Généralités sur la modélisation et la simulation dynamique des systèmes

Ce chapitre donne quelques notions fondamentales permettant de comprendre la modélisation mise en œuvre dans le travail de thèse, sans toutefois rentrer dans des considérations théoriques trop précises. On peut trouver pour cela des ouvrages comme ceux de Fritzson (2011, 2015) [1], [2].

4.1. La notion de système

Un système est un ensemble qui comprend des éléments et des relations entre ces éléments. Dans le domaine des sciences pour l'ingénieur, on parle généralement de système physique puisqu'il s'agit d'étudier les propriétés physiques d'une pièce, d'une machine, d'un réseau, d'une centrale, ou même d'un élément ou phénomène naturel. Un système est caractérisé par des variables, qui selon le cas, peuvent être des entrées ou des sorties. Lorsque le système est décrit par des équations, les variables peuvent être des entrées ou des sorties et on parle alors de relations d'acausalité.

Dans ce travail de thèse, les systèmes dont nous traitons sont ceux se rapportant aux centrales DSG linéaires, qu'ils soient de grandes tailles et complexes (le champ solaire, le circuit vapeur), ou simplement à l'échelle du composant (collecteur cylindro-parabolique, tube absorbeur, ballon séparateur, vanne de réglage, etc.).

4.2. La notion de modèle

Un modèle est une représentation d'un système sur laquelle on peut réaliser des expériences, afin d'obtenir des informations sur le système en question. On ne peut définir un modèle que par rapport au système auquel il se rapporte et ne le qualifier que par rapport à une expérience précise. L'expérience menée sur un modèle est appelée communément une simulation. L'intérêt d'un modèle est justement de pouvoir mener des simulations sans toucher au système réel, car cela est trop complexe, trop couteux, ou trop dangereux. Cela permet également de penser, de dimensionner, de designer un système n'existant pas encore.

Dans les travaux dont il est question ici, les modèles utilisés sont de type mathématique, c'est-à-dire qu'ils décrivent le système par des équations, des fonctions mathématiques, et des algorithmes. On peut tout d'abord classifier les modèles en trois catégories, selon le niveau de connaissance du système modélisé :

- Le modèle physique (« white box » model) : Les relations internes du système décrit sont entièrement explicitées par des équations représentant les phénomènes physiques ;
- Le modèle empirique (« black box » model) : Le comportement du système est décrit par des relations (dites empiriques) issues d'expériences réalisées sur le système, sans que le comportement et les relations internes ne soient connus. Typiquement, un modèle empirique consiste en des relations algébriques entre les entrées et les sorties du système ;
- Le modèle semi-empirique (« grey-box » model) : Certaines relations du système sont décrites de façon explicite grâce aux équations de la physique, tandis que d'autres sont approchées par des relations empiriques, soit à cause de leur méconnaissance, soit dans un esprit de simplification ;

Ensuite, il me parait important de définir la notion de modélisation dynamique, puisque c'est l'objet d'une grande partie de ce travail de thèse. On distingue les modèles statiques des modèles dynamiques :

- Les modèles « dynamiques » prennent en compte le temps en tant que variable dans les équations du système, et celles-ci peuvent contenir des termes de dérivées en temps (dans ce cas, l'état du système à l'instant considéré dépend entre autres de l'état à l'instant précédent) ;
- Les modèles sont dits « statiques » si les équations ne comportement pas de termes utilisant la variable temps. Ces modèles, aussi appelés « stationnaires », ne peuvent donc pas représenter les états transitoires d'un système, mais uniquement des situations stationnaires décrivant un équilibre.

Pour visualiser la différence, on peut prendre l'exemple simple d'un écoulement de vapeur surchauffée à travers un volume de contrôle. En imposant le débit d'entrée et en le faisant varier à la manière d'un créneau entre deux valeurs, on observe l'évolution du débit de sortie :

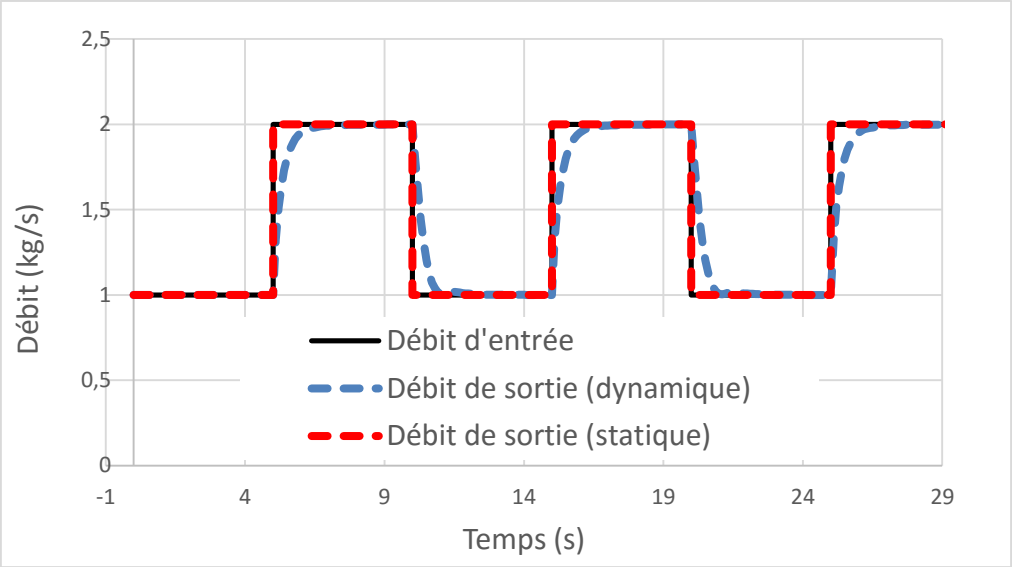

Figure 4-1 : Réponse du débit de sortie d'un écoulement de vapeur à travers un volume à une variation périodique du débit d'entrée. Simulation Modelica/Dymola en utilisant la librairie ThermoSysPro

