Chapitre 2

Contexte et état de l'art de la simulation

Les systèmes étudiés de nos jours deviennent plus grands et complexes, ce qui les rend difficiles à comprendre. Une analyse structurée du système peut nous amener à une description pertinente pour la simulation du système.

Abordons le problème de la complexité. Nous allons d'abord présenter une approche générale pour aborder ce genre de problème, souvent connue comme *approche systémique*. Ensuite les questions relatives à la simulation seront traitées. Toutes les stratégies adoptées par les divers laboratoires afin de réduire la complexité d'un système thermique utilisent la même idée de base : la modularité. Il n'est pas étonnant de trouver des approches différentes pour s'attaquer à ce problème. Un recensement des environnements de simulation actuellement disponibles peut nous donner une idée de l'état actuel de l'avancement des travaux. Nos propres idées pour un environnement de simulation terminent ce chapitre.

2.1 Contexte

2.1.1 Étude de systèmes : modélisation et simulation

Le mot *modélisation* revêt des significations très différentes selon les interlocuteurs :

- pour certains la modélisation correspond à l'ensemble des activités qui permettent la création, la mise au point et l'exécution sur un ordinateur de maquettes virtuelles des systèmes à étudier.

- pour d'autres, la modélisation se confond avec la *simulation numérique*, c'est à dire avec la résolution d'équations d'évolution de la physique.

- pour d'autres enfin, il s'agit de l'*élaboration de relations* entre les variables caractéristiques d'un système ou processus physique donné, capables de bien simuler le comportement de ce système dans un contexte donné.

Notre point de vue correspond à la dernière définition. Elle est certainement plus restreinte que la première, mais dans l'étude d'un système nous voulons distinguer ici deux phases différentes : le développement de modèles d'une part et leur utilisation dans une étude concrète d'autre part. Seule la première phase sera appelée la *modélisation* et alors que pour la deuxième phase, nous étendrons le sens du mot *simulation*. L'exploitation de modèles n'est certainement pas limitée à la simulation. La simulation est un des objectifs possibles de la modélisation ; mais c'est ce qui nous intéresse ici. Cette séparation en deux phases a comme objectif la séparation possible des travaux d'un développeur de modèles de ceux d'un ingénieur de la simulation (utilisateur de modèles). La distinction entre les deux peut contribuer à l'échange des modèles et à leur application.

La première phase est la modélisation. Cette démarche essaye d'abstraire le monde réel pour trouver une image pertinente de la réalité. Cette création d'un concept se fait par des exemples et des généralisations[1]. En fait, chaque objet du monde réel n'a pas de définition *per se* et ne relève pas de manière absolue d'un modèle ou d'un autre. Un mur peut être vu comme un support pour des affiches, comme construction statique pour porter les étages supérieurs, comme résistance thermique pour garder la chaleur à l'intérieur de la maison, ... Chaque être humain attache sa propre idée à l'objet du monde réel. La communication entre les hommes ne peut se faire qu'à travers des parties communes et partagées d'un concept. Un des objectifs de la modélisation est alors la description formelle de ce concept (généralement exprimé sous forme mathématique).

La deuxième phase est l'exploitation de cette modélisation, qui consiste dans notre cas à « faire de la simulation ». La difficulté de cette phase est l'identification d'un objet réel comme une instance d'un modèle existant. On associe l'idée générale à l'objet concret. Cette reconnaissance de l'objet réel se fait en vue d'une future simulation du système. On est donc déjà guidé par les objectifs de son étude et on applique des simplifications. Le mur NORD de notre bâtiment comme objet concret se voit associé le phénomène conduction thermique monodimensionnelle. La simplification ici est de ne pas considérer les effets acoustiques, par exemple. Pour un système concret, on utilise les modèles construits dans la phase de modélisation, et on les applique à son problème. C'est ici, dans la deuxième phase, que l'on utilise une simulation du système modélisé pour obtenir des résultats qui sont autrement trop difficilement accessibles.

Remarquons que dans le langage courant, la modélisation d'un système comprend la spécification de paramètres pour les différents modules utilisés. Nous considérons cette affectation de valeurs comme une étape de la simulation.

1. Une autre manière de construire un modèle est purement abstraite. Il résulte d'un raisonnement et d'une construction logique sur des modèles et axiomes existants. C'est le cas dés mathématiques et de la philosophie

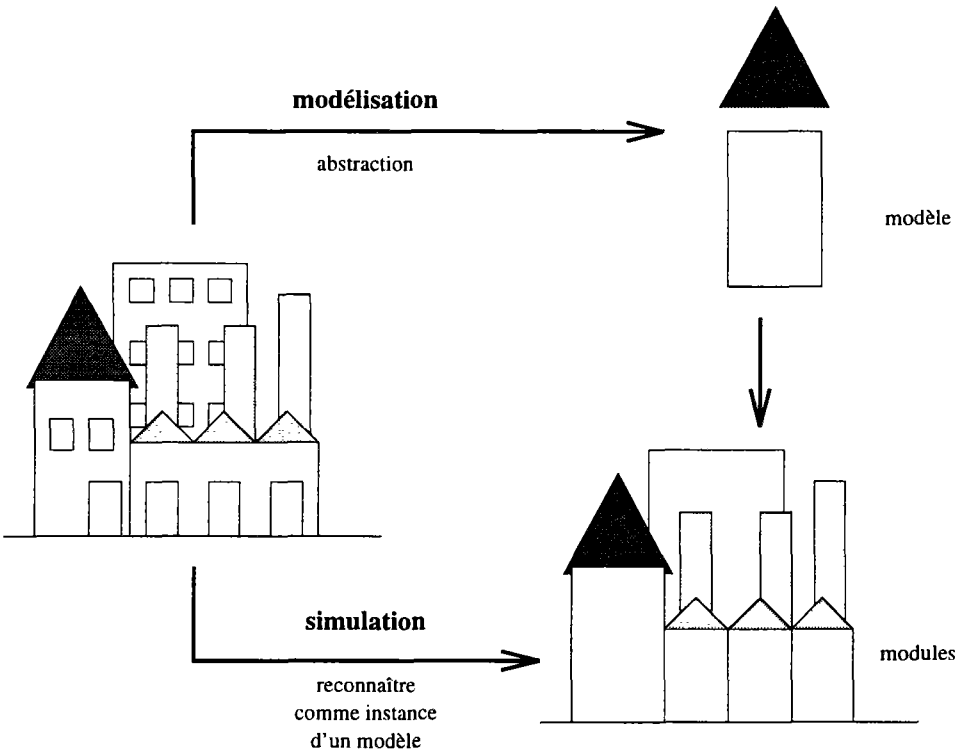


FIG. 2.1 - *Les deux deux phases principales d'une étude de système : la* modélisation *et la* simulation.

Nous associons aussi à la distinction entre ces deux phases une distinction de terminologie : pendant la modélisation – comme le mot l'indique – nous créons des *modèles* de la réalité ; dans la deuxième phase – l'exploitation de modèles, et plus particulièrement la simulation – nous utilisons des *modules* pour décrire un système. Un module est une instance d'un modèle général avec des paramètres concrets. Cette distinction entre les phases et le choix du vocabulaire en résultant seront approfondis dans le chapitre 4.1.

Nous allons donc présenter d'abord des méthodes de modélisation, et surtout les problèmes liés à la complexité. Ensuite les démarches de la deuxième phase d'une étude de système seront regardées plus en détail.

2.1.2 Complexité

La notion de complexité est très liée à celle de système. Par *système* nous entendons un ensemble d'objets qui interagissent, ce qui ajoute une nouvelle qualité à l'ensemble. Il faut insister ici sur l'activité d'un objet. Pour nous un système devient de plus en plus *complexe*, si les interactions entre les objets sont tellement multiplement entrelacées, que les rapports entre les objets et le comportement du système – aussi bien au niveau global qu'au niveau local – deviennent de moins en moins apparents. La difficulté de la complexité est en nous même, c'est un problème d'intelligibilité. Mais la complexité ne s'arrête pas à l'énumération des composants et de leurs rapports dans un système.

La qualité supplémentaire ajoutée à l'ensemble, qui est l'organisation, est une condition de complexité. On peut lire chez E. MORIN, que c'est « *la capacité d'un système à, à la fois, produire et se produire, relier et se relier, maintenir et se maintenir, transformer et se transformer* » [60]. J.-L. LE MOIGNE décrit le *système général* comme : « - *quelque chose (n'importe quoi, présumé identifiable),* - *qui* dans *quelque chose (environnement),* - pour *quelque chose (finalité ou projet),* - fait *quelque chose (activité = fonctionnement),* - par *quelque chose (structure = forme stable),* - *[et] qui* se transforme *dans le temps (évolution)* » [53]. On remarque dans les deux définitions l'accent mis sur l'activité et l'évolution d'un système qui le rendent *complexe*. LE MOIGNE utilise le terme *processeur* pour décrire les composants d'un système afin d'en souligner l'aspect actif. Par ailleurs, ces définitions ne définissent même pas un *système complexe*, mais plutôt un *système* tout court. Le caractère complexe est toujours sous-entendu dans la notion de système. Néanmoins les système en équilibre peuvent être complexes eux-aussi, car leur état et leurs dépendances intérieures sont quelques fois encore plus difficiles à déterminer.

L'*approche systémique* est donc une tentative de développer des théories et méthodes avec lesquelles on peut comprendre, examiner, interpréter, évaluer, construire, manipuler et changer des systèmes, mettre en évidence les dépendances. L'approche systémique cherche à mieux comprendre la complexité organisée en fournissant une vision macroscopique du comportement global, tout en acceptant que certains détails ne sont pas traités en profondeur.

Notre objectif ici n'est pas une approche générale pour tout système. Nous nous intéressons plutôt à la conception et au contrôle des systèmes techniques et artificiels. Par rapport à la sociologie par exemple, l'application de l'approche systémique dans les sciences physiques et naturelles est souvent allégée par des mesures plus facilement quantifiables. Il faut néanmoins remarquer qu'il n'y a pas de système complètement technique. L'interaction humaine existe partout. La complexité des systèmes technique se révèle lors d'une conception plus détaillée, d'une conception sur mesure (nouveaux matériaux et traitements), avec une multiplication de dépendances. Au niveau calculatoire, on travaille souvent dans un champ de fonctionnement restreint, avec des phénomènes dont la description mathématique pose des problèmes à la résolution (phénomènes non-linéaires et non-monotones). Les systèmes techniques et artificiels sont le plus souvent équipés de dispositifs de contrôle de plus en plus sophistiqués, ce qui ajoute encore à la difficulté de les comprendre et de les modéliser.

