T H E R M O P T I M ®

NON NOMINAL

PILOTE POUR GENERATEUR DE VAPEUR DE REACTEUR A EAU PRESSURISEE

RESOLUTION PAR MINPACK 1

VERSION JAVA 2.72 ET 2.82

SOMMAIRE

1 Introduction - pré-requis

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

2 Principe de calcul du générateur de vapeur en régime non-nominal

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

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

La méthode du NUT peut être présentée comme suit :

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

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

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

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

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

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

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

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

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

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

sortie par (3) et les équations de bilan.

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

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

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

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

3 Choix et paramétrage de l'écran technologique

Supposons que l'on veuille dimensionner un générateur de vapeur de réacteur à eau pressurisée (figure 1) pour qu'il fournisse une puissance thermique de 1 000 MW environ.

Les transfos du circuit primaire sont placées dans la partie haute du schéma, et celles du circuit secondaire en-dessous.

Un débit d'eau pressurisée à 155 bar et 325 °C entre dans le GV, et en ressort à 294 °C environ.

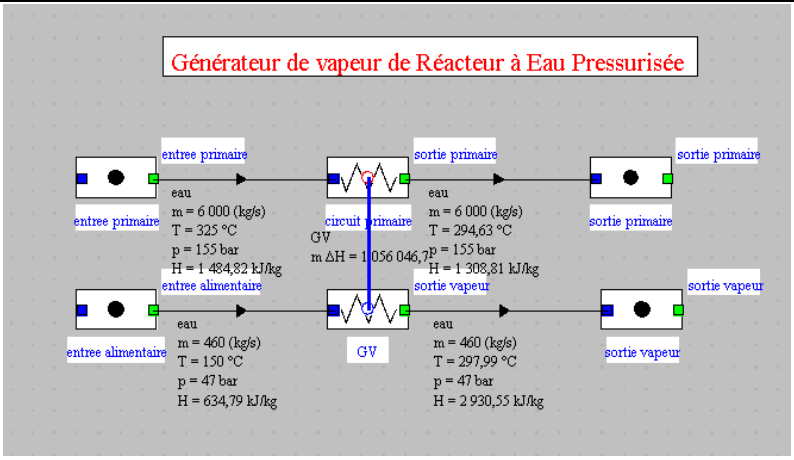

Figure 1 : Modèle de générateur de vapeur

L'eau alimentaire du circuit secondaire entre dans le GV à 47 bar et 150 °C, et en sort sous forme de vapeur surchauffée à 298 °C.

Les pertes de charge sont ici négligées, bien qu'elles soient calculées.

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

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

3.1 Création de l'écran technologique

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

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

```
        //initialisations des PointThopt et des TechnoDesign
        //attention : les noms des points et composants doivent être exacts, sous
peine de générer une erreur empêchant le chargement des TechnoDesign
        amontChaudiere=new PointThopt(proj,"entree alimentaire");
        avalChaudiere=new PointThopt(proj,"sortie vapeur");
        entreePrimaire=new PointThopt(proj,"entree primaire");
        sortiePrimaire=new PointThopt(proj,"sortie primaire");
        fluideThermo=(rg.corps.Corps)avalChaudiere.lecorps;


        //noms des composants
          boilerName="GV";
```