On voit qu'avec une modélisation statique, le système est à l'équilibre en tout temps et le débit d'entrée est toujours égal au débit de sortie. En revanche, avec une modélisation dynamique, l'accumulation de masse dans le volume due à la légère variation de pression et de densité est prise en compte, et on voit que le débit de sortie subit une phase transitoire avant d'atteindre son état stationnaire. La simulation dynamique décrit donc les états transitoires du système, tandis que la simulation statique n'est qu'en fait la simulation d'états équilibrés successifs.

4.3. Le langage Modelica

Le langage Modelica est un langage orienté objet de modélisation par équations. C'est un langage libre développé et maintenu par l'Association Modelica, une association internationale dont le but non lucratif est de promouvoir et de développer l'utilisation de ce langage.

On parle de modélisation « orientée objet » puisque chaque modèle élémentaire (on parle « d'objet ») appartient à un certain type de modèle (on parle alors de « classe »). Les modèles élémentaires d'un même type ont les mêmes propriétés (entrées, sorties, équations), mais les paramètres peuvent être différents. Du point de vue de la sémantique, on parle alors de différentes instances d'une même classe.

Pour une description approfondie des propriétés du langage Modelica, on pourra consulter les références [1], [3]. On peut toutefois mentionner ici les propriétés les plus importantes, afin de comprendre les grands principes de la modélisation avec ce langage :

- La section d'**initialisation** : dans chaque modèle programmé en langage Modelica, outre la section comportant les équations décrivant le système physique, une section initialisation comporte les équations représentant l'état initial du système. C'est également le cas pour les variables : le langage prend en charge leur initialisation ;
- Les sections **équations** et **algorithmes** : dans la première de ces sections, les équations décrivent le système physique de manière acausale, et dans la deuxième, des actions de calcul sont exécutées avec une causalité ;
- Les **connecteurs** : l'intérêt de la modélisation orientée objet proposée par Modelica est de pouvoir construire des modèles complexes multi-physiques avec des modèles élémentaires. Les connecteurs sont ce qui permet aux modèles élémentaires d'échanger des informations entre eux. Ce sont typiquement des ports échangeant des valeurs de grandeurs physiques comme la pression, l'enthalpie, le débit, etc. ;
- Les **événements** : le langage Modelica gère les événements temporels, c'est-à-dire qu'il peut être introduit dans un modèle un morceau de code spécifique, valable uniquement lorsque le temps de simulation atteint une certaine valeur ;
- Les **annotations** : il s'agit de morceaux additionnels de code, qui ne concernent pas la modélisation physique du système, mais qui permettent d'apporter des informations supplémentaires, par exemple pour donner un aspect graphique au modèle. Les plateformes de modélisation comme Dymola permettent en effet de composer des modèles en assemblant de façon graphique les modèles élémentaires.

Du point de vue des dimensions, on parle généralement de modélisation « 0D-1D » puisque les modèles n'ont en général pas de dimension dans l'espace. Cela veut dire que les grandeurs n'ont qu'une seule valeur représentant l'ensemble de l'espace dans le sous-système modélisé, grandeur qui va toutefois varier dans le temps. On peut néanmoins créer des modèles disposant d'une dimension dans l'espace, comme par exemple les tubes de la librairie ThermoSysPro (abordés en détail plus loin) qui ont une dimension axiale et qui sont discrétisés.