2.1.3 Modélisation

La complexité croissante des systèmes a augmenté la difficulté de l'analyse et a mis en évidence la nécessité des nouvelles méthodes pour mieux aborder ce type de problème. Aux différents niveaux, on a pu constater des développements, comme les algorithmes sur les grandes matrices en algèbre linéaire, ou le calcul parallèle. Pour la compréhension et la conception des systèmes complexes, l'approche systémique est souvent utilisée.

Née de la cybernétique dans les années 50, l'approche systémique est aujourd'hui appliquée dans des domaines très différents comme la sociologie [20], l'économie [72], [43], l'écologie [79]; de plus depuis quelque temps elle a été (re-)découverte par les ingénieurs [67], [48], [42].

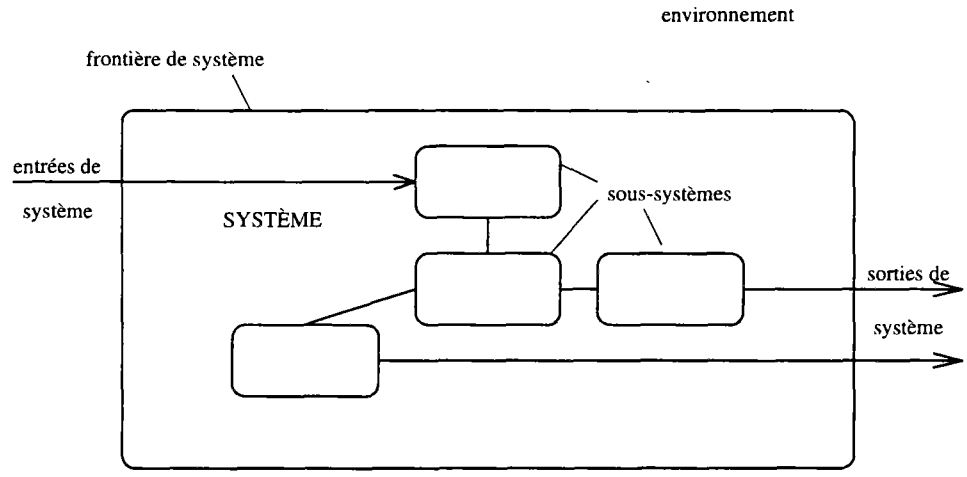


FIG. 2.2 - *Le concept général d'un système avec le découpage de l'environnement, des entrées/sorties et des sous-systèmes. La structure du système, le nombre et le type des éléments et le nombre et le type de raccordements entre les éléments et avec l'extérieur déterminent le système et sa complexité.*

Quelques questions sont typiques pour l'approche systémique, mais elles s'appliquent également à l'analyse d'un système en général. Il convient d'en être conscient pendant toutes les phases d'une étude. Les problèmes suivant nous accompagnent tout au long d'une étude systémique:

Découpage Un système n'est pas isolé de son environnement. Il faut que la partie choisie de la réalité soit plus ou moins autonome (« *système quasi isolé* »). Un critère possible pour le découpage est la minimisation des débits à travers les frontières. Il faut faire attention au fait que les différents types de débits (d'énergie, de matériaux, d'informations) ont en général des frontières à débit minimale différentes. Selon le critère choisi on découpe le système de son environnement en des endroits différents.

Intégralité Cette préoccupation est complémentaire au problème de découpage. N'a-t-on rien oublié de ce qui est important pour la compréhension du système? Une question à laquelle on ne peut pas répondre *a priori*.

Granularité C'est la question du niveau de détail avec lequel nous devons décrire le système. Une granularité trop fine rend le modèle trop lourd et compliqué; une granularité trop grossière ne prend pas en compte des détails importants. Il peut éventuellement s'avérer nécessaire de travailler avec une gamme de modèles à granularités différentes et changer de modèle en fonction des besoins réels.

Échelle de temps Ici il s'agit de choisir des constantes de temps qui sont importantes pour notre problème. (Regarde-t-on l'évolution en millisecondes ou en heures?) Granularité et échelle de temps sont souvent liées. Plus la granularité est fine, plus l'échelle de temps doit être courte. Généralement on peut souvent supposer une inertie plus petite pour un petit système. La conclusion réciproque n'est pas toujours vraie ; même une description grossière du système peut nécessiter une échelle de temps courte.

Représentation La structure mathématique de la description du système définit la représentation. Le choix d'une représentation fait donc partie de la traduction d'une image « intuitive » vers une description mathématique et informatique (nous allons approfondir l'idée d'une représentation à niveaux d'abstractions différents dans le § 4.1). Généralement on est conscient des problèmes mentionnés jusqu'ici. Mais le type de questions que l'on peut se poser au sujet d'un modèle dépend fortement de la représentation choisie.

Prenons comme exemple un mur. On peut modéliser le champ de température comme un problème mono-, bi-, ou tridimensionnel. De plus il faut se décider, entre autre, entre une représentation analytique (au moins pour le cas monodimensionnel), une approche variationnelle, une représentation purement paramétrique ou une discrétisation spatiale. Une représentation par des modes propres est difficile à interpréter physiquement, mais elle apporte des avantages comme la possibilité d'une analyse particulièrement pertinente et riche et d'une réduction de modèle pour la simulation. Une modélisation sans capacité thermique caractérise directement le régime permanent. On peut aussi envisager un modèle qui ne prend pas en compte le comportement thermique mais plutôt ses propriétés mécanique ou acoustiques, bien que notre préoccupation principale reste attachée aux problèmes thermiques. Il existe alors un grand nombre de représentations différentes auxquelles on ne peut pas poser les mêmes questions.

Il peut être éventuellement utile de garder plusieurs représentations à la fois. Selon les simplifications que l'on s'autorise et selon les questions auxquelles on veut des réponses, on peut alors choisir parmi une gamme de modèles pour trouver le modèle le plus adapté. Il se pose alors la question du raccord entre des modèles différents.

Complexité Tout modèle de système est *a priori* une simplification ; la réalité est toujours plus riche que le modèle. Un choix n'est justifiable qu'*a posteriori*, si on prouve (par exemple par expérimentation) que le modèle s'est comporté comme la réalité dans le cadre de la précision choisie et pour l'environnement prévu (grandeurs et fréquences d'excitations).

Autrement dit, si l'on veut obtenir des résultats fiables, il faut que la complexité du système simulant soit plus élevée que celle du système à simuler [6]. Ceci étant impossible, il faut se contenter de résultats approximatifs. Dans la pratique, on ne veut pas simuler *tout* le système,

mais seulement certains aspects. Cela permet des modèles moins complexes que la réalité et quand même fiables dans le domaine observé.

Il faut par ailleurs remarquer, qu'un comportement apparemment complexe n'a pas forcement son origine dans des lois complexes. Par exemple, les systèmes chaotiques comme les ensembles de MANDELBROT sont souvent déterminés par des lois très simples.

Différentes méthodes ont été développées pour appliquer de l'approche systémique à un contexte concret ; par exemple la *méthode de sensibilité* de VESTER mise au point pour l'UNESCO [78]. Cette méthode part de la création du « bon» ensemble de variables, pour déterminer ensuite le réseau d' « interdépendances» entre les modules. Une distinction entre éléments *actifs, critiques, neutres* et *passifs* permet d'évaluer la dynamique du système. Malgré de grands efforts, l'application de l'approche systémique au domaines techniques (génie d'ingénieur) n'est pas encore aussi bien formalisée.

2.1.4 Exploitation de la modélisation : la simulation

Les questions détaillées plus haut sont plutôt liées à des démarches de modélisation qui prennent en compte une future exploitation par simulation. Pour nous, la simulation est la deuxième phase principale d'une étude de système. Cette exploitation de la modélisation est composée de plusieurs étapes, qui ne s'enchaînent pas toujours séquentiellement. Assez souvent on est contraint de revenir à une étape précédente pour réévaluer les décisions.

Un ordinateur peut alléger cette étude de différentes façons. Il intervient surtout dans la phase numérique (phase de la résolution du système d'équations), mais aussi lors de la modélisation des nouveaux composants et de l'interprétation des résultats.

Ces différentes phases sont soutenues sur l'ordinateur par des modules de logiciel différents. Le plus souvent on ne parle pas d'un programme de simulation mais plutôt d'environnement de simulation. Les produits disponibles (et ceux qui sont en cours de développement) ne sont souvent pas limités à un seul programme mais ils sont eux mêmes composés de plusieurs modules, chacun conçu pour traiter un aspect particulier du processus de modélisation. Cet ensemble constitue un environnement de simulation.

On peut par exemple énumérer les phases de la simulation – dans le sens large d'une deuxième phase d'une analyse systémique –, et étudier comment on peut profiter de l'aide d'un ordinateur à chaque étape.

Description générale du problème On commence par une description du problème d'une façon informelle, dans un langage naturel. On observe le système tant que possible; on interroge les personnes concernées (un système n'est jamais purement technique). L'avis des experts et l'extraction de l'information contenue dans la documentation existante relative au domaine comme les article et les livres, approfondit la perception.

Pour mieux structurer cette phase initiale, l'aide de l'ordinateur peut intervenir non seulement au niveau de l'organisation comme par des logiciels de bureautique ou par des logiciels d'organisation des projets, mais aussi par des logiciels de calcul formel et des bases de données électroniques comme des bibliothèques de fonctions.

Identification des processeurs et leurs rapports Puis on cherche à identifier les modules qui constituent le système. Les liens et dépendances (topographiques, logiques, énergétiques) entre les objets sont aussi importants que les objets mêmes. Différentes méthodes existent pour les mettre en évidence (par exemple [78])

L'identification des composants passe par un découpage du système en sous-systèmes. Un découpage peut se présenter de façon très naturelle. On fait par exemple la distinction entre le bâtiment (murs, plancher etc.) et les équipements de bâtiments (par exemple le système de chauffage). Mais on peut aussi essayer de découper le système hiérarchiquement (par exemple les étages, pièces, ...). On retrouve alors ici la notion de granularité et de structure (jusqu'où faut-il couper? quelle est la structure du système?). Une analyse par découpage nous amène à un « graphe » de dépendances et d'appartenances entre les composants.

Un environnement graphique et interactif comme support sur ordinateur améliore certainement la convivialité du travail de la description. On peut imaginer l'intégration d'une méthode schématisée dans cet environnement qui dirige le progrès de la description. Un guide automatisé, géré par un système expert, peut éventuellement simplifier la mise au point d'une description pertinente du système pour l'utilisateur.

