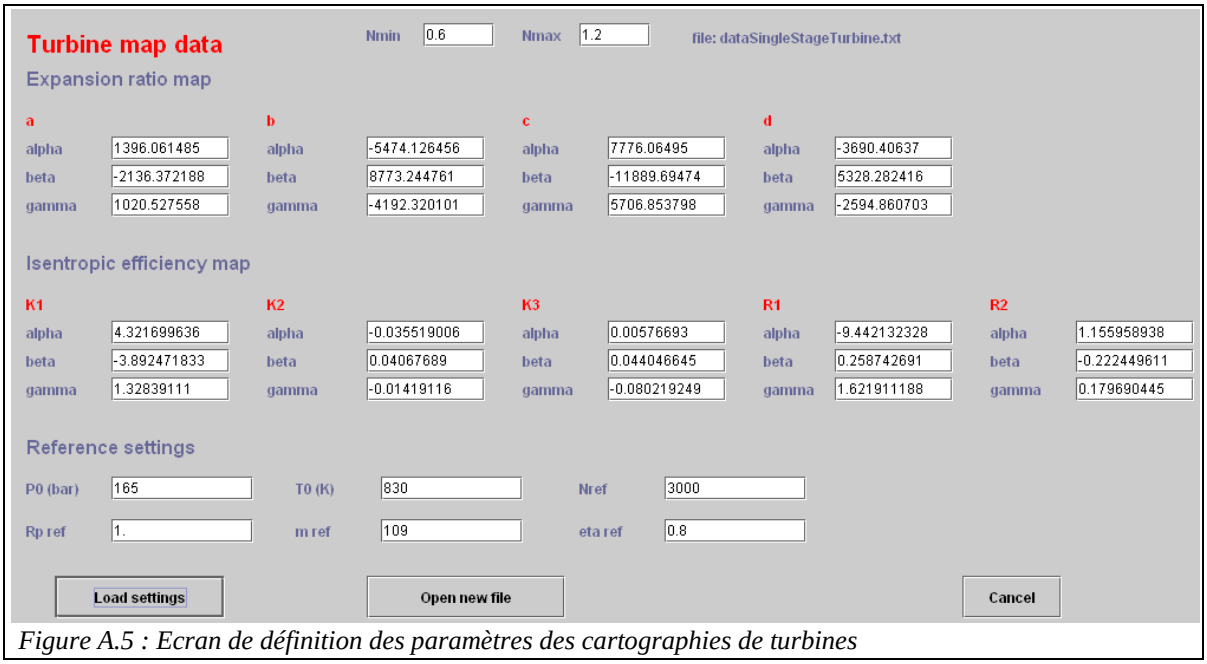

Figure A.5 : Ecran de définition des paramètres des cartographies de turbines

A droite de la figureA.4, apparaît en médaillon une image correspondant aux caractéristiques retenues, si elle a été chargée, et sinon le libellé "Turbine maps" qui permet par double-clic d'ouvrir un écran de chargement d'image (classe MapImage).

Les fichiers de données et d'images des cartographies doivent être placés dans le répertoire "maps" à partir duquel travaillent les deux classes. Leurs noms doivent être sauvegardés par le pilote.

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

La cartographie utilisée est ici exprimée par rapport au débit normé. Lors du dimensionnement, on cale le point nominal de la cartographie sur celui de la transfo détente (dans un deuxième temps, on pourra en déduire les dimensions, mais ce n'est pas encore effectué), ce qui permet d'obtenir les valeurs "eta design", "Rp design" et "m design" affichées à gauche de l'écran.

Au dessous d'elles apparaissent les valeurs adimensionnelles permettant de situer le point de fonctionnement sur la cartographie utilisée. Leurs valeurs sont bien évidemment corrigées en fonction des conditions d'admission dans la turbine.

La valeur de la vitesse de rotation peut être modifiée depuis l'écran du pilote si on le souhaite.

Classe TechnoNozzle

Cette classe définit les paramètres de dimensionnement technologique pour les tuyères (figureA.6).