La figure suivante montre le code Modelica d'un modèle de perte de charge singulière (vanne, coude, etc.) :

```
model SingularPressureLoss "Singular pressure loss"
  parameter Real K=1.e3 "Pressure loss coefficient";
  parameter Boolean continuous_flow_reversal=false
    "true: continuous flow reversal - false: discontinuous flow reversal";
  parameter Integer fluid=1 "1: water/steam - 2: C3H3F5";
  parameter Modelica.SIunits.Density p_rho=0 "If > 0, fixed fluid density";
  parameter Integer mode=0
    "IF97 region. 1:liquid - 2:steam - 4:saturation line - 0:automatic";
protected
  constant Real pi=Modelica.Constants.pi "pi";
  parameter Real eps=1.e-3 "Small number for pressure loss equation";
  parameter Modelica.SIunits.MassFlowRate Qeps=1.e-3
    "Small mass flow for continuous flow reversal";
public
  ThermoSysPro.Units.DifferentialPressure deltaP "Singular pressure loss";
  Modelica.SIunits.MassFlowRate Q(start=100) "Mass flow rate";
  Modelica.SIunits.Density rho(start=998) "Fluid density";
  Modelica.SIunits.Temperature T(start=290) "Fluid temperature";
  Modelica.SIunits.AbsolutePressure Pm(start=1.e5) "Average fluid pressure";
  Modelica.SIunits.SpecificEnthalpy h(start=100000) "Fluid specific enthalpy";

equation
  C1.P - C2.P = deltaP;
  C2.Q = C1.Q;
  C2.h = C1.h;
  h = C1.h;
  Q = C1.Q;
  /* Flow reversal */
  if continuous_flow_reversal then
    0 = noEvent(if (Q > Qeps) then C1.h - C1.h_vol else if (Q < -Qeps) then
      C2.h - C2.h_vol else C1.h - 0.5*((C1.h_vol - C2.h_vol)*Modelica.Math.sin(pi
      *Q/2/Qeps) + C1.h_vol + C2.h_vol));
  else
    0 = if (Q > 0) then C1.h - C1.h_vol else C2.h - C2.h_vol;
  end if;
  /* Pressure loss */
  deltaP = K*ThermoSysPro.Functions.ThermoSquare(Q, eps)/rho;
  /* Fluid thermodynamic properties */
  Pm = (C1.P + C2.P)/2;
  pro = ThermoSysPro.Properties.Fluid.Ph(Pm, h, mode, fluid);
  T = pro.T;
  if (p_rho > 0) then
    rho = p_rho;
  else
    rho = pro.d;
  end if;

end SingularPressureLoss;
```

Figure 4-2: Structure du code d'un modèle Modelica : perte de charge singulière (ThermoSysPro)

On peut voir les différentes sections de déclaration des variables utilisées dans le code. Certaines sont dans une section « public », ce sont celles qui seront visualisables après calcul. Les variables privées, utilisées dans les équations, mais non visualisables, sont déclarées dans la section « protected ». Les paramètres modifiables via l'interface graphique du modèle sont déclarés hors de ces sections, avec le préfixe « parameter ».

4.4. Plateforme de modélisation et de simulation : le logiciel Dymola

Le logiciel commercial Dymola a été choisi comme support du travail de modélisation et de simulation pour les travaux de thèse présentés dans ce manuscrit. Dymola (pour DYnamic MOdeling LAboratory) est développé et maintenu par l'éditeur de logiciels Dassault Systèmes. Il est distribué commercialement avec CATIA, le logiciel plus connu du même éditeur. Dymola est à la fois un support de modélisation (composition graphique d'un modèle, puis traduction en code Modelica), de simulation (traitement préparatoire du modèle mathématique, exécution du solveur), et de visualisation des résultats.

4.4.1. Construction du modèle global

L'aspect graphique donné au modèle par les sections « annotations » dans le code trouve un intérêt avec l'utilisation d'une plateforme de modélisation qui permet de construire graphiquement un modèle en

assemblant différents sous-modèles. La figure suivante montre le modèle graphique, réalisé avec Dymola, de l'écoulement de vapeur à travers un volume traité en exemple à la section 4.2

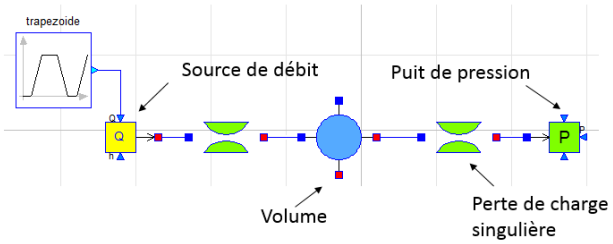

Figure 4-3: Allure graphique du modèle Modelica/Dymola ayant servi à simuler l'écoulement de vapeur surchauffée à travers un volume

Construire ce modèle revient finalement à « brancher » les connecteurs des sous-modèles. Cet assemblage graphique permet de générer le code Modelica du modèle global dans lequel sont listés les sous-modèles utilisés et les connexions mises en œuvre, code montré sur la figure ci-dessous en exemple :

```
model test_VolumeEcoulementVapeur

  ThermoSysPro.WaterSteam.PressureLosses.SingularPressureLoss    singularPressureLoss  a;
  ThermoSysPro.WaterSteam.PressureLosses.SingularPressureLoss    singularPressureLoss1   a;
  ThermoSysPro.WaterSteam.Volumes.VolumeA volumeA(V=5, dynamic_mass_balance=true) a;
  ThermoSysPro.WaterSteam.BoundaryConditions.SourceQ sourceQ(h0=3e6) a;
  ThermoSysPro.WaterSteam.BoundaryConditions.SinkP sinkP(P0=100000) a;
  ThermoSysPro.InstrumentationAndControl.Blocks.Sources.Trapezoide trapezoide(
    amplitude=1,
    rising=0.001,
    largeur=5,
    falling=0.001,
    periode=10,
    n=5,
    offset=1,
    startTime=5)
    a;
equation
  connect(singularPressureLoss.C2, volumeA.Ce1) a;
  connect(volumeA.Cs1, singularPressureLoss1.C1) a;
  connect(sourceQ.C, singularPressureLoss.C1) a;
  connect(singularPressureLoss1.C2, sinkP.C) a;
  connect(trapezoide.y, sourceQ.IMassFlow) a;
  a
end test_VolumeEcoulementVapeur;
```

Figure 4-4 : Code Modelica du modèle d'écoulement de vapeur surchauffée à travers un volume

Les connexions des ports des sous-modèles, réalisées ici avec le mot-clé « connect », génèrent des échanges de valeurs entre ces ports, qui imposent l'égalité exacte grandeur par grandeur. Le modèle global obtenu consiste donc en un système acausal des équations propres à chaque sous-modèle et des équations issues de leurs connexions.