Modélisation des phénomènes Pour chacun des composants, un modèle de représentation est nécessaire. On associe aux modules des phénomènes physiques ou des lois empiriques (voir § 2.2.2 pour des différents types de modèle). Les modèles sont une représentation mathématique ou algorithmique qui relient les paramètres et les variables entre elles permettant d'effectuer ultérieurement une étude qualitative.

Ces modèles de calcul avec les équations constitutives identifiées peuvent être stockées dans une bibliothèque de modèles. Une telle bibliothèque contenant des modèles couramment utilisés est pratique, parce qu'elle aide à éliminer les duplications. Cela apporte un gain de temps et nous épargne des redéveloppement de modèles généraux. Néanmoins, les paramètres libres restent toujours à déterminer. En utilisant une telle bibliothèque, la description d'un système consiste finalement à raccorder des modules pré-définis.

Un environnement de simulation doit permettre le développement de nouveaux modèles. Il ne peut pas être uniquement basé sur une « modélothèque » de composants existants. Il doit fournir les outils pour la modélisation comme la description de nouvelles équations, l'exploitation de résultats expérimentaux ou la description algorithmique d'états.

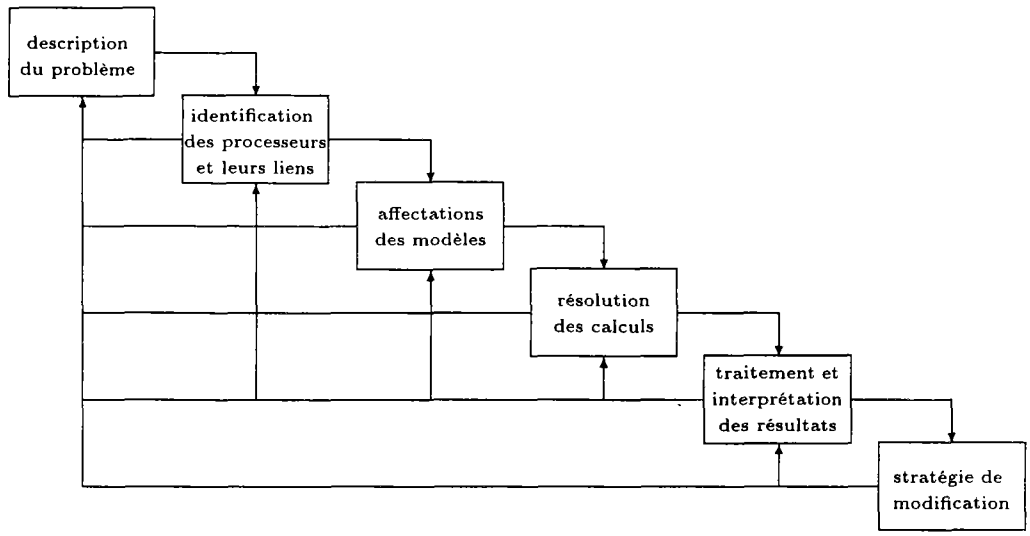


FIG. 2.3 - *Les phases d'une étude de système sont des démarches itératives. Des réflexions sont nécessaires à chaque étape s'il faut revenir à une étape antérieure ou si l'on peut progresser.*

Résolution des calculs La représentation finale d'un problème est un ensemble d'algorithmes sous forme d'un programme d'ordinateur. Pour obtenir des résultats quantitatifs, plusieurs approches existent. Par exemple l'analyse et la synthèse modale permettent de prévoir le comportement dynamique d'un bâtiment rien que par observation de ses modes (voir [52, chap. 1.4]). Dans notre cas, on s'intéresse surtout à la simulation quantitative du comportement temporel du système. En général, il n'y a pas de solution analytique et l'on a recours à des méthodes numérique pour la résolution. Dans le cas de système dynamique, c'est une intégration temporelle des équations différentielles. Aujourd'hui on se repose presque toujours sur la puissance d'un ordinateur pour exécuter les nombreux calculs.

Traitement des résultats Finalement, on peut exploiter les résultats obtenus, par exemple visualiser des champs de température, déterminer les besoins de chauffage d'un bâtiment ou évaluer le risque de surchauffe d'un moteur.

Les résultats se laissent mieux comprendre lorsqu'ils sont représentés par des courbes et des dessins en couleur plutôt que par des listes de nombres. Une présentation graphique de résultats permet de mieux comprendre l'évolution. Ensuite, on peut plus facilement en déduire les conséquences pour la configuration du système, au moins de façon qualitative.

Interprétation des résultats L'interprétation détermine le rôle des éléments dans le système et les propriétés de la structure globale. On peut évaluer la stabilité, la souplesse et/ou l'efficacité du système.

Un support possible par ordinateur peut être un système expert. Mais le degré d'amélioration que l'on peut apporter au système étudié par une

interprétation des résultats par un système expert reste une question ouverte. Il existe quelques recherches dans ce domaine et les epérences sont grandes [55]. Les résultats applicables n'existent pas pour l'instant.

Stratégie Avec cette interprétation, on peut finalement développer une stratégie pour modifier le système. En fonction des sensibilités obtenues et connaissant les réponses probables du système, on a les moyens de modifier ou d'interagir avec le système. Une nouvelle conception du système est possible. Ceci peut être une adaptation des paramètres pour faire évoluer le système, mais on peut aussi penser à une réorganisation du modèle pour adapter la structure du système.

Ces phases de l'exploitation ne se suivent pas forcement dans l'ordre que nous avons mentionné ici. C'est plutôt un processus itératif ou à chaque étape les résultats sont à réévaluer. Si les résultats ne sont pas satisfaisants (par rapport aux valeurs d'expérience ou d'expérimentation) on recommence éventuellement à une phase précédente comme par exemple à l'identification des modules, ou on cherche une meilleure représentation et modélisation des modules existants.

L'outil idéal d'analyse systémique est un environnement qui offre un support spécifique pour chacun des points mentionnés dans les paragraphes précédents. Comme on ne peut pas prévoir tous les problèmes, les outils doivent s'adapter facilement aux besoins des études menées.

La résolution numérique du problème après sa modélisation est une phase parmi d'autres dans l'étude d'un système, comme on l'a vu ici. Assez souvent il s'agit de déterminer le comportement du modèle dans le temps, afin de mieux comprendre le système réel. Dans le paragraphe suivant nous examinons ce processus plus en détail.

2.2 Simulation

Le domaine de la simulation par ordinateur subit en ce moment un phénomène explosif d'innovation et d'élargissement des domaines d'application. Les capacités nouvelles du matériel (« transputers » par exemple) et des logiciels (parallélisme extrême et méthodes d'Intelligence Artificielle) accélèrent encore ce processus. Par ailleurs le prix du matériel ne cesse de baisser et les puissances de calcul accessibles ne cessent de croître.

2.2.1 Objectifs et méthodes de la simulation

Apparemment la simulation est intéressante surtout pour des problèmes complexes, qui sont difficilement abordables par l'expérimentation comme par exemple l'étude du confort des occupants d'un bâtiment. Dans ce cas, la simulation remplace l'expérimentation. Par ailleurs, la simulation est également souvent utilisée pour reproduire des tests, et de cette façon elle peut contribuer

à leur compréhension. Elle peut étendre le champs de ces tests sur le système à étudier et elle permet d'en modifier les paramètres caractéristiques pour comparer plusieurs cas similaires. De la même manière, la structure de l'ensemble et les modèles élémentaires peuvent être modifiés. Comprendre des phénomènes par modélisation peut être un moyen d'acquérir de nouvelles connaissances sur ce domaine. C'est la tâche du modélisateur.

L'exploitation des modèles qui sont validés par simulation ajoute une dimension supplémentaire aux tests. Cela peut donner des indications sur les actions futures à mener et donner des possibilités d'extrapolation. De cette façon la simulation est une aide précieuse à la conception des nouveaux systèmes, et incombe donc à l'utilisateur du logiciel de simulation.

Les objectifs des études qui reposent sur la simulation sont :

- la compréhension de la structure et des rapports à l'intérieur d'un système.
- l'analyse et la prédiction du comportement du système.
- la conception et le dimensionnement de l'équipement.
- l'aide à l'expérimentation, qui peut éventuellement finir par ...
- le remplacement de l'expérimentation, souvent très coûteuse et même quelque fois impossible.
- la validation des systèmes de contrôle.

Les résultats de simulation sont évalués en fonction de différents critères qui sont souvent concurrents. Pour une étude thermique de bâtiment cela peut être la qualité, le confort, les coûts ou la consommation.

Pour analyser la réalité par l'expérimentation et interpréter les résultats, on a développé des modèles qui sont des simplifications de cette réalité. Dans les sciences physiques, les modèles sont surtout des modèles mathématiques. Mais il existe aussi d'autres formes de représentation, comme un champ de distribution de probabilité, des règles pour un système expert, ou des connaissances stockées dans un réseau neuronal. En utilisant ces modèles, on peut calculer le comportement dans le temps du système. Par rapport à l'expérimentation, la simulation est plus simple, plus pratique et surtout moins chère. Dans le domaine du bâtiment, elle est souvent la seule possibilité d'étudier un système en raison des grandes dimensions spatiales qui induisent la taille du problème. Elle aide à la conception de l'équipement et de l'enveloppe. Simultanément elle apporte des connaissances sur la structure et sur les dépendances entre éléments à l'intérieur du système.

2.2.2 Types de simulation

La simulation est la reproduction artificielle – informatique essentiellement – d'un comportement réel. On suppose que cette image de la réalité se comporte

« de la même manière » que le phénomène à étudier. C'est le cas pour les simulations basées sur les modèles physiques.

Simulation analogique – simulation digitale La simulation peut être une expérimentation à échelle réduite qui permet l'extrapolation à l'échelle réelle. On peut observer cette approche dans les tunnels de vent par exemple. Par ailleurs on fait des expérimentations sur des systèmes plus simples, mais dans lesquels il existe une analogie des lois physiques. Les ordinateur analogues en sont un exemple. Parfois cette approche est appelée une *simulation analogique*. Aujourd'hui on a le plus souvent recours à la puissance de calcul des ordinateurs, la *simulation digitale*. L'ordinateur est un outil qui permet un large éventail d'approches très différentes pour mener des calculs.

Temps discret – événements discrets Les systèmes thermiques sont en général des systèmes dynamiques, une simulation du régime permanent est insuffisante. La dynamique des phénomènes thermiques au sein d'un système est souvent continue. Deux types d'approches existent pour prendre en compte cette dynamique : simulation par *temps discret* et simulation par *événements discrets*.

Dans le premier cas la simulation passe par une discrétisation du temps. L'horloge des modèles « *avance en segments discrets en passant d'une valeur à la prochaine avec un pas de temps spécifié* » [81]. Dans ce cas, la simulation consiste à calculer l'état du système à chaque pas de temps. Un changement d'état du système est déterminé aux intervalles définis.