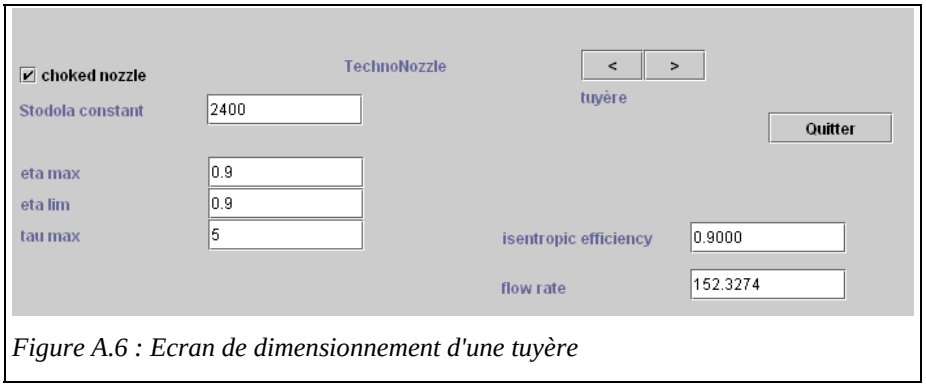

Figure A.6 : Ecran de dimensionnement d'une tuyère

En régime non-nominal, une tuyère peut être modélisée de manière analogue à une turbine, et caractérisée par une constante de Stodola et un rendement isentropique. Son écran correspond donc à la partie supérieure de celui d'une turbine mono-étagée.

Classe TechnoTurboCompr

Cette classe définit les paramètres de dimensionnement technologique pour les turbocompresseurs.

L'écran technologique (figure A.7) fait appel à une série de coefficients décrivant les deux caractéristiques.

débit corrigé réduit dans le compresseur $$mc = \frac{\dot{m}\, T_1}{K_c\, N\, P_1} \qquad (A.4)$$

caract. réduite du compresseur $$\psi = \psi_{coeff}\,(mc - \varphi\psi_{max})^2 + \psi_{max} \qquad (A.5)$$

Les paramètres à définir sont les suivants :

- Kc, caractéristique du compresseur (équation (A.4)) ;
- le diamètre du compresseur ;
- ψ est représenté par une branche de parabole, définie (équation (A.5)) par ψ_{max}, valeur maximale, ψ_{coeff}, et $\varphi\psi_{max}$, abscisse du maximum du débit corrigé mc ;

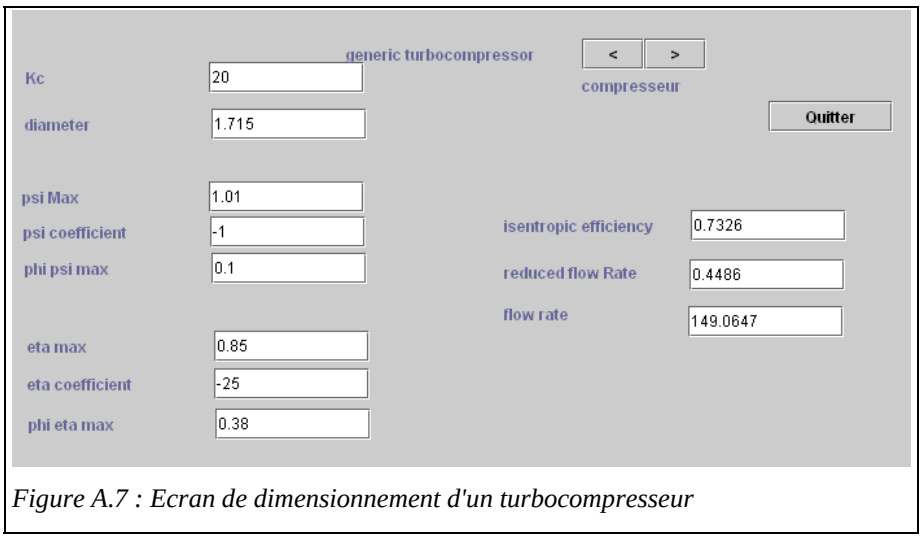

Figure A.7 : Ecran de dimensionnement d'un turbocompresseur

- η est aussi représenté par une branche de parabole, définie par η_{max}, valeur maximale, η_{coeff}, et $\varphi\eta_{max}$, abscisse du maximum du débit corrigé mc.