4.4.2. Résolution du modèle global : les différentes étapes

4.4.2.1. Projection, traduction et traitement du modèle

La première étape exécutée par le logiciel est la projection et la traduction du modèle Modelica en un modèle mathématique, en regroupant et en projetant toutes les équations. Le modèle mathématique obtenu consiste généralement en un système hybride constitué principalement d'équations différentielles algébriques - le terme anglais DAE pour *Differential Algebraic Equations* est souvent utilisé - . On parle de système hybride, car on y trouve également des équations discrètes, liées aux événements temporels spécifiés dans le modèle. Les modèles Modelica typiques aboutissent à des

systèmes à un très grand nombre d'équations – parfois jusqu'à plusieurs centaines de milliers -, dont une partie importante d'équations comportant seulement un petit nombre de variables. Ces caractéristiques imposent l'utilisation de méthodes spécifiques pour traiter le système, de façon préliminaire à sa résolution par des solveurs. En effet, bien que les solveurs numériques de la famille DASSL (*Differential/Algebraic System Solver,* solveur par défaut de Dymola) soient conçus pour résoudre des systèmes de type DAE, Dymola transforme ce système mathématique en un système d'équations différentielles ordinaires, dit ODE (pour *Ordinary Differential Equation*). Un système ODE ne contient pas d'équations algébriques impliquant les fonctions à résoudre, ce qui rend plus simple la résolution du système.

L'étape de traduction du modèle Modelica est exécutée par la commande « Translate » dans le logiciel Dymola. La projection du code du modèle est visible dans le fichier « *modèle*.mof » généré par l'exécution de cette commande. Le fichier liste l'intégralité des variables déclarées, des équations, des fonctions appelées, des algorithmes utilisés, etc. Un autre fichier, « dmodel.mof » est également généré par la traduction du modèle. Ce fichier contient le détail des systèmes d'équations du modèle, et son analyse mathématique.

On peut reprendre l'exemple de l'écoulement de vapeur surchauffée dans un volume, en analysant comment le traducteur de Dymola traite le système d'équations du modèle, selon qu'une manipulation symbolique suffise ou que le système doive résoudre une boucle algébrique :

	Modélisation statique	***Modélisation dynamique***
Système d'équations du modèle	$0 = Q_{entrée} - Q_{sortie}$ (IV-1) $V\left[\rho\frac{\partial h}{\partial t}\right] = Q_{entrée}h_{entrée} - Q_{sortie}h_{sortie}$ (IV-2)	$V(\frac{\partial \rho}{\partial P}\frac{\partial P}{\partial t} + \frac{\partial \rho}{\partial h}\frac{\partial h}{\partial t}) = Q_{entrée} - Q_{sortie}$ (IV-3) $V\left[\left(h\frac{\partial \rho}{\partial P} - 1\right)\frac{\partial P}{\partial t} + (h\frac{\partial \rho}{\partial h} + \rho)\frac{\partial h}{\partial t}\right] = Q_{entrée}h_{entrée} - Q_{sortie}h_{sortie}$ (IV-4)

Manipulation des équations en code Modelica (fichier dsmodel.mof)	// Linear system of equations
	// Symbolic solution
	/* Original equation
	
	volumeA.V*volumeA.pro.d*der(volumeA.h) = volumeA.BH; */
	der(volumeA.h) := volumeA.BH/(volumeA.V*volumeA.pro.d);
	// End of linear system of equations
		// Linear system of equations
		// Matrix solution:
		/* Original equations:
		volumeA.V*(volumeA.pro.ddph*der(volumeA.P)+volumeA.pro.ddhp*der(volumeA.h))
		= volumeA.BQ;
		volumeA.V*((volumeA.h*volumeA.pro.ddph-1)*der(volumeA.P)+(volumeA.h*
		
		volumeA.pro.ddhp+volumeA.pro.d)*der(volumeA.h)) = volumeA.BH;
		
		J[1, 1] := volumeA.V*volumeA.pro.ddhp;
		J[1, 2] := volumeA.V*volumeA.pro.ddph;
		J[2,1] := volumeA.V*(volumeA.h*volumeA.pro.ddhp+volumeA.pro.d);
		J[2,2]:= volumeA.V*(volumeA.h*volumeA.pro.ddph-1);
		b[1] := volumeA.BQ;
		b[2] := volumeA.BH;
		
		x := Solve(J, b); // J*x = b
		{der(volumeA.h), der(volumeA.P)} := x;
		
		// Torn part
		// End of linear system of equations

Tableau 4-1 : Equations bilan du volume de vapeur surchauffée

Les variables d'état du système sont la pression P et l'enthalpie spécifique h. Q est le débit, V le volume de contrôle, ρ sa masse volumique.