La simulation par *événements discrets* utilise un temps continu. La simulation n'est pas avancée par des pas de temps, mais plutôt par une liste d'événements à venir. Cette liste « *contient des instants d'horloge auxquels on assigne aux composants un nouvel état déterminé d'une manière interne* » [81]. L'avantage d'une simulation par événements discrets est une souplesse qui évite la perte de précision d'un pas de temps trop long et la lourdeur d'un pas de temps trop fin. Les inconvénients d'une simulation par temps discret sont surmontés aujourd'hui par des pas de temps variables, gérés pour diminuer l'erreur de calcul. La simulation par événement discret est basée sur deux conditions qui ne sont que rarement vraies pour des systèmes thermiques continus : 1) l'apparition d'un événement peut être déduite à partir de l'apparition d'autres événements. 2) Si l'apparition d'un événement ne peut pas être prévue, les composants ne changent pas d'état, sauf si ce changement d'état est déclenché par le changement d'état d'un composant qui a été programmé [81]. Dans ces conditions, une simulation a lieu avec des pas de temps irréguliers et l'horloge est avancée irrégulièrement à l'instant du prochain événement. Habituellement, on installe un gestionnaire d'événements qui envoie des signaux aux modules concernés.

Dans presque toutes les applications modernes on trouve des dispositifs de régulation qui, quant à eux, se laissent très bien décrire par une représentation

aux états discrets. Une simulation générale de systèmes thermiques devrait permettre un mode mixte. Dans le cadre du présent travail nous nous intéressons à une simulation déterministe, continue et dynamique, par temps discrétisé.

2.2.3 Types de modèle et algorithmes de simulation

Il existe différentes façons d'effectuer une simulation. La méthode à choisir pour la simulation dépend fortement des types de modèles qui sont utilisés pour décrire le système, ses composants et ses liens intérieurs.

- Un *modèle de connaissance* est une construction formelle déduite d'une théorie. Le plus souvent, et en particulier pour ce qui concerne la thermique, cette construction utilise une formulation mathématique. Le langage mathématique est un des moyens pour une description abstraite de la réalité.

  La modélisation des problèmes continus dans l'espace, dans le temps et/ou d'autres dimensions se fait généralement à l'aide d'équations différentielles partielles. On parle des champs qui sont décrits par des *paramètres distribués*. Un flux d'énergie ou de matière peut apparaître dans toute direction.

  Si l'espace est discrétisé ou si le problème ne dépend pas des paramètres continus dans l'espace, on peut représenter le système par des équations différentielles ordinaires. Les paramètres sont concentrés localement (*lumped parameters*). C'est le cas des système tels que l'équipement de chauffage ou de climatisation dans un bâtiment. Ici, le flux d'énergie ou de matière est restreint à certaines voies discrètes.

  Si à la fois l'espace et le temps sont discrets (ou discrétisés), on obtient des équations algébriques orientées par bloc. C'est le seul type d'équation qui puisse être résolu sur un ordinateur. Tout autre type d'équation doit être transformé en équation algébrique pour pouvoir le résoudre sur ordinateur. Ce type d'équation est surtout obtenu par une discrétisation de l'espace. Différentes techniques de discrétisation spatiale existent comme les différences finies, les formulations variationnelles, les éléments finis, ou les volumes finis.

  Associés à ces types de modèles (paramètres distribués, paramètres concentrés, paramètres par bloc) sont des composants typiques de système. Un milieu continu est décrit par un système d'équations différentielles partielles ou les paramètres sont les propriétés locales du problème. Cela est le cas, par exemple, de l'équation différentielle instationnaire pour la conduction, construite à partir du bilan thermodynamique d'un élément différentiel :

  $$\rho A(x) c_p \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}(\lambda A(x) \frac{\partial T}{\partial x})$$

  La conductivité λ, la densité ρ et la surface A sont des paramètres continus qui peuvent changer de valeur en fonction de la variable libre x.

Le comportement des éléments d'un circuit (électrique, thermique, hydraulique, ...) est bien représenté par des équations différentielles ordinaires ou même algébriques. Ses paramètres sont des attributs, des grandeurs spécifiques de l'élément.

- Une autre catégorie de modèle est l'ensemble des *modèles empiriques.* Ces modèles s'expriment de la même façon que les précédents, mais ils sont construits différemment, en simplifiant les équations physiques et en ajustant les paramètres empiriquement. Pour la modélisation de la convection naturelle simplifiée, on peut exprimer le débit traversant la facette séparant deux cubes par :

$$\dot{m} = Sk\rho\Delta p^n$$

où k et n prennent des valeurs empiriques pour décrire la perméabilité de la facette et le type de l'écoulement.

- Les modèles *paramètriques* sont obtenus en construisant des équations qui approchent le mieux possible l'expérimentation. Les paramètres sont ensuite calibrés par identification. Ce type de modèle n'aide pas à expliquer les phénomènes physiques, car les modèles sont créés d'une manière inductive par rapport aux modèles de connaissance dont la construction est déductive.

Nous allons maintenant présenter une évaluation des environnements de simulation existants et estimer leur capacités à couvrir les différentes phases d'une simulation.

2.3 Quelques méthodes et environnements de simulation actuels

Pour la simulation numérique, les équations constitutives des modèles sont traduites en un programme d'ordinateur grâce à des algorithmes numériques. Ces dernières deux décennies ont vu apparaître de nombreux programmes de ce type ; quelques uns sont présentés dans les paragraphes suivants.

Nous voulons distinguer deux types différents d'environnements de simulation. Des environnements avec un traitement direct et des environnements avec une génération intermédiaire de code.

- Le premier groupe contient les environnements globaux ; ils travaillent directement sur une entrée qui en général contient la description du problème. Les calculs de simulation sont immédiatement effectués par le même programme. Nous appelons ces environnements des traitements « à une étape ».

- Il existe ensuite les environnements qui procèdent en plusieurs étapes. Ils traitent d'abord la description du problème. À l'aide de cette description, ils génèrent ensuite un programme de simulation approprié. C'est ce programme généré qui exécute effectivement la simulation. Nous les qualifions de « multi-étape ».

Par rapport à la première approche, la deuxième voie nécessite une étape supplémentaire de compilation. Mais généralement elle génère des simulateurs plus efficaces grâce au choix de méthodes numériques mieux adaptées au problème particulier. Ils essaient de remplacer le travail de l'analyste numéricien. Souvent, les environnements multi-étapes sont plus modulaires et mieux adaptés aux phases de la simulation.

Nous avons testé les logiciels de simulation TRNSYS, TUTSIM, Neptunix, Spark et Zoom. Dans les paragraphes suivants, nous allons les présenter brièvement. Une évaluation de leurs avantages et inconvénients sera donnée pour ce qui concerne la description du système, la possibilité d'étendre et alimenter la bibliothèque de modèles, et finalement la pertinence des résultats obtenus. TRNSYS, TUTSIM et Zoom appartiennent au premier groupe des simulateurs à une étape. le deuxième groupe des environnements avec génération du code est composé de Neptunix et Spark.

2.3.1 TRNSYS

L'environnement de simulation le plus connu et utilisé au niveau international dans le domaine du bâtiment est probablement TRNSYS [73]. Le programme TRNSYS fut développé par le Solar Energy Laboratory de l'Université du Wisconsin, USA en 1975. Fin 1990, la version 13.1 était disponible.

TRNSYS définit un système comme un ensemble de composants connectés entre eux. À l'origine il était conçu pour la simulation des systèmes d'équipement de bâtiment (HVAC), des systèmes solaires. Il est aujourd'hui disponible dans une version généralisée pour la simulation d'un bâtiment complet.

Le programme principal contient une collection de 75 composants pré-définis (par exemple capteurs solaires, radiateurs, bâtiments multizones, etc.) dont la représentation interne est documentée sous forme écrite, dans un classeur. Dans le fichier d'entrée (appelé TRNSYS desk) l'utilisateur indique les composants qu'il veut utiliser pour simuler son système. Pour chaque composant, il spécifie les paramètres (par exemple la valeur d'une capacité calorifique) et les connexions entre les composants. Ces connexions déterminent l'ordre de calcul. Pour stocker les résultats, il y a la possibilité de connecter la sortie d'un composant avec une «imprimante» virtuelle qui place dans un fichier l'évolution de cette variable pendant la simulation. Il faut que chaque sortie d'un composant soit connectée avec une entrée d'un autre composant. Il y a des composant qui lisent un fichier et qui alimentent le système en données externes (par exemple les données météorologiques).

Les modèles associés aux composants et disponibles dans la bibliothèque sont désignés sous le nom de types. Il y a un sous-programme pour chaque type avec une structure fixe de paramètres. Ce sous-programme est appelé pour chacun des composants de ce type à chaque pas de temps, et éventuellement à chaque pas d'itération (pour les composants contenant des équation différentielles).

Extension du système

Si le choix des composants pré-définis ne suffit pas pour un problème donné, l'utilisateur peut écrire ses propres modules de comportement sous la forme d'un sous-programme FORTRAN. Puisque l'interface entre les composants et le programme principal est bien spécifié [73] et les calculs d'intégration sont presque indépendants des calculs dans les sous-programmes, on peut les programmer sans trop de problèmes numériques. Cependant l'utilisateur doit savoir programmer en FORTRAN et connaître la structure des données interne de TRNSYS.

Méthode numérique

Dans la boucle principale où TRNSYS appelle tous les composants pour un pas de temps, TRNSYS peut aussi intégrer une équation différentielle dans les sous-programmes. La méthode d'*Euler modifiée*[2] est utilisée pour calculer un nouveau vecteur d'entrées en fonction des valeurs des dérivées par rapport au temps.

La routine qui interprète le fichier d'entrée classe les composants dans un certain ordre. Au début sont placés les composants qui ont des variables dépendantes du temps et nécessitent l'intégration numérique. Puis suivent les composants qui n'ont pas de dépendance du temps (dont les relations sont résolues par une méthode de *Newton-Raphson*[3]) et finalement les composants qui écrivent les résultats dans un fichier (output units). Ceci est en fait un algorithme de résolution des systèmes d'équations différentielles ordinaires avec équations algébriques.

TRNSYS effectue l'intégration numérique seulement pour les composants qui sont marqués parmi les fonction dépendantes du temps dans la description du fichier d'entrée. Sur l'ensemble des modules, TRNSYS opère une relaxation (avec des options d'accélération ou ralentissement en cas de difficultés numériques).