Le TechnoDesign est du type TechnoEvaporator, classe spécifiquement créée pour ce type de composant. Il permet de calculer le générateur de vapeur comme un échangeur multizones, selon les équations données en annexe 1.

```
          //instanciation des TechnoDesign dans les classes externes
            technoChaudiere=new TechnoEvaporator(proj, boilerName,
entreePrimaire, sortiePrimaire, amontChaudiere, avalChaudiere);
          addTechnoVector(technoChaudiere);

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

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

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

3.2 Ecran du pilote

L'écran du pilote est donné figure 2.

Il permet de modifier d'une part la surface de l'échangeur, et d'autre part la température et le débit du circuit primaire, ainsi que la pression et le débit du circuit secondaire. Il possède en outre des options de guidage de l'algorithme qui seront précisées plus loin.

Commencez par cliquer sur "Dimensionnement technologique" pour instancier le TechnoDesign et effectuer un premier dimensionnement technologique, en l'occurrence calculer la surface du GV correspondant au paramétrage du fichier de projet, sur la base des valeurs par défaut des paramètres du TechnoDesign.

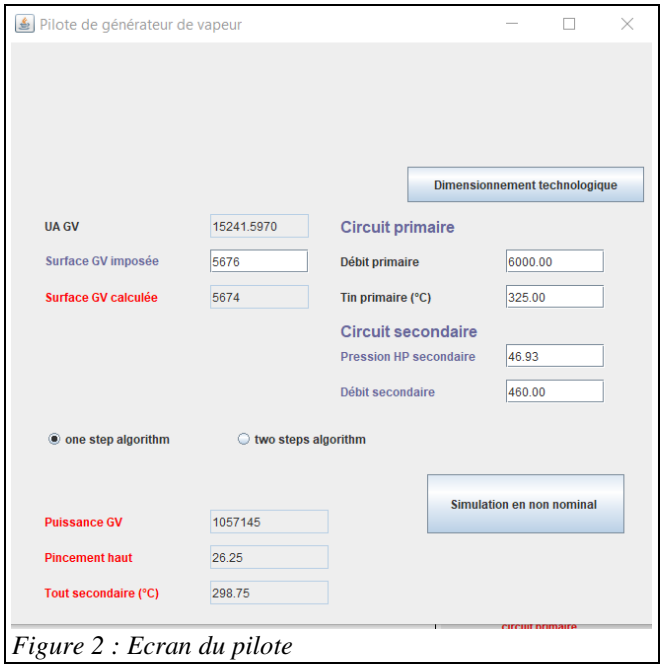

Figure 2 : Ecran du pilote

3.3 Paramétrage de l'écran technologique

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

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

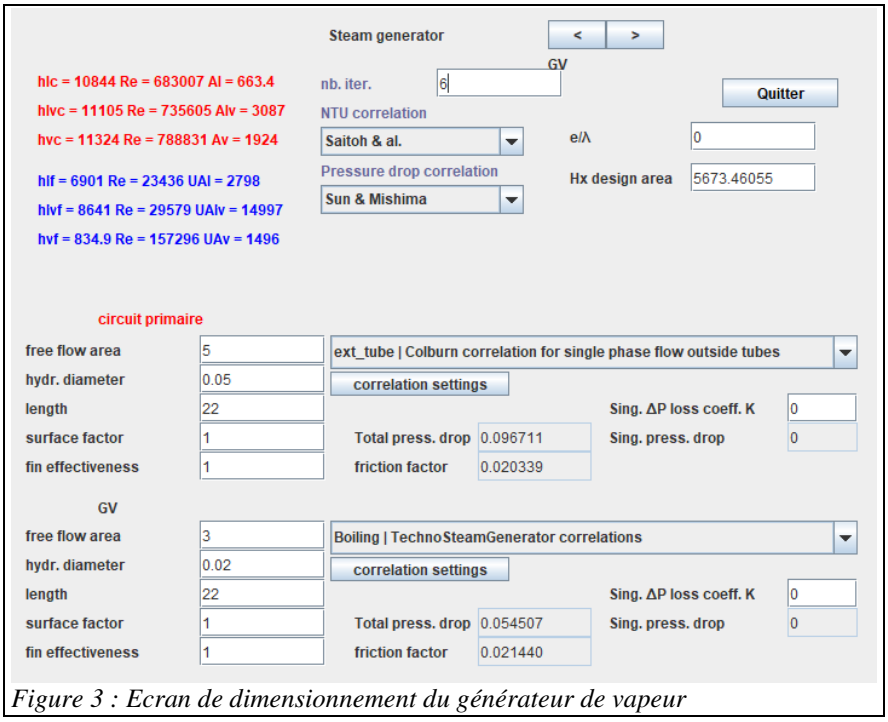

Figure 3 : Ecran de dimensionnement du générateur de vapeur

On considère que le générateur de vapeur a côté circuit secondaire une section de passage du fluide de 3 m^2 et un diamètre hydraulique de 2 cm avec une longueur de tubes de 22 m, et côté circuit primaire une section de passage de 5 m^2 et un diamètre hydraulique de 5 cm. Choisissez le type de configuration ("ext tube Colburn correlation" pour le circuit primaire, et « Boiling | TechnoSteam... » pour l'évaporation à l'intérieur des tubes pour la partie "GV" (la vapeur), ce qui vous permet de choisir les corrélations dans la partie supérieur : « Saitoh & al. » pour le calcul du NUT, et « Sun & Mishima » pour celui des pertes de charge), et paramétrez l'écran comme indiqué figure 3.

Pour réaliser le dimensionnement une fois l'écran technologique renseigné, cliquez de nouveau sur le bouton " Dimensionnement technologique " de l'écran du pilote (figure 2).

Les résultats sont affichés dans les écrans technologiques : surface d'échange de 5 340 m^2 pour le GV.

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

4 Initialisations de dimensionnement au point nominal

Le paramétrage de l'écran technologique permet de déterminer la surface d'échange, à partir des initialisations correspondant au point nominal, qui servent à fixer un certain nombre de valeurs à partir de celles du projet Thermoptim.

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

Expliquons l'initialisation du générateur de vapeur.

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