Les valeurs du rendement isentropique, du débit réduit et du débit massique apparaissent sur la droite de l'écran.

Classe MultiStageMappedTurboCompr

Cette classe, qui hérite de TechnoTurboCompr, définit les paramètres de dimensionnement technologique pour les turbocompresseurs représentés par une cartographie détaillée, telle que définie dans la note de modélisation citée précédemment.

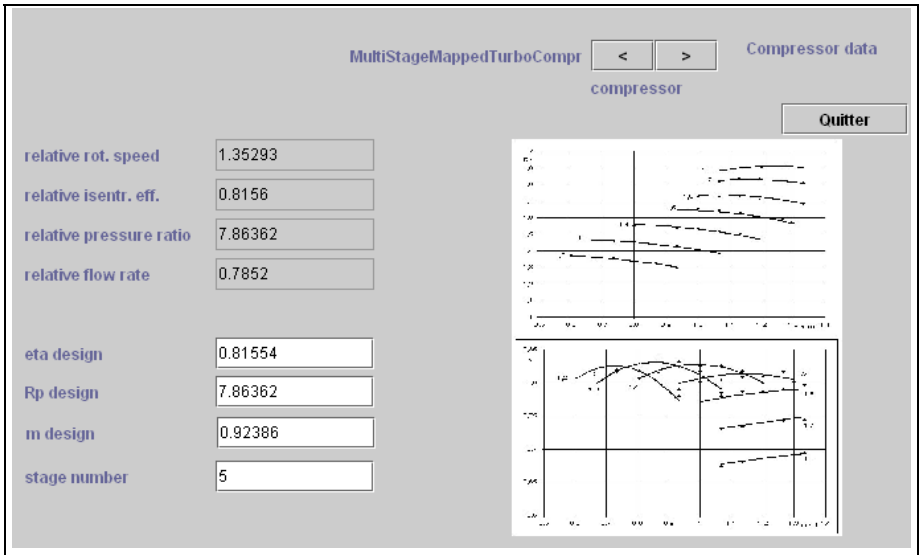

Figure A.8 : Ecran de dimensionnement d'un turbocompresseur avec cartographie

L'écran technologique (figureA.8) possède dans sa partie supérieure droite un libellé ("Compressor data") qui donne accès par double-clic à l'écran permettant d'afficher la série de coefficients décrivant les caractéristiques de débit et de rendement isentropique (classe TurboComprMapDataFrame, figureA.9).

A droite de la figureA.9, apparaît en médaillon une image correspondant aux caractéristiques retenues, si elle a été chargée, et sinon le libellé "Compressor maps" qui permet par double-clic d'ouvrir un écran de chargement d'image (classe MapImage).

Les fichiers de données et d'images des cartographies doivent être placés dans le répertoire "maps" à partir duquel travaillent les deux classes. Leurs noms doivent être sauvegardés par le pilote.

La cartographie utilisée est ici exprimée par rapport au débit normé. Il faut entrer dans le fichier de données l'ensemble des valeurs de référence : P_0, T_0, N_{ref}, Rp_{ref}, m_{ref}, eta_{is_ref}, ainsi que l'intervalle de variation de la vitesse réduite (ici 1,2 - 1,8).

Lors du dimensionnement, on cale le point nominal de la cartographie sur les conditions de la transfo compression (dans un deuxième temps, on pourra en déduire les dimensions, mais ce n'est pas encore effectué), ce qui permet d'obtenir la vitesse de rotation du compresseur et les valeurs "eta design", "Rp design" et "m design" affichées à gauche de l'écran, et éventuellement modifiables.

Au dessus d'elles apparaissent les valeurs adimensionnelles permettant de situer le point de fonctionnement sur la cartographie utilisée. Leurs valeurs sont bien évidemment corrigées en fonction des conditions d'admission dans le compresseur.