Évaluation

Un des grands avantages (mais aussi inconvénients) de TRNSYS est son âge. Dans le temps, a été développée toute une panoplie de modèles pour des usages

2. voir annexe D.2 pour une description des méthodes de résolution numérique des équations différentielles ordinaires.

3. les méthodes de résolution de systèmes d'équations non-linéaires se trouve dans l'annexe D.1.

très différents. Un club d'utilisateurs assiste le débutant et lui donne accès à une grande bibliothèque de modèles non-standard, ainsi qu'aux dernières modifications. Le programme, écrit en FORTRAN66, est très portable et « tourne » sur presque toutes sortes d'ordinateurs. L'utilisateur type est le thermicien qui ne veut pas développer ses propres modèles, mais plutôt utiliser ce qui existe. En général, il n'est pas nécessaire pour l'utilisateur de s'occuper des équations internes ou des problèmes numériques.

Un point faible de TRNSYS est certainement l'interaction avec l'utilisateur. L'entrée de données est très pénible suite à une écriture désuète. L'utilisateur doit scrupuleusement respecter l'ordre et l'emplacement des chiffres qui décrivent les liaisons et les paramètres. Des développements récents comme CSTBât essayent de faciliter cette lourde tâche. Par ailleurs, les modèles TRNSYS sont conçus d'une manière fixe pour ce qui concerne le nombre et la qualité des paramètres et des entrées/sorties. Si une variable qui existe déjà comme sortie doit devenir une entrée, l'utilisateur doit écrire une nouvelle sub-routine FORTRAN pour effectuer cette nouvelle mission.

2.3.2 Bondgraphs / TUTSIM

La méthode des « bondgraphs » contient une approche *top-down* pour la simulation des systèmes. Son application suppose d'abord une description globale du système, et ensuite elle examine les détails des composants constitutifs.

Le système des « bondgraphs » est plutôt une approche pour la compréhension d'un système technique quelconque qu'un environnement de simulation au sens strict. Les trois principaux domaines d'application sont l'électricité, la mécanique et l'hydraulique. Son usage n'est pas très répandu dans le domaine de la thermique, car on doit alors gérer des flux d'entropie, ce qui n'est pas très naturel.

Les bondgraphs sont une représentation « par réseau » d'un système. Le réseau comporte les éléments constitutifs du système et les liaisons qui les relient. Plus particulièrement les bondgraphs sont une représentation topologique des débits énergétiques entre les éléments discrets d'un système. Pour cette schématisation on utilise un graphisme symbolique avec :

- des signes (lettres) représentant les composants du système,
- des liens orientés entre les composants appelés « bonds », et
- des connexions entre des liens (voir les *triports* plus bas)

Plusieurs composants peuvent être regroupés dans un seul bloc. Le système, une fois schématisé par les bondgraphs, se traduit facilement en un système d'équations différentielles ordinaires et ensuite en un programme de simulation (voir plus bas pour les supports sur ordinateur). La schématisation par graphes constitue une phase initiale avant l'étude quantitative du système. Elle aide à la

construction du système d'équations ; aucune méthode numérique n'est spécifiée par les Bondgraphs.

Par exemple, la schématisation simple d'une voiture contient le moteur, la boîte de vitesse et les roues. Leurs relations peuvent être schématisées par des bondgraphs comme dans la figure 2.4.

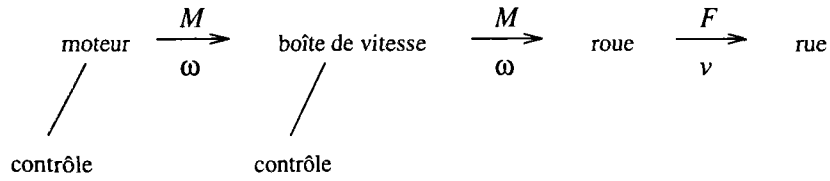


FIG. 2.4 - *Schématisation simple d'une voiture.*

Chaque mot représente ici un composant principal ou un sous-groupe. Les flèches entre les mots sont les « bonds » qui indiquent une « puissance directionnée ». Les traits droits signifient un flux d'information (par exemple le contrôle du moteur par l'accélérateur).

Les bonds contiennent deux variables principales. Elles sont appelées *effort* et *flux*. Leur produit est en général une puissance. Sont toujours associées les variables d'*impulsion* et de *déplacement* qui sont des intégrales d'effort et de flux dans le temps. On dessine un bond comme une flèche qui indique la direction d'écoulement de l'effort et du flux. Dans le domaine de l'électricité on utilise naturellement tension U (= *effort*) × courant I (= *flux*); en mécanique sont utilisés force F (*effort*) × vitesse v (*flux*) ou moment M× vitesse angulaire ω comme dans la fig. 2.4. En thermique les équivalences ne sont pas immédiates : température T (= *effort*) × flux d'entropie $\dot{S}$ (= *flux*). Bien que satisfaisant intellectuellement, le travail avec un flux d'entropie n'est pas naturel pour un thermicien. On peut aussi utiliser la température et le flux de chaleur s'il n'y a pas de conversion d'énergie thermique / mécanique dans le système.

En ce qui concerne les composants, il y a trois classes standards que l'on distingue par le nombre de liaisons possibles : monoport, diport, tri- ou multiport, qui ont respectivement un, deux, trois ou plusieurs points de connexion. Un composant est plus qu'une simple relation mathématique. Il a trois niveaux de description:

- équation constitutive
- relation d'énergie et de puissance
- réversibilité ou non (au sens du 2^{e} principe de la thermodynamique)

Pour les composants monoports on utilise une forme généralisée des composants électriques : R (*résistance*, pour les relations de proportionnalité), C (*capacité*, pour les relations différentielles) et I (*inertie*, pour les relations intégrantes). Des sources d'effort (SE) et des sources de flux (SF) sont également des composants monoports. Chacun de ces composants a exactement une seule

possibilité de raccordement (voir fig. 2.5 pour l'exemple d'un bondgraph représentant un mur bicouche).

À titre d'exemple, regardons les trois niveaux de description pour un élément C: l'équation constitutive relie la charge et la tension (électrique) (soit en général l'*effort* et le *déplacement*) et est indépendante du temps. Les éléments C conservent l'énergie, mais pas la puissance. La puissance est absorbée en chargeant l'élément qui stocke l'énergie. Cela change l'état de l'élément. En revenant à l'ancien état l'énergie est rendue à travers le *bond*. Les éléments C sont réversibles.

Il y a seulement deux composant avec deux liaisons: les transformateurs (TR) et les gyrateurs (GY). Les transformateurs gardent la proportionnalité entre les efforts entrants et sortants (transformateur électrique, boîte de vitesse). Un gyrateur indique une proportionnalité entre l'effort entrant et le flux sortant (ou flux entrant et effort sortant). Le plus souvent il est utilisé pour un transfert de puissance entre deux domaines physique. par exemple un moteur électrique dont la rotation (flux sortant) est proportionnelle à la tension d'entrée (effort entrant).

Les triports sont des jonctions entre plusieurs bonds. Les jonctions parallèles (appelées p ou 0) ont une égalité de l'effort dans toutes les branches et la somme de flux est nulle. Les jonctions en série (appellées s ou 1) qui ont une égalité du flux dans toutes les branches et la somme d'efforts est zéro (voir fig. 2.5). [77, chap. 2.3].

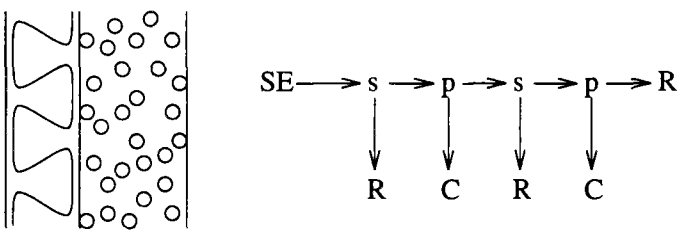


FIG. 2.5 - *Un mur bicouche sollicité par une température à gauche ; chacune des couches représentées par une capacité thermique, séparées par des résistances thermiques.*

Causalité

Un principe fondamental des bondgraphs est la causalité. Par indication de la causalité, on donne un sens aux formules constitutives des composants qui sont *a priori* non-orientées. C'est une phase nécessaire pour passer à la simulation. Un tiret supplémentaire est ajouté, soit au début soit à la fin de chaque bond pour indiquer la « causalité » de l'élément connecté. La causalité indique quelle variable de la liaison est calculée en fonction de l'autre. De cette façon, le sens (l'ordre) des calculs peut être déterminé.

Voyons un exemple pour un élément du type *proportionnalité* dont l'équation constitutive sans indication de la direction de calculs est:

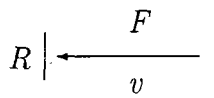
$$\textit{vitesse} = \textit{force} \;/\; \textit{résistance}$$

ou

$$force = résistance \cdot vitesse$$

si on prend un cas d'application en mécanique où le flux est une vitesse et l'effort est une force.

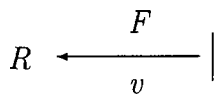
On choisit l'une des deux représentations par le tiret de causalité. Si le tiret est près du composant, le composant calcule le flux en fonction de l'effort (fig. 2.6).



$$v = f(F) = \frac{F}{R}$$

FIG. 2.6 - *Causalité pour calculer le flux en fonction de l'effort*

Si le tiret est de l'autre côté du composant, le composant calcule l'effort en fonction du flux (fig. 2.7).



$$F = f(v) = v \cdot R$$

FIG. 2.7 - *Causalité pour calculer l'effort en fonction du flux*

Quelques causalités sont imposées, sinon le système d'équations n'est pas soluble. C'est évident pour les sources, une source d'effort ne peut pas livrer un flux en fonction de l'effort. Les jonctions déterminent également des causalités. Une jonction parallèle a exactement *un* effort qui entre, les autres sortent ; une jonction en série a *un* flux entrant, sur les autres branches ils sortent. Le choix de la causalité est quelque fois arbitraire. Pour des raisons numériques, on choisit de préférence une causalité intégrante et on évite les causalités qui entraînent des dérivations numériques.

La méthode de bondgraph aide à la description des systèmes. Elle sert à déterminer les interdépendances entre les équations constitutives et pour trouver un bon ordre de calcul, cohérent avec le principe de causalité.

Pour l'étude d'un système et la description de la simulation on dessine d'abord un bondgraph du système à examiner avec les symboles définis (composants et bonds). Il existe ensuite plusieurs logiciels qui permettent de saisir directement une description formelle d'un bondgraph (voir fig. 2.8) et qui exécutent la simulation : TUTSIM (voir en bas) éventuellement avec FANSIM [35], ENPORT [69] et le couple CAMP/ACSL [18].