```
if(!boilerName.equals("")){//initialisation du générateur de vapeur
  args=new String[2];
  args[0]="heatEx";
  args[1]=boilerName;
  vProp=proj.getProperties(args);
  Double f=(Double)vProp.elementAt(15);
  UAevap=f.doubleValue();
  String fluideChaud=(String)vProp.elementAt(0);
  args[0]="process";
  args[1]=fluideChaud;
  vProp=proj.getProperties(args);
  f=(Double)vProp.elementAt(4);
  double DeltaH=-f.doubleValue();
  f=(Double)vProp.elementAt(3);
  primaryFlow=f.doubleValue();
  primaryFlow_value.setText(Util.aff_d(primaryFlow,2));
  Tinprimaire_value.setText(Util.aff_d(entreePrimaire.T-273.15,2));

  amontChaudiere.getProperties();
  avalChaudiere.getProperties();
  DTsurch=avalChaudiere.DTsat;
  entreePrimaire.getProperties();
  sortiePrimaire.getProperties();
    Tf=entreePrimaire.T;

  mCpCalopChaudiere=DeltaH/(entreePrimaire.T-sortiePrimaire.T);
  mCpRefrigChaudiere=DeltaH/(avalChaudiere.T-amontChaudiere.T);
  DeltaHevap=DeltaH;
```

Les méthodes getProperties() des PointThopt fournissent directement l'état thermodynamique complet des points amont et aval du GV, ce qui permet d'initialiser la surchauffe.

```
  amontChaudiere.getProperties();
  avalChaudiere.getProperties();
  DTsurch=avalChaudiere.DTsat;
```

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

```
  entreePrimaire.getProperties();
  sortiePrimaire.getProperties();
    Tf=entreePrimaire.T;
```

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

```
mCpCalopChaudiere=DeltaH/(entreePrimaire.T-sortiePrimaire.T);
 mCpRefrigChaudiere=DeltaH/(avalChaudiere.T-amontChaudiere.T);
 DeltaHevap=DeltaH;
 UAevap_value.setText(Util.aff_d(UAevap,4));
```

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

//initialisations du TechnoDesign