En dessous de ces valeurs apparaît le champ permettant de définir le nombre d'étages si on utilise une cartographie générique d'étage (ici 5).

La valeur de la vitesse de rotation du compresseur peut être modifiée depuis l'écran du pilote si on le souhaite.

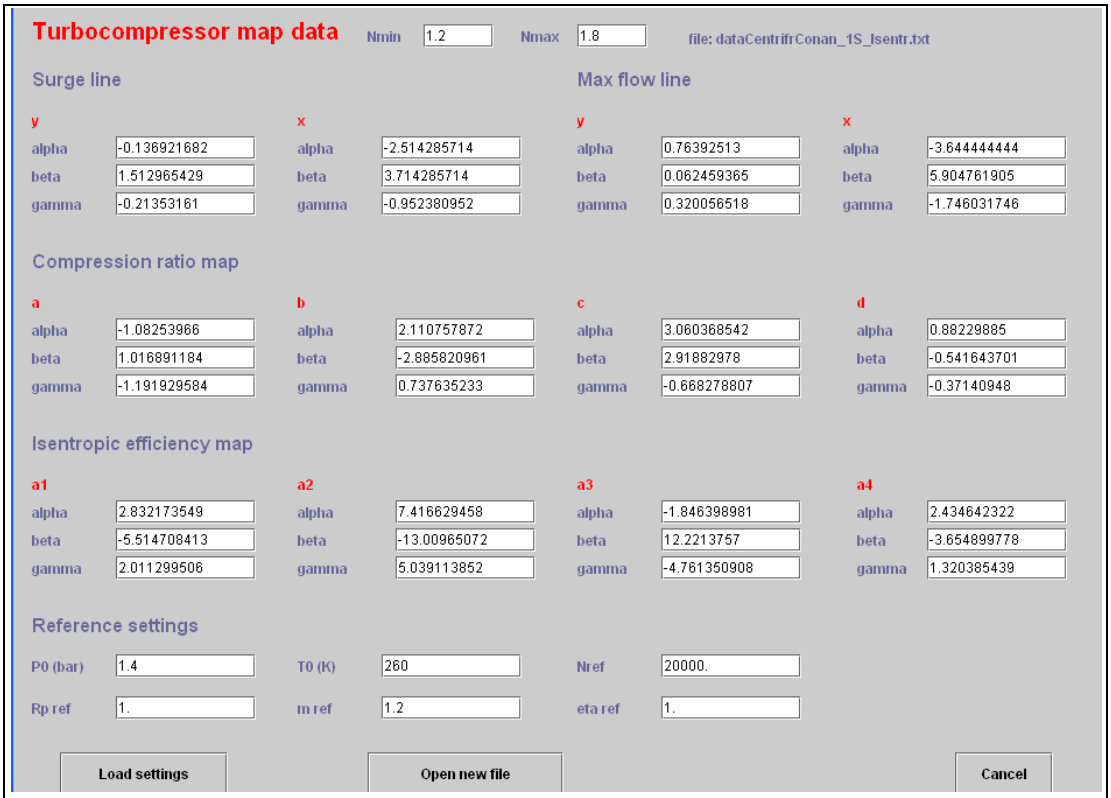

Figure A.9 : Ecran de définition des paramètres des cartographies de turbocompresseur

Remarques sur les valeurs de référence des fichiers de cartographie

Les dernières lignes des fichiers de cartographie (bas d'écran des figures A.5 et A.9) fournissent les valeurs de référence qui permettent d'adapter une cartographie à un problème donné. Les pression et température d'entrée et la vitesse de rotation correspondent généralement à celles du simulateur pour le point nominal. Le rapport de compression/détente et le rendement isentropique/polytropique devraient être égaux à 1 si la cartographie correspond bien au compresseur utilisé, mais ils peuvent être modifiés si nécessaire. La valeur du débit de référence sera généralement proche de celle du débit du compresseur au point nominal, étant donné que les cartographies ont été établies pour des débits normés.