TUTSIM

Parmi les logiciels d'application de la méthode des bondgraphs, nous avons choisi TUTSIM et sa version pour PC [57] en raison de sa disponibilité et sa facilité d'utilisation.

L'utilisateur de ce programme décrit les éléments de son problème dans un ordre quelconque à l'aide d'un programme de saisie spécial. Non seulement les

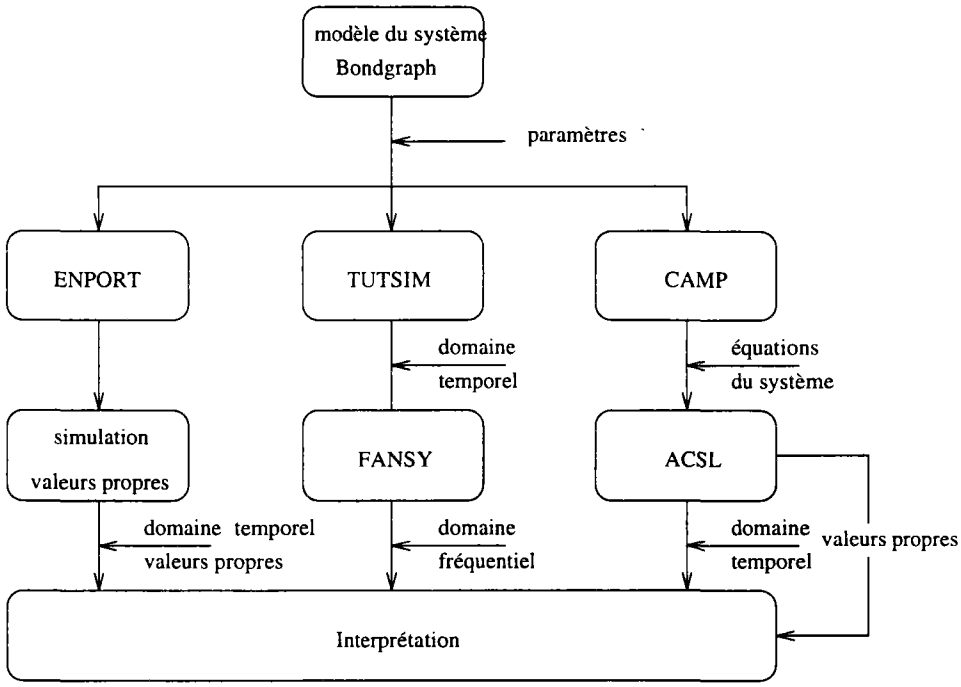


FIG. 2.8 - *Une carte de logiciels pour obtenir des résultats d'un bondgraph.*

éléments de base des bondgraphs sont permis, mais aussi un grand nombre d'autres composants qui sont apparemment des restes du temps des ordinateurs analogiques. Après la saisie du problème, TUTSIM fait des contrôles de calculabilité, et il simplifie le graphe des connexions.

L'exécution effective de la simulation passe par un algorithme d'*Euler* et TUTSIM crée un fichier de résultats pour un postprocesseur. Mais un des avantages de TUTSIM est l'affichage direct des résultats sur un écran graphique. L'utilisateur peut temporairement changer les paramètres des équations. Ceci rend le programme très souple et interactif, car on peut observer et on obtient directement et en temps réel les conséquences d'une modification du système sur les résultats.

Évaluation

La méthode des bondgraphs fournit une description pertinente des systèmes continus par une représentation bien formalisée – des « flux de puissance» entre des composants discrets. L'analyse que l'on est obligée de faire pour dessiner le graphe et décider les causalités permet une meilleure compréhension des liens et dépendances à l'intérieur du système. Cette méthode est un outil de modélisation puissant, c'est à dire du développement d'une représentation mathématique du système. C'est la raison pour laquelle on ne trouve pas de bibliothèque de modèles pré-définis.

Un composant d'un bondgraph est limité à une fonction multiplication, intégration, dérivation, ou une combinaison de celles-ci. Cela permet la représen-

tation de toutes les équations continues. Pour représenter des états discrets et logiques, la méthode est encore mal adaptée. Si une fonction est définie sur différentes plages ou si elle agit en fonction d'un seuil, il faut utiliser des composants non-standards et non-définis dans le concept (mais disponible dans TUTSIM).

Un environnement graphique pour la saisie d'un système n'existe pas encore, bien que le formalisme s'y prête. Par contre, la présentation graphique des résultats et la souplesse dans la modification des paramètres est bien réussie dans le produit TUTSIM.

2.3.3 Neptunix

Neptunix est un environnement de simulation à deux étapes. Il génère un programme solveur pour simuler des systèmes qui sont décrits par un système d'équations algébro-différentielles (SEAD). Son grand avantage est la possibilité de pouvoir traiter des équations non continues par un changement d'état du simulateur.

L'utilisateur doit développer un modèle mathématique de son problème avant l'exploitation de Neptunix, puisque le programme ne fournit aucun support, comme des formules pré-définies par exemple. À partir de la description du modèle mathématique faite dans son propre langage de pilotage [22], Neptunix génère des fichiers source FORTRAN pour construire le programme final de la simulation. On distingue alors deux phase différentes :

- la phase de génération du simulateur à partir de la description mathématique de son comportement,
- la phase de l'exploitation du simulateur ainsi construit.

Les paramètres de la deuxième phase, comme la durée de la simulation et certains autres paramètres des équations, peuvent être contrôlés par un autre langage de pilotage [21].

Puisque Neptunix est plutôt un solveur des SEAD, il n'est pas limité aux problèmes thermiques. Tous les systèmes que l'on peut décrire par des SEAD peuvent être traités par Neptunix.

ALLAN

ALLAN est un interpréteur graphique conçu et développé par Gaz de France [23]. Son objectif est de décharger les ingénieurs d'études de l'analyse numérique et de la programmation informatique liées à l'utilisation des modèles. En fournissant les outils de gestion d'une bibliothèque de composants, de sous-systèmes ou même de systèmes complets, ALLAN est un outil qui facilite la réutilisation des études réalisées précédemment.

ALLAN est essentiellement un préprocesseur pour un langage de simulation. Pour l'instant, des sorties existent pour Neptunix et Astec.

Trois type de composants sont connus par ALLAN :

- des *composants simples*, dont la fonction est décrite par des équations algébriques ou différentielles.
- des *composants composés*. Dans un composant composé sont définis les raccordements entre des sous-composants (simples ou eux-même composés).
- des modèles de simulation. Leur schéma fonctionnel est utilisé pour générer le code de résolution dans le langage Neptunix. Avant cette étape ALLAN contrôle si toutes les initialisations et les paramètres propres à la simulation sont déclarés.

À travers l'interface ALLAN l'utilisateur peut totalement piloter le solveur aussi bien pour la génération du code que pour la simulation et le post-traitement. Des commandes spéciales permettent la visualisation graphique des résultats.

En cas de problèmes numériques ou suite à des disfonctionnements d'ALLAN, ou tout simplement pour plus de souplesse, l'utilisateur peut modifier directement les fichiers générés pour agir sur les équations ou pour guider l'algorithme de résolution.

Évaluation

Le grand avantage de Neptunix est la génération automatique d'un solveur spécialement dédié au problème. Ceci permet d'utiliser des algorithmes optimisés pour le système à traiter. Comme résultat, on obtient un code très efficace et sûr. Un autre point fort est la connexion avec un *automate* pour gérer des discontinuités dans les équations. Ce mode mixte permet la combinaison de fonctions continues, définies par intervalles, et de fonctions discrètes. La définition des équations sous forme implicite évite le problème d'orientation et d'inversion des équations. Suite à la génération de code spécifique, on perd en souplesse si l'on veut modifier le système ou même une seule équation, car un nouveau programme doit être généré et compilé à nouveau, bien que cela se passe automatiquement. Il n'est pas très commode pour les calculs vectoriels et matriciels tels que ceux qui résultent d'une modélisation en différences finies, par exemple.

De même que pour TRNSYS, la définition du système d'équations et la saisie des paramètres sont lourds. Le travail au niveau mathématique ne présente aucun rapport avec le problème physique de départ. Ces inconvénients sont effacés par le préprocesseur ALLAN. Cet outil permet la création d'une bibliothèque de modèles. Une interface graphique conviviale qui existe à la fois pour la saisie de modèles, pour la description d'un système composé et aussi pour l'affichage des résultats, facilite le travail avec Neptunix.

2.3.4 EKS / Spark

Le nom EKS est l'abréviation de « Energy Kernel System », un système général pour la simulation des systèmes énergétiques. À l'intérieur de l'EKS se trouve le Simulation Problem Analysis Research Kernel (Spark)[4] en cours de développement par le Lawrence Berkeley Laboratory et l'University of Fullerton en Californie [16].

Spark est un système de logiciels pour la description et résolution des équations algébro-différentielles non-linéaires. Il est composé de plusieurs logiciels modulaires. L'utilisateur a quatre points d'interaction avec ce système : il peut définir des objets (qui peuvent être élémentaires ou composés et qui sont accessibles dans une bibliothèque), il peut définir un problème à résoudre par assemblage des objets; il doit spécifier des données pour l'exécution (données météorologique, paramètres ...) et il peut spécifier les sorties souhaitées.

Les objets de base sont des équations pour Spark. Chaque équation est un objet élémentaire séparé. À un niveau supérieur existent des *macros* qui rassemblent plusieurs équations ou macros dans un objet. Si une macro fait référence à une sous-macro, on obtient une structure hiérarchisée de description. L'algorithme interne de Spark résout cette hiérarchie en mettant toutes les équation « à plat ».

L'utilisateur pourra décrire les problèmes par connexion d'objets à l'aide d'une interface graphique qui est en cours de développement. Pour l'instant Spark est encore limité à une écriture « à la main » des fichiers nécessaires ; on peut cependant utiliser un préprocesseur (voir plus bas). L'éditeur graphique produit un fichier contenant les spécifications en NSL [3], un langage pour les spécification d'un réseau. Spark détermine un ordre de calcul pour la résolution du système d'équations et en fonction de ces résultats, il génère le programme de simulation adapté au système d'équations. Ce simulateur lit les données et calcule la solution. Un postprocesseur peut lire les résultats et les afficher grâce à une interface graphique.

Algorithmes internes

Spark analyse le fichier de spécifications qui lui est soumis, et traduit de manière interne les équations sous forme de graphe. Une équation est représentée par un nœud, et une variable par un arc. Un arc relie deux nœuds si la variable associée est commune aux deux équations associées.