```
//initialisations du TechnoDesign
technoChaudiere.makeDesign();
AevapReel=Util.lit_d(technoChaudiere.ADesign_value.getText());
Uevap=technoChaudiere.UA/AevapReel;
AcalculatedChaudiere_value.setText(Util.aff_d(technoChaudiere.A,0));
```

La valeur du pincement à l'intérieur du GV est aussi un paramètre important à suivre :

```
pinch_value.setText(Util.aff_d(technoChaudiere.DTmin,2));
```

5 Résolution du système d'équations du générateur de vapeur en non-nominal

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

Dans cet exemple, nous retiendrons la température de retour primaire, la variable liée étant la puissance thermique mise en jeu.

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

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

5.1 Méthode de résolution

Pour résoudre le problème, on opère en deux temps : on commence par effectuer le dimensionnement technologique pour le régime nominal, puis on cherche en régime non-nominal le paramétrage permettant d'obtenir le même dimensionnement technologique pour des conditions d'entrée et des sollicitations différentes de celles du régime nominal.

Une des difficultés du problème est en effet qu'il faut concilier deux modes de calcul :

- celui des écrans classiques (phénoménologiques) de Thermoptim, qui ne connaissent pas les paramètres de dimensionnement technologique et effectuent leurs calculs sans en tenir compte
- celui des TechnoDesign, qui prennent en compte les paramètres de dimensionnement technologique.

En pratique, cela signifie que le pilote effectue les calculs en régime non-nominal et détermine comment les écrans phénoménologiques doivent être reparamétrés pour que l'ensemble reste cohérent.

Dans le cas précis de notre GV, le pilote cherche une solution telle que le dimensionnement effectué pour un état autre que le régime nominal conduise à une surface de même valeur que celle retenue par l'utilisateur. Il fait varier la température de retour du primaire jusqu'à obtenir la bonne surface, les valeurs de U dépendant des débits et des températures, et celle de UA de l'équilibrage de l'échangeur.

Au niveau numérique, nous utilisons la méthode de Levendberg-Marquardt correspondant aux algorithmes mis au point en Fortran sous le nom de minPack 1, et traduits sous Java. Cette méthode combine l'algorithme de

Gauss-Newton et la méthode de descente de gradient. Son intérêt principal est d'être très robuste et de ne nécessiter comme initialisation qu'une solution approchée.

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

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

Le guidage de l'algorithme se fait en pratique en jouant sur deux critères de précision, l'un portant sur la valeur d'un résidu, et l'autre sur la précision du calcul des dérivées partielles, estimées par différences finies. Dans ce cas simple, nous cherchons à résoudre une équation non linéaire à une inconnue, ce qui peut se révéler numériquement difficile. A l'usage, il est apparu intéressant de proposer plusieurs options de calcul.

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

Selon les cas, l'utilisateur pourra opter pour l'une ou l'autre, en fonction des difficultés numériques rencontrées. Un indicateur de précision, correspondant à la norme L2 des résidus, est affiché dans la partie résultat de l'écran du pilote (figure 4).

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

5.1.1 Vecteur des variables

Le vecteur des variables est ici le suivant : `x[1] = sortiePrimaire.T;`

5.1.2 Initialisations et appel à l'algorithme de résolution

Les initialisations se font sur la base des valeurs affichées dans l'écran du pilote, pour la surface d'échange du générateur de vapeur, la température et le débit du circuit primaire, ainsi que la pression et le débit du circuit secondaire. Les autres valeurs sont celles des écrans du simulateur.

Afin de faciliter la convergence de l'algorithme, on met en place une rampe entre les valeurs initiales et celles que l'on cherche à atteindre.