Dymola cherche donc à transformer les équations de bilan en un système ODE explicite du type :

$$\dot{\boldsymbol{x}}(\boldsymbol{t}) = \boldsymbol{f}(\boldsymbol{x}(\boldsymbol{t}), t) \tag{IV-5}$$

Avec $\dot{\boldsymbol{x}}$ le vecteur des dérivées des variables d'état :

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dfrac{\partial h}{\partial t} \\ \dfrac{\partial P}{\partial t} \end{bmatrix} \tag{IV-6}$$

On voit que pour la modélisation statique, Dymola effectue une simple manipulation symbolique pour obtenir la forme désirée. On observe en revanche que pour la modélisation dynamique, le traducteur calcule une matrice Jacobienne J pour la résolution d'un système de la forme :

$$J\dot{\boldsymbol{x}} = \boldsymbol{b} \tag{IV-7}$$

le vecteur $\boldsymbol{b}$ contenant les termes d'entrée-sortie du volume de contrôle, ici :

$$\boldsymbol{b} = \begin{bmatrix} Q_{entrée} - Q_{sortie}\ (= BQ) \\ Q_{entrée}h_{entrée} - Q_{sortie}h_{sortie} (= BH) \end{bmatrix} \tag{IV-8}$$

Cette manipulation des équations, dont un exemple est montré ici pour un système linéaire simple, est exécutée pour un grand nombre de systèmes d'équations, dans le but de transformer ces systèmes DAE hybrides en des systèmes ODE. On parle également de réduction d'indice pour désigner cette manipulation.

Dans le domaine de la résolution des systèmes d'équations issus d'une modélisation orientée objet, on utilise en effet cette notion d'indice du système DAE, que l'on définit comme le nombre minimum de dérivations nécessaires de ses équations pour le transformer en système ODE explicite. L'indice de ce dernier est donc 0, tandis que celui des systèmes DAE simples est généralement 1. Cependant, de par le grand nombre de variables et d'équations algébriques qu'ils génèrent, les modèles Modelica donnent des indices souvent supérieurs à 1. Il est possible d'influencer cet indice en jouant sur les modèles, c'est-à-dire en adaptant le nombre et le type des composants utilisés. Il reste cependant difficile d'observer l'effet réel sur le traitement numérique, celui-ci n'étant pas totalement transparent sous le logiciel Dymola.

Une combinaison de deux algorithmes est utilisée pour effectuer une réduction de l'indice :

- L'algorithme de Pantelides sert au calcul de l'indice et à l'identification des opérations à mener sur les équations du système ;
- La méthode des dérivées « muettes » consiste à modifier certaines équations en replaçant des termes de dérivées de variables par des variables algébriques.

Pour plus d'informations concernant le traitement symbolique et la réduction d'indice des systèmes DAE issus de modèles Modelica, on pourra consulter les chapitres 17 et 18 de l'ouvrage de Fritzson (2015) [2]. On note toute de même qu'il est possible de

4.4.2.2. Résolution des systèmes d'équations

Après l'étape de traitement et d'optimisation des systèmes d'équations générés par le modèle Modelica global, Dymola procède à la résolution des systèmes ODE obtenus. Le solveur utilisé par défaut dans Dymola est une version modifiée du solveur DASSL, nommée DASPK. DASSL, conçu par la chercheuse Linda Petzold en 1983 [4], est un solveur à pas de temps d'intégration variable, utilisant une méthode numérique implicite à plusieurs étapes dite « BDF » (pour *Backward Differentiation Formula*).

DASPK a été proposé par Brown et al. (1994) [5] en tant que version améliorée de DASSL pour la résolution de systèmes DAE à grand nombre d'équations. Le solveur est néanmoins désigné par l'acronyme DASSL dans l'interface de configuration de la simulation dans le logiciel Dymola. La figure ci-dessous montre l'évolution du pas de temps du solveur pour les simulations stationnaire et dynamique de notre exemple de référence. On peut voir que la résolution des équations du modèle dynamique requiert des pas de temps plus faibles.

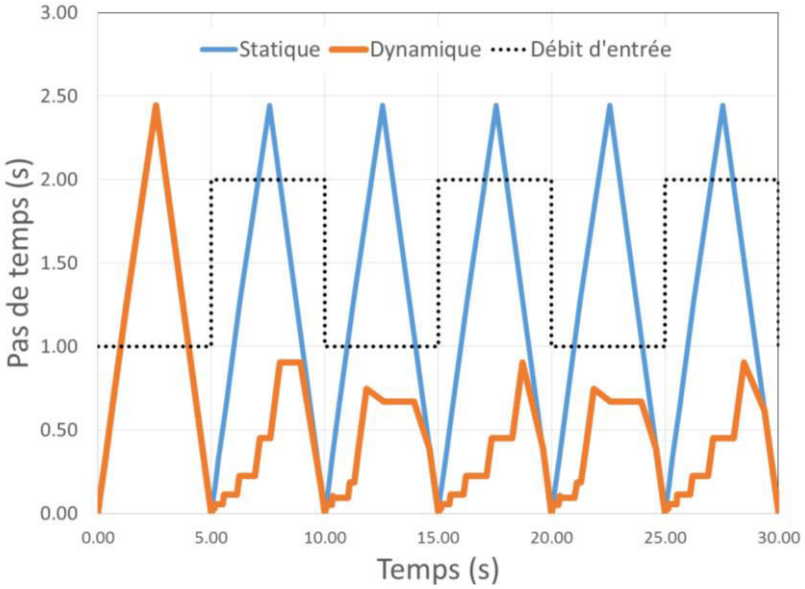

Figure 4-5 : Simulation de l'écoulement de vapeur surchauffée à travers un volume avec le solveur DASPK