Ensuite un algorithme de théorie des graphes, dit algorithme de DINIC [4], couple chaque équation à une et une seule de ses inconnues, qui dès lors sera calculée exclusivement par son équation associée. Cela revient à orienter les arcs du graphe, l'arc sortant d'un nœud si il correspond à la variable couplée à l'équation-nœud, et rentrant sinon. Puis un autre algorithme heuristique, dit algorithme de LEVY-LOW [54], détermine un ensemble « de petite taille» d'arcs

4. noyau de recherche de l'analyse du problème de simulation (!)

tels que couper ces arcs élimine tous les cycles du graphe. Cela revient à trouver un « petit » ensemble de variables telles que leur connaissance exacte permet de dériver toutes les autres inconnues de façon explicite. Spark détermine donc un ensemble de variables d'itération du problème, de taille n_{it} inférieure à la taille du problème. Il suffit donc de résoudre le problème non linéaire réduit, avec seulement les variables d'itération comme inconnues. Comme résoudre un problème non-linéaire de taille N passe par l'inversion de la matrice Jacobienne du système (voir plus bas), qui a un coût $O(N^3)$, on réduit le temps calcul d'un facteur $O(r^3)$, ou r est la réduction $\frac{N}{n_{it}}$.

Une fois le problème réduit résolu, la solution est propagée aux inconnues restantes.

Interface symbolique

Si on dispose d'un logiciel de calcul formel tel que Macsyma [59] ou Maple [19], on peut l'utiliser comme préprocesseur pour Spark. L'interface symbolique crée du code qui ensuite peut être utilisé par Spark. La puissance des outils de calcul formel permet d'utiliser directement des représentations non-orientées des objets de base que sont les équations.

Il existe en effet des commandes pour créer toutes les formes explicites d'une équation implicite. L'utilisateur peut spécifier de « mauvaises inverses », c'est à dire une liste des variables de l'équation en lesquelles on ne souhaite pas essayer de résoudre l'équation, soit parce que ce ne sera jamais utile, soit parce qu'il est évident qu'il n'existe pas de solution simple ou intéressante. De manière générale, plus il y a de façons locales de résoudre une équation, plus Spark est capable de trouver une solution efficace du système global.

Par ailleurs existent aussi des commandes pour la création automatique de macro-objets, d'objets dynamiques, de macros-objets dynamiques, la création d'un fichier de simulation, et d'entrée de paramètres.

Évaluation

Spark génère également un programme adapté spécifiquement au problème posé. L'optimisation atteinte par cette génération est une réduction du nombre de variables sur lesquelles l'algorithme de résolution peut itérer. Cela donne à l'utilisateur une plus grande liberté pour exprimer les équations.

Spark est entièrement basé sur des objets sous forme équationnelle, ce qui permet des optimisations. Des comportements avec discontinuités ou hystérèsis peuvent être décrits dans un objet sous forme de conditions dans lesquelles l'équation est applicable. Dans son état actuel, l'algorithme de résolution ne traite pas spécifiquement le problème des équations discontinues.

Bien que l'interface graphique ne soit pas encore disponible, le travail de l'utilisateur est déjà facilité par une bibliothèque d'objets existants et par la facilité qu'il y a à créer de nouveaux objets.

2.3.5 FET / ZOOM

L'environnement ZOOM (Zone Organized Optimal Modelling) et sa base mathématique, le FET (Formalisme d'Évolution par Transfert) sont en cours de développement par le CNRS, à Toulouse et à Orsay [9]. Le but du logiciel ZOOM est la simulation de systèmes complexes dans lesquels les divers processus qui entrent en jeu sont fortement couplés. En plus de la modularité nécessaire à la simulation d'un ensemble de modèles variés, le FET met l'accent sur les couplages entre les évolutions des différentes parties du système. Il est un cadre conceptuel qui permet de décrire le comportement du système dans le temps ou en régime permanent. Un mode mixte avec des parties dynamiques et des parties stationnaires est même possible.

À la différence de beaucoup d'autres environnements, ZOOM utilise une visibilité complète entre les composants et les connexions intérieures du système. L'algorithme interne demande une représentation linéaire (ou linéarisée) du problème.

Le formalisme d'évolution par transfert (FET)

Ce formalisme est développé depuis 10 ans pour la simulation des systèmes thermiques et a abouti à une maquette du logiciel ZOOM qui implémente ce formalisme. Les deux opérations fondamentales décrites dans le paragraphe sur l'approche systémique (§ 2.1.4) sont utilisées aussi ici : le découpage du système en différentes parties et puis le raccordement de ces parties pour retrouver le système global. Un découpage en plusieurs niveaux hiérarchisés est possible. Le raccordement entre les objets à un niveau donné est appelé une famille. Dans cet objectif, le FET utilise deux types d'objets fondamentaux :

- les *cellules* qui représentent les composants du système. On suppose que l'évolution de l'état d'une cellule α dépend de son état η_α et d'un vecteur contenant des information sur le couplage φ de la cellule avec son environnement. Le comportement de la cellule est décrit par l'équation :

$$\frac{\partial \eta_\alpha}{\partial t} = G_\alpha(\eta_\alpha, \varphi_\alpha, t)$$

- les *transferts*, qui représentent des interfaces physiques ou des processus de couplage entre les cellules. Les transferts sont définis comme des relations où la variable d'interface φ est exprimée en fonctions des états des cellules connectées par cette interface et éventuellement des autres transferts φ' connectés à ces mêmes cellules.

$$\varphi = f(\eta, \varphi', t)$$

  Ceci représente les équations de contraintes auxquelles sont soumises les variables de transfert.

Une des différences entre ces deux types d'objets est le fait que le modèle des cellules est un modèle d'évolution au cours du temps avec des variables d'état, alors que le modèle des transferts est une contrainte instantanée dans les interfaces. Les cellules possèdent une inertie physique qui fait dépendre leur évolution temporelle non seulement des transferts mutuels mais aussi de leur histoire propre. Quant à l'interface, c'est un phénomène sans épaisseur ; son état dépend instantanément de son environnement.

Pour calculer l'évolution du système construit par les deux types d'équations ci-dessus, on passe par une discrétisation du temps et une *linéarisation* de l'équation des cellules. Ceci donne

$$A_{\alpha\alpha}\delta\eta_\alpha + B_{\alpha\varphi}\delta\varphi = \Gamma_\alpha \delta t$$

$$\delta t \sum_\alpha C^+_{\varphi\alpha}\delta\eta_\alpha - (1 + D_{\varphi\varphi})\delta\varphi = \Omega_\varphi \delta t$$

où α représente l'ensemble de toutes les cellules du modèle et $\delta\varphi$ l'évolution de toutes les variables de transfert au cours du pas de temps. La matrice C^+ décrit l'influence des cellules sur un transfert. La résolution dans le cadre du FET passe par l'élimination des variables d'état $\delta\eta_\alpha$ et on peut écrire

$$\delta\eta_\alpha = \delta\eta_{\alpha,\mathrm{dec}} + \mathcal{F}_{\alpha\varphi}\delta\varphi$$

$$(1 + D_{\varphi\varphi} - \delta t \sum_\alpha C^+_{\varphi\alpha}\mathcal{F}_{\alpha\varphi})\delta\varphi = \delta t(\sum_\alpha C^+_{\varphi\alpha}\delta\eta_{\alpha,\mathrm{dec}} - \Omega_\varphi)$$

Ici $\delta\eta_{\alpha,\mathrm{dec}} = A^{-1}_{\alpha\alpha}\Gamma_\alpha\delta t$ représente l'évolution spontanée qu'aurait la cellule si elle était découplée de son environnement (c'est à dire de ces transferts). La matrice $\mathcal{F}_{\alpha\varphi} = A^{-1}_{\alpha\alpha}B_{\alpha\varphi}$ décrit l'influence d'une évolution des transferts sur l'évolution de la cellule α. L'influence directe des autres transferts sur un transfert est contenue dans la matrice $D_{\varphi\varphi}$.

Par élimination des variables d'état on obtient ensuite l'équation-clef:

$$(1 + D_{\varphi\varphi} - \delta t \sum_\alpha C^+_{\varphi\alpha}\mathcal{F}_{\alpha\varphi})\delta\varphi = \delta\varphi_{\mathrm{ins}}$$

avec laquelle on peut calculer l'évolution $\delta\varphi$ des variables de transfert au cours du pas de temps δt. La variable $\varphi_{\mathrm{ins}} = \delta t(\sum_\alpha C^+_{\varphi\alpha}\delta\eta_{\alpha,\mathrm{dec}} - \Omega_\varphi)$ est l'évolution insensible des transferts. Elle indique la variation qu'auraient les transferts si les cellules suivaient leur évolution découplée. L'expression entre parenthèses à gauche de l'équation est appelée la matrice de couplage. L'analyse de cette matrice permet de repérer les transferts que l'on veut plus particulièrement observer.

Un autre aspect de ce schéma est la réduction du nombre des variables. Cette réduction se fait sélectivement de manière à faire ressortir les transferts que l'on veut analyser. La méthode consiste à regrouper les cellules de la partition initiale du système en différents sous-systèmes qui sont les *familles*. Ce regroupement en familles emboîtées donne une structure arborescente du système (cf. 4.2).

La procédure de résolution, appelée la *Navette*, consiste à parcourir la structure hiérarchique des familles et des cellules afin d'éliminer les variables internes

des familles qui se trouvent au niveau le plus bas, jusqu'au calcul des transferts qui se trouvent tout en haut de la structure. À chaque étape d'élimination, la matrice de couplage est modifiée et le vecteur de variables considérées est réduit. On substitue ensuite les valeurs obtenues par l'équations ci-dessus en redescendant jusqu'au niveau des cellules.

On peut trouver une description plus détaillée de la technique du formalisme FET dans [83], [84], et [82].

ZOOM

L'environnement logiciel qui implémente ce formalisme de résolution est appelé ZOOM. Sa structure a été développée pour rendre le FET utilisable en l'appliquant aux problèmes thermiques, dans un objectif de recherche.

Pour les deux types fondamentaux d'objets, il y a dans l'implémentation informatique ZOOM deux types de processeurs génériques correspondants : les *processeurs d'objets* et les *processeurs de transferts*. Les *processeurs d'objets* calculent l'équation

$$\delta\eta = \delta\eta_{\text{dec}} + \mathcal{F}\delta\varphi$$

en fonction du pas de temps δt, l'état η_0 et les transferts φ_0 au début du pas de temps. Les résultats sont $\delta\eta_{\text{dec}}$ qui décrit l'évolution de la cellule si elle était découplée de son environnement et la matrice $\mathcal{F}$ qui représente l'influence des transferts sur la cellule.