```
            //mise en place d'une rampe de variation des valeurs

            avalChaudiere.getProperties();
          entreePrimaire.getProperties();
            double AdesignChaudiereOld=AdesignChaudiere;
            double PevapOld=avalChaudiere.P;
            double massFlowOld=massFlow;
            double entreePrimaireTOld=entreePrimaire.T;
            double primaryFlowOld=primaryFlow;
            UAevap=Util.lit_d(UAevap_value.getText());
            double
AdesignChaudiereTarget=Util.lit_d(AdesignChaudiere_value.getText());
            double PevapTarget=Util.lit_d(Pevap_value.getText());
            double massFlowTarget=Util.lit_d(massFlow_value.getText());
```

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

```
            double
entreePrimaireTTarget=Util.lit_d(Tinprimaire_value.getText())+273.15;
            double primaryFlowTarget=Util.lit_d(primaryFlow_value.getText());
            double nRampe=20;
```

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

```
                    int m = 1;
                    int n = 1;

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

                    x[1] = sortiePrimaire.T;

                    double residu0;
                    double residu1;

                    fcn(m,n,x,fvec,iflag);
                    residu0 = optimization.Minpack_f77.enorm_f77(m,fvec);

                    nfev2 = 0;
                    njev2 = 0;

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

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

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

5.1.3 Fonction fcn()

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

Comme indiqué dans fcn(), la fonction fvec est une méthodes resAevap(), définie ci-dessous.

```
      public void fcn(int m, int n, double x[], double fvec[], int iflag[]) {
            if (iflag[1]==1) this.nfev++;
            if (iflag[1]==2) this.njev++;

            //mise à jour des variables "physiques" pour une meilleure
compréhension du code
            sortiePrimaire.T=x[1];
            sortiePrimaire.update(UPDATE_T,!UPDATE_P,!UPDATE_X);
            sortiePrimaire.getProperties();

        Vector vProp;
        Double f;
```

La première étape consiste à mettre à jour tous les points et transfos du modèle pour que les calculs du résidu soient effectués sur la base des valeurs du vecteur x. Ici, on cherche à déterminer la température de retour du circuit primaire, dont la connaissance permet de calculer la puissance thermique du GV, et donc l'état de la vapeur à sa sortie.

```
        DeltaHevap= (entreePrimaire.H-sortiePrimaire.H)*primaryFlow;
         avalChaudiere.H=amontChaudiere.H+DeltaHevap/massFlow;

avalChaudiere.T=avalChaudiere.corps.lecorps.getT_from_hP(avalChaudiere.H,avalChaud
iere.P);
        // pour bien faire, il faudrait calculer X, ce qui peut se faire assez
facilement
        //et afficher un message si X<=1
        avalChaudiere.update(UPDATE_T,!UPDATE_P,UPDATE_X);
        avalChaudiere.getProperties();

        mCpRefrigChaudiere=DeltaHevap/(avalChaudiere.T-amontChaudiere.T);
        mCpCalopChaudiere=DeltaHevap/(entreePrimaire.T-sortiePrimaire.T);
```

La seconde étape consiste à calculer l'échangeur par la méthode du NUT, ce qui fournit UAevap.

```
        double R=mCpRefrigChaudiere/mCpCalopChaudiere;
        if(mCpRefrigChaudiere<mCpCalopChaudiere) {
             mCpmin=mCpRefrigChaudiere;
        }
        else {
             mCpmin=mCpCalopChaudiere;
             R=mCpCalopChaudiere/mCpRefrigChaudiere;
        }
        epsilon=(avalChaudiere.T-amontChaudiere.T)/(entreePrimaire.T-
amontChaudiere.T);

        double NUT=Util.NUT_epsi(epsilon,R);
        UAevap=NUT*mCpmin;

        if(debug)System.out.println("\nDeltaHevap: " + DeltaHevap
                    +"  mCpRefrigChaudiere: " + mCpRefrigChaudiere
                    +"  mCpCalopChaudiere: " + mCpCalopChaudiere);
        if(debug)System.out.println("UAevap: " + UAevap+"  epsilon: " + epsilon+"
mCpmin: " + mCpmin
                    +"  NUT: " + NUT+"  R: " + R
                    );
```

La troisième étape consiste à recalculer le simulateur pour que tous ses écrans soient mis à jour. On le fait plusieurs fois pour que les calculs se stabilisent. On recalcule alors le TechnoDesign, puis on estime le résidu.