Dymola propose de nombreux autres solveurs numériques, notamment l'algorithme LSODAR, à pas de temps variable, qui combine les solveurs LSODE1 et LSODE2 (*Livermore Solver for Ordinary Differential Equations*). Le premier utilise une méthode à plusieurs étapes dite de « Adams-Bashforth-Moulton », le deuxième une méthode BDF, mieux adaptée aux systèmes d'équations différentielles raides. Une équation différentielle est dite raide si sa résolution par une méthode numérique explicite est difficilement stable, à moins d'utiliser de très faibles pas de temps. L'algorithme LSODAR permet de passer de l'un à l'autre durant la résolution. On pourra consulter la référence [6] pour plus d'informations sur les solveurs de la famille LSODE, et la documentation technique de Dymola pour la liste complète des solveurs disponibles dans le logiciel.

Les modèles Modelica développés dans ce travail comportant à la fois un très grand nombre d'équations, de nombreux événements temporels, et beaucoup d'équations différentielles raides, le solveur DASSL semble à priori le plus adapté pour les simulations.

4.4.3. Traitement des résultats

Dymola fournit à l'utilisateur une interface pour visualiser le résultat de la simulation effectuée. Il est possible de tracer l'évolution des variables du modèle déclarées en « parameter » ou dans la section « public ». On peut également visualiser les variables des connecteurs. La figure suivante montre l'interface de visualisation de Dymola pour le même exemple utilisé précédemment. On voit que les variables sont « déroulées » par sous-modèle à la façon d'un arbre.

Malgré cette interface disponible, et ce dans un souci d'automatisation et de praticité, nous avons développé durant cette thèse des codes de post-traitement pour récupérer les variables des modèles sous

forme de vecteurs de données, afin de réaliser toutes sortes d'opération et de traitement. Ces codes ont été développés dans un premier temps avec le logiciel Matlab, car Dymola fournit dans son répertoire d'installation des fichiers *.m*, qui contiennent des fonctions utilisables sous Matlab pour interagir avec Dymola. Ces fonctions incluent le « pilotage » du logiciel, c'est-à-dire l'exécution des principales actions de Dymola (compilation-traduction, simulation) depuis Matlab. Outre ces fonctions d'interface, il apparait clairement que le logiciel de Dassault Système a été développé dans une optique d'utilisation en parallèle du logiciel de traitement mathématique, car les données de simulation sont stockées dans des fichiers *.mat*, format de stockage propre au logiciel Matlab.