Les processeurs de transferts prennent comme entrées les mêmes valeurs que les processeurs de cellules : δt,η_0, et φ_0. Ils traitent l'équation

$$\varphi = f(\{\eta_{\alpha 0}\}, \{\varphi_0\})$$

Ses résultats sont les valeurs de transferts φ, la matrice $\boldsymbol{C}^+$ qui décrit l'influence des cellules sur le transfert et la matrice $\boldsymbol{D}$ qui décrit l'influence directe des autres transferts couplés. Il est possible d'utiliser un pas de temps variable que l'utilisateur peut spécifier. Des développements sont en cours pour permettre aussi des pas de temps non-homogènes dans les différentes parties du système. Ceci donne la possibilité de séparer les objets en familles pour profiter des calculs adaptés à des constantes de temps spécifiques.

Il existe divers langages descriptifs (voir fig. 2.9) associés aux différents niveaux d'abstraction de l'approche FET[5]. Pour l'instant, il y a six langages plus ou moins rattachés à trois niveaux d'abstraction.

La structure du système, c'est à dire le partitionnement et raccordement, sera décrite par le ZDL (Zoom Design Language). Les processeurs correspondant à un modèle physique sont décrits par le PDL (Processor Design Language). L'utilisateur ne devrait pas avoir de contact avec les autres langages qui sont plutôt des utilitaires pour un traitement mathématique (HMB, constructeur

5. Il n'est pas étonnant de retrouver ici une approche et notation similaire de l'approche SYMBOL, car les deux projets ont bénéficié des développements respectifs de l'autre.

physique	ZDL	PDL	
mathématique		HMB	
informatique		ZBM	ZTM
		ALLIS	

FIG. 2.9 - *les langages de ZOOM aux différents niveaux d'abstraction*

de matrices Hessiennes) ou pour l'aide à la programmation (ALLIS, Ada like language, ZBM, Zoom Bank Manager). L'ensemble de l'environnement ZOOM est implémenté en FORTRAN sur des machines UNIX. Un gestionnaire de base de données (ZEBRA) est par ailleurs mis à profit pour l'environnement ZOOM.

Plusieurs exemples ont été traités et comparés avec d'autres environnements de simulation. Les résultats montrent une fiabilité au niveau de la précision des résultats obtenus et une souplesse au niveau de l'interaction avec l'utilisateur.

Évaluation

La description des modèles d'une part et la description d'un système concret sont bien formalisées et structurées dans le cadre du FET et ZOOM. Des langages spécifiquement développés aident l'utilisateur à effectuer cette tâche, quelque fois un peu lourde. On trouve aujourd'hui une bonne bibliothèque de modèles existants.

La documentation sur cette approche particulière est une grande collection de brochures et d'articles. Comme l'idée principale de cette méthode est un peu compliquée, des cours de formation sont offerts par les laboratoires impliqués dans le projet.

L'environnement de logiciels ZOOM est très souple en ce qui concerne l'interaction avec et les modifications faites par l'utilisateur. ZOOM est strictement hiérarchisé, et cela pour deux raisons. D'une part, le système est décrit par découpage en morceaux de plus en plus petits qui donnent une structure d'arbre pour la représentation. D'autre part, il existe une distinction entre différentes couches d'abstraction pour la description des modèles. Ceci permet à l'utilisateur de travailler avec des notations qui lui sont plus familières comme une pompe, un tuyau, plutôt que des équations mathématiques.

Depuis quelques temps on peut effectuer une simulation à pas de temps variable ou la durée du pas est spécifiée par l'utilisateur.

2.3.6 Quelques autres environnements

Il existe bien d'autres environnements qui se situent dans le domaine de la thermique que nous allons brièvement mentionner :

ACSL est un produit qui existe maintenant depuis une bonne vingtaine d'années. Similaire à Neptunix, ACSL accepte un langage de description pour

ensuite générer un programme FORTRAN qui effectue la simulation. Ce langage est basé sur une définition de langages de simulation. Le sigle ACSL veut d'ailleurs dire Advanced Continuous Simulation Language.

ACSL est particulièrement adapté aux systèmes continus avec paramètres concentrés (*lumped parameters*). Mais il n'a pas l'avantage de Neptunix qui gère les discontinuités.

IDA (ou anciennement MODSIM) est un environnement de modélisation graphique et interactif qui est en cours de développement par le *Swedish Institut of Applied Mathematics* [71] [70]. Le noyau central de IDA est un solveur numérique qui est basé sur l'algorithme de GEAR sous sa forme DASSL [63]. Ce solveur peut traiter de grands systèmes d'équations algébriques et différentielles non-orientées. Pendant la simulation un pas de temps variable est utilisé ; il prend en compte les discontinuités et les états discrets ainsi que les phénomènes d'hystérèsis. Le format pour la représentation de modèles sur ordinateur est le *Neutral Model Format* [74], développé en coopération avec l'équipe Spark, où il peut être utilisé comme un préprocesseur supplémentaire pour la description de modèle (voir § 2.3.4 en haut). Le NMF permet une description des modèles indépendants des environnements. L'interface graphique de IDA facilite la saisie des connexions [15]. La compatibilité entre les types de variables est vérifiée et la hiérarchie de description est mise à plat.

CLIM 2000 est développé par *Électricité De France* depuis 1985 [40]. Conçu comme un outil de recherche, il est destiné à une large gamme d'applications concrètes comme l'évaluation énergétique et financière des projets, l'analyse de confort, le développement de système de contrôle. Pour cet objectif on utilise le concept d'une stricte séparation entre une bibliothèque de modèles et un solveur standard pour les calculs numériques (ASTEC [46]). Des études pour utiliser un autre solveur sont en cours.

Un élément de la bibliothèque comprend toutes les informations nécessaires pour son traitement et son installation. Il est décrit à l'aide d'un PROFORMA (voir §. 7.3.1). Un composant se présente sur l'écran graphique à l'utilisateur sous la forme d'une icône avec des pattes de connexion. Depuis 1989, le logiciel est opérationnel et on cherche à alimenter la bibliothèque. Dans ce but, a été mis en place une méthode de modélisation basée sur la distinction entre trois points de vue qui sont ceux d'un *utilisateur*, d'un *modélisateur*, et d'un *développeur de logiciels* [68].

2.4 Problèmes courants

La complexité des problème étudiés implique aussi une certaine complexité au niveau de la simulation. C'est pourquoi on a pu constater en thermique du bâtiment le développement de programmes volumineux. Les premières approches ont créé des programmes statiques d'une taille importante, souvent

limités à l'étude d'aspects spéciaux. Des problèmes qui étaient proches ou similaires n'étaient pas pris en compte pendant le développement du code initial, et demandaient chaque fois de nouveaux développements. L'inconvénient de cette approche est le manque de souplesse du code. Une modification localisée entraîne d'autres modifications éparpillées dans le programme. Le programme est difficile à corriger en cas d'erreur.

Bien que résolus dans certains environnement de simulation, quelques problèmes restent d'actualité :

- un premier point est la description pertinente non ambigüe de modèles. Ceci concerne aussi bien la définition d'un modèle élémentaire et composé, que sa représentation informatique des modèles. C'est un problème qui s'aggrave encore, lorsque l'on veut construire des bibliothèques de modèles et si l'on veut permettre l'échange de modèles entre différentes applications.

  Il existe des propositions pour les deux aspects de ce problème. Une première initiative consiste à normaliser la description avec les PROFORMAS [28]. Nous y revenons dans le chapitre 7.3.1. La deuxième approche est plus liée à l'exploitation informatique. Un format uniforme peut être le Neutral Model Format (NMF), qui est proposé par les développeurs du simulateur IDA.

- un autre problème qui est partiellement lié au point précédent, est la saisie du système sur ordinateur. La saisie comprend la structure du système et les paramètres. Tous les laboratoires qui ont développés les environnements recensés, développent finalement aussi une interface graphique pour faciliter la saisie. Exceptés TUTSIM et le préprocesseur ALLAN qui sont des produits commercialisés, aucun environnement ne supporte encore cette interaction graphique avec l'utilisateur.

- On peut constater des progrès constants de la qualité numérique de la résolution des grands systèmes d'équations (différentielles ou non). Mais la convergence des calculs de simulation n'est toujours pas assurée dans tous les cas. La gestion des discontinuités dans les équations et un mode mixte des simulation *temps discret – événements discrets* reste un des grands problèmes numériques.

- Bien que la puissance des ordinateurs ait atteint un niveau considérable pour des prix tout à fait raisonnables, la durée des simulations peut être trop longue. On peut diminuer le temps de simulation en faisant travailler plusieurs processeurs sur un même problème. Les environnements de simulations traités ici n'utilisent pas (encore) un possible parallélisme dans l'architecture des machine ou dans un réseau d'ordinateurs interconnectés.

- Un point intéressant pour la simulation est une évolution de la structure au cour de la simulation. Comment peut-on représenter de nouvelles

connexions qui s'établissent dans le temps? Comment peut-on représenter l'état d'un module d'une manière neutre pour qu'au prochain pas de temps ce module puisse être simulé par un modèle différent?

2.5 Le simulateur du projet SYMBOL : Motor-2

À l'intérieur du groupe GISE a été développé le projet SYMBOL pour notre propre environnement de simulation. Cet environnement reprend quelques idées déjà présentes dans les environnements présentés en haut. Mais nous nous centrons encore plus sur l'aspect de la réutilisabilité des parties de travaux effectués pour des études précédentes. Cela veut dire que les efforts mis dans la compréhension des phénomènes physiques, la modélisation, l'implémentation informatique ne sont pas perdus, mais peuvent éventuellement être repris dans une nouvelle étude d'un système thermique. Une méthode a été mise au point pour systématiser la description tant des objets réels que des modèles utilisés et de la représentation sur ordinateur. Elle nous permet de bien distinguer les différentes phases et niveaux d'une étude de comportement thermique. Les aspects de modularité pour des raisons de flexibilité et souplesse ont été prépondérants dans la conception du projet.

Parmi d'autres utilitaires de l'environnement SYMBOL, se trouve le programme de simulation Motor-2. Deux points importants le distinguent des simulateurs habituels. Une description hiérarchisée du système est maintenue pour la simulation. On ne met pas à plat toutes les connexion, mais on garde un partitionnement du découpage pour la résolution numérique. D'autre part, des mesures sont prises pour profiter du parallélisme éventuel, d'une part de l'architecture des ordinateur ou d'autre part, d'un réseau d'ordinateur pour diminuer les temps de calculs.

Dans le chapitre suivant nous allons expliquer et présenter plus en détail les idées de base du projet SYMBOL et du simulateur Motor-2.