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

        amontChaudiere.getProperties();
        avalChaudiere.getProperties();
```

```
        entreePrimaire.getProperties();
        sortiePrimaire.getProperties();

          technoChaudiere.makeDesign();
          AcalculatedChaudiere_value.setText(Util.aff_d(technoChaudiere.A,0));

          //calcul du résidu
            fvec[1] = resAevap();
}
```

5.1.4 Fonction de résidu

Dans ce cas, le résidu a été normé pour équilibrer les poids des différentes fonctions d'écart, en utilisant une méthode simple, mais valable uniquement si la valeur à atteindre n'est pas nulle, ce qui est toujours le cas ici.

```
       double resAevap(){//renvoie le résidu de la surface de l'évaporateur
        UAevap_value.setText(Util.aff_d(UAevap,4));
          AevapReel=technoChaudiere.A;
          AcalculatedChaudiere_value.setText(Util.aff_d(AevapReel,0));
          double z= AevapReel-AdesignChaudiere;
          z= 2*z/(AevapReel+AdesignChaudiere);
          if(debug)System.out.println("AevapReel: " + AevapReel + ",
AdesignChaudiere "+   AdesignChaudiere);
          if(debug)System.out.println("resAevap: " + z + ", UAevap "+  UAevap);
          if(debug)System.out.println();

        return z;

    }
```

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

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

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

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

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

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

                        System.out.println();
```

```
                    System.out.println(" Initial L2 norm of the residuals: " +
residu0);
                System.out.println("Final L2 norm of the residuals: " +
residu1);
                System.out.println("Number of function evaluations: " + nfev);
                System.out.println("Number of Jacobian evaluations: " + njev);
                System.out.println("Info value: " + info[1]);
                System.out.println("Final approximate solution: " + x[1]  );
                     System.out.println(); /**/
                  algorithmResults.setText("Algorithm precision: " +
Util.aff_d(residu1,8));
                 }
            eff_value.setText(Util.aff_d(-tauTurb/DeltaHevap,4));
            DeltaHevap_value.setText(Util.aff_d(DeltaHevap,0));
            massFlow_value.setText(Util.aff_d(massFlow,4));
            Pevap_value.setText(Util.aff_d(Pevap,4));
            pinch_value.setText(Util.aff_d(technoChaudiere.DTmin,2));
            Tmax_value.setText(Util.aff_d(avalChaudiere.T-273.15,2));
        }
```

6 Utilisation du pilote

6.1 Simulation en non-nominal

L'écran du pilote est donné figure 4.

Si cela n'a pas été fait, commencez par cliquer sur "Dimensionnement technologique" pour initialiser le pilote.

Entrez ensuite les valeurs que vous souhaitez modifier (ici la température du circuit primaire, qui passe à 315 °C au lieu de 325), en faisant attention à ne pas vous écarter trop rapidement du point de départ, car des difficultés de convergence peuvent apparaître et vous obliger à arrêter Thermoptim. Incrémentez donc peu à peu les variables que vous souhaitez modifier, et effectuez des sauvegardes intermédiaires dont vous pourrez repartir en cas de problème.

Une fois les nouvelles valeurs saisies, soit cliquez sur "Simulation en non nominal", soit ouvrez l'écran des TechnoDesign depuis le simulateur, puis cliquez sur "Calculer le pilote". Cette deuxième manière de faire est préférable, car elle permet de suivre la convergence tout en gardant la main sur Thermoptim, pour afficher des valeurs intermédiaires ou modifier les calculs du pilote.