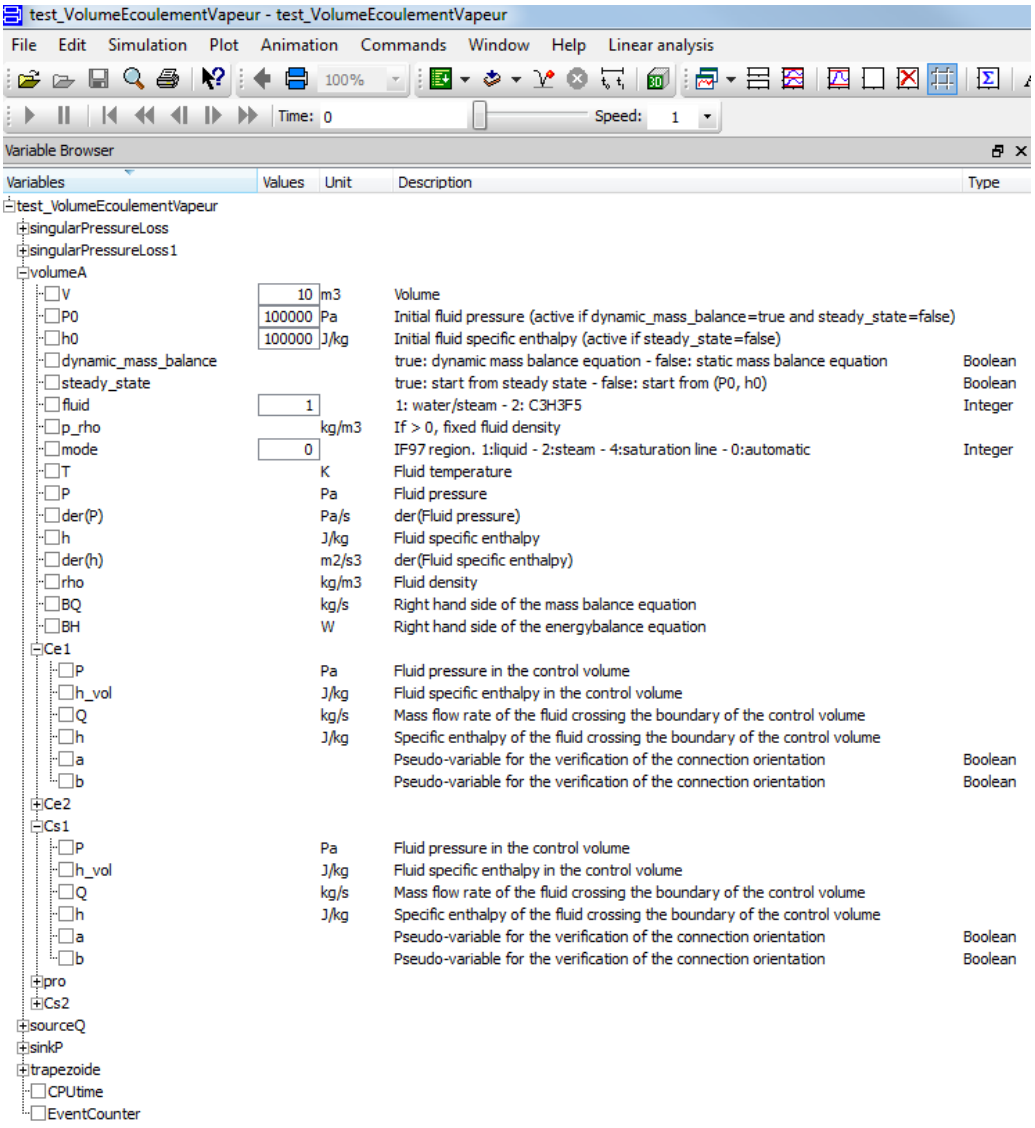

Figure 4-6 : Vue de l'interface de visualisation des variables de simulation de Dymola

Dans un deuxième temps, des codes de post-traitement ont été développés pour une utilisation sous Scilab, un équivalent de Matlab disponible sous licence libre, ce qui permet de s'affranchir du besoin d'accès payant à des licences. Un script a notamment été développé, en collaboration avec l'équipe du laboratoire travaillant sur la modélisation des réseaux de chaleur (également utilisatrice de Dymola), permettant d'extraire de manière ordonnée les variables voulues d'un fichier *.mat*. On dispose donc d'un équivalent Scilab (*.sci* ou *.sce*) de la fonction *dymget.m* fournie dans le répertoire de Dymola (*Mfiles/dymtools*).

4.5. Les librairies de modèles, et la librairie ThermoSysPro

Les librairies (ou bibliothèques) Modelica sont des ensembles cohérents de modèles, dédiés généralement à la modélisation de systèmes appartenant à des domaines particuliers. Si le langage Modelica est sous licence libre, certaines librairies sont distribuées également sous licence libre, tandis que d'autres sont sous licence commerciale.

La librairie standard de Modelica (MSL, pour Modelica Standard Library) est une librairie libre développée et maintenue par l'Association Modelica. Elle est la librairie de modèles de base à laquelle font appel les modèles de certaines autres librairies, libres ou commerciales.

La bibliothèque ThermoSysPro est développée est maintenue par le département R&D de EDF, à l'origine pour la modélisation de centrales électriques conventionnelles ou de leurs composants [7], [8] . Elle a été appliquée plus récemment à la modélisation et la simulation de centrales CSP [9–11]. La figure ci-dessous à gauche montre l'organisation de la librairie en termes de packages. Ces packages sont des regroupements de sous-modèles tournés chacun vers un même thème d'applications techniques.

Figure 4-7 : Composition (gauche) et logo (droite) de la librairie ThermoSysPro

ThermoSysPro a été choisie comme base pour la modélisation des champs solaires DSG principalement pour l'utilisation du package *WaterSteam*, ainsi que pour l'utilisation du modèle de collecteur cylindro-parabolique du package *Solar*. Le package *WaterSteam* contient des modèles permettant de simuler des procédés thermo-hydrauliques basés sur des écoulements eau/vapeur (tuyauterie, vannes, échangeurs thermiques, volumes et ballons, etc.). Le package *Properties* contient également un « sous-package » *WaterSteam* qui permet de fournir aux modèles de la librairie les propriétés de l'eau/vapeur.

La formulation industrielle IF97 (Industrial Formulation 97) de l'association internationale pour les propriétés de l'eau et de la vapeur (IAPWS-International Association for the Properties of Water and Steam) est transcrite et utilisée dans la librairie, ce qui a aussi motivé son choix. L'IAWPS est une association internationale à but non lucratif dont une des tâches majeures est de proposer des formulations normalisées de calcul des propriétés de l'eau/vapeur. La formulation IF97 est le standard le plus largement utilisé aujourd'hui. Cette formulation propose un fonctionnement optimisé du point de vue de l'algorithmique en présentant un faible temps d'utilisation du processeur.

Le standard IF97 consiste concrètement en un ensemble d'équations permettant d'accéder aux propriétés du fluide, ces équations différant selon la région d'état thermodynamique du fluide. Il divise l'ensemble des états en 5 régions :

- Région 1 : eau liquide sous-refroidie ;
- Région 2 : vapeur surchauffée ;
- Région 3 : saturation et zone diphasique ;
- Région 4 : vaporisation et condensation
- Région 5 : vapeur surchauffée avec p < 50 MPa et 1073.15 K < T < 2273.15 K

La figure ci-dessous montre ces cinq régions sur un diagramme pression-température. Pour plus de détails concernant la formulation IAWPS-IF97, on pourra consulter les travaux de Bonilla et al. [12].

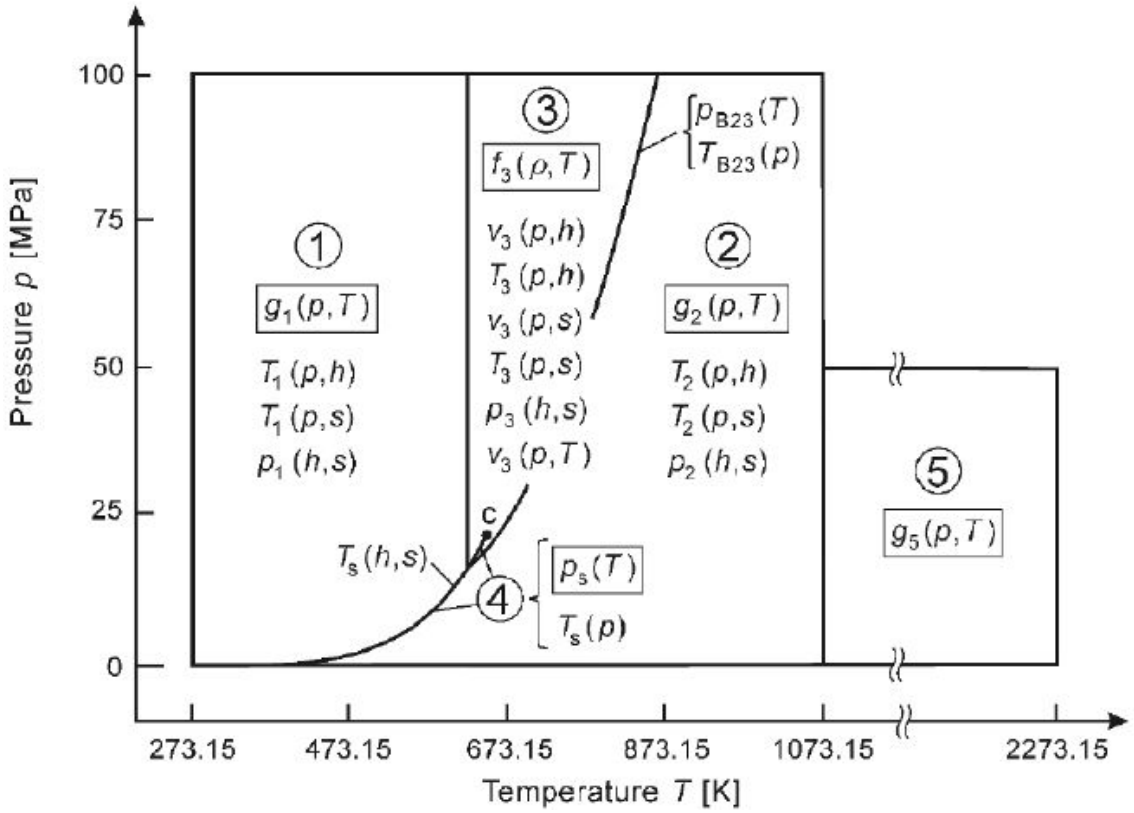

Figure 4-8 : Régions de la formulation IAWPS-IF97 sur un diagramme pression-température, extrait de [12]

4.6. Références

[1] P. Fritzson, *Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica*. Wiley-IEEE Press, 2011.

[2] P. Fritzson, *Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. The Institute of Electrical and Electronics Engineers, Inc, 2015.

[3] "http://www.modelica.org." .

[4] L. Petzold, "A description of DASSL: a differential/algebraic system solver.," *Scientific Computing*, pp. 65-68, 1983.

[5] P. Brown, A. . Hindmarsh, and L. Petzold, "Using Krylov methods in the solution of large-scale differential/algebraic systems," *SIAM Journal on Scientific Computing*, vol. 15, no. 6, pp. 1467-1488, 1994.

[6] A. . Hindmarsh, "ODEPACK, A systematized collection of ODE solvers," *IMACS Transactions on Scientific Computation*, vol. 1, pp. 55-64, 1983.

[7] B. El-hefni, D. Bouskela, and G. Gentilini, "Dynamic modelling of a Condenser / Water Heater with the ThermoSysPro Library," in *Proceedings of the 9th International Modelica Conference*, 2012, pp. 745-758.

[8] D. Bouskela, G. Lebreton, and B. El-hefni, "Dynamic modelling of a combined cycle power plant with ThermoSysPro Introduction to ThermoSysPro," in *Proceedings of the 8th International Modelica Conference*, 2011, pp. 365-375.

[9] B. El-hefni, "Dynamic modeling of concentrated solar power plants with the ThermoSysPro library (Parabolic Trough collectors , Fresnel reflector and Solar-Hybrid)," *Energy Procedia*, vol. 49, pp. 1127-1137, 2014.

[10] J. B. Zhang, J. C. Valle-marcos, B. El-hefni, Z. F. Wang, G. F. Chen, and G. C. Ma, "Dynamic simulation of a 1MWe concentrated solar power tower plant system with Dymola ®," in *SolarPACES2009*, 2013, vol. 0.

[11] S. J. Liu et al., "Dynamic simulation of a 1MWe CSP tower plant with two-level thermal storage implemented with control system," *Energy Procedia*, vol. 69, pp. 1335-1343, 2015.

[12] J. Bonilla, E. Zarza, L. J. Yebra, and S. Dormido, *Modeling and Simulation of Two-phase flow*. Coleccion Documentos Ciemat.