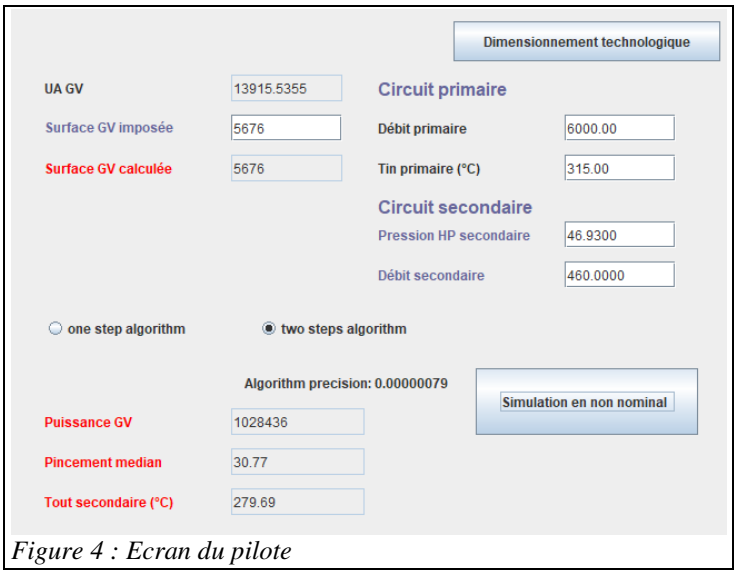

Figure 4 : Ecran du pilote

Les résultats sont affichés à l'écran une fois la convergence obtenue. Vous pouvez utiliser la macro de post-traitement[4] de Thermoptim pour exploiter les fichiers de projet obtenus comme résultats.

La valeur du pincement interne dans le GV est affichée. L'écran phénoménologique de

[4] https://direns.mines-paristech.fr/Sites/Thopt/fr/co/macro-excel-post.html

l'échangeur ne voit pas ce pincement, alors que le TechnoDesign choisi dans le pilote (un TechnoEvaporator) effectue ses calculs en distinguant les trois zones internes et tient donc compte du pincement.

Surveillez sa valeur. Si elle devient inférieure à 5 °C, vous risquez fort de rencontrer des problèmes numériques qui traduisent le fait que le GV ne peut plus fonctionner en pratique.

Il faut pour la même raison bien avoir à l'esprit que la valeur du DTML indiquée dans l'écran de l'échangeur est trompeuse du fait que le GV est représenté par un seul échangeur. Le DTML apparent reste à peu près inchangée autour de 70 °C, alors que le pincement passe de 26,25 à 30,7 °C.

La puissance du GV a baissé de 1,057 MW à 1,028 MW.

6.2 Limites du modèle

Ce modèle ne comporte qu'un seul composant, le générateur de vapeur. Dans un cycle réel, il y en aurait bien sûr d'autres, en particulier une turbine HP traversée par un débit qui varie en fonction de ses conditions d'admission, par exemple par une loi du type Stodola. Le débit du circuit secondaire varierait donc au lieu d'être imposé comme dans ce pilote.

Par ailleurs on a utilisé leTechnoEvaporator disponible, mais il serait possible de modéliser les échanges dans le GV par un TechnoDesign construit sur mesure. De la même manière, les corrélations choisies sont celles disponibles dans les classes externes distribuées avec Thermoptim, mais il serait plus précis d'en sélectionner une spécifique.

Annexe 1 : Principe de calcul du GV comme échangeur multizones

Générateur de vapeur : fluide froid diphasique et fluide chaud en sensible

Les équations sont les suivantes (figure 5) :

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

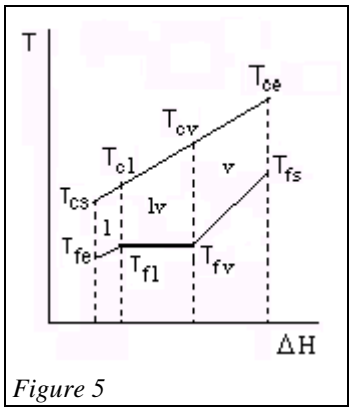

Figure 5

Les relations donnant epsilon et R sont alors :

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

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

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

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

Le pincement DTmin est ici égal à Tcl – Tfl.