

Chapitre 5

Modules et Raccordement : les deux entités majeures de Motor-2

Nous avons vu dans les chapitres précédents les approches de l'environnement SYMBOL et plus particulièrement Motor-2 pour diminuer la complexité apparente pour l'utilisateur. Le *module* est l'entité de base que l'on obtient par le découpage. Chaque composant du système est représenté par un module dans la description du problème que l'on veut simuler et qui est nécessaire pour reconstruire le système original.

La deuxième notion nécessaire pour la reconstruction du système est le *raccordement*. Il décrit les liens et les dépendances entre les modules pour conserver la cohérence de l'ensemble.

5.1 La notion de module

Nous rappelons brièvement notre terminologie en ce qui concerne la description d'un système. Un *module* est une unité « facilement identifiable »¹ à l'intérieur d'un ensemble ; plus particulièrement, un module est un composant de notre système. La combinaison ou juxtaposition ou mieux le raccordement des modules permet la construction d'un système composé.

À chaque étape du découpage d'un système (voir 4.2), nous appelons les composants obtenus des *modules*. Cela crée deux types de modules. Nous distinguons des *modules élémentaires* qui sont les modules terminaux du découpage (*les feuilles*) et des *modules composés* qui sont des modules de raccordement (*les ramifications*). Un module composé est un ensemble de plusieurs sous-modules et de liens de raccordement qui gèrent les informations sur la connexion.

1. par *facilement identifiable* nous indiquons que l'unité peut être facilement repérée et délimitée dans son environnement. Cette identification n'est pas toujours « facile », et n'est pas une démarche absolue.

Si nous revenons aux deux phases d'une étude de système qui sont la modélisation et la simulation, nous employons le terme *modèle* dans la première phase (construction de modèles) et le terme *module* dans la deuxième phase (un système est composé de plusieurs modules). Lors de la description de notre système réel, nous associons à chaque module un certain type, un *modèle de représentation*. Ceci s'applique surtout pour les modules élémentaires. On dit par exemple que le module du MUR NORD est représenté par un modèle de DIFFÉRENCES FINIES 1D. Pour les modules composés qui sont situés aux niveaux supérieurs de l'arbre de découpage, nous disons tout simplement que le module est du type COMPOSÉ.

Nous avons appliqué aux modules les règles qui traduisent une visibilité limitée. Les deux points de vue d'un objet (voir 4.3) s'appliquent également au module. La vue extérieure spécifie la partie visible d'un module, la vue intérieure contient son implémentation. C'est la vue intérieure qui établit le lien entre un module et son type, le modèle associé. Il doit y avoir aussi cohérence entre les spécifications, la partie visible, d'un module et les entrées/sorties des spécifications du modèle associé. C'est à dire, que l'on ne peut pas associer un modèle qui nécessite trois entrées à un module qui n'a que deux entrées, par exemple.

5.1.1 Vue extérieure

Comme les modules doivent échanger des informations, on est alors conduit à définir des *frontières* de communication. Dans le cas d'un mur par exemple on peut définir les faces gauche et droite comme frontières ; le champ de température à l'intérieur du mur n'est pas vu de l'extérieur. Toute communication avec le module MUR passe par ces frontières GAUCHE et DROITE. Son état interne n'est pas directement visible.

En général, une frontière regroupe plusieurs variables ou ce que nous appelons des *pattes* (*pins*). Ces pattes sont orientées, c'est à dire que les variables sont désignées comme variables d'entrée ou de sortie pour le module. Dans une représentation graphique, nous utilisons des ellipses qui signifient des frontières pour en souligner la nature composée. De plus, nous associons un type à chaque frontière. La face gauche du mur comprend une température et un flux de chaleur. Cette frontière peut être du type DIRICHLET, par exemple. Le type d'une frontière nous donne la possibilité de vérifier la description des connexions. Nous pouvons par exemple refuser le raccordement d'une frontière de type DIRICHLET avec une frontière du type CONVECTION VERTICALE.

L'avantage de la notion d'une *frontière* est de pouvoir remonter à un niveau plus abstrait pour définir complètement la partie visible d'un module. Nous pouvons ainsi rester aux niveaux technique / physique dans la phase descriptive du système, car le raccordement entre modules peut s'exprimer maintenant en termes techniques, plutôt que par une expression mathématique. On dit que la face extérieure du béton est en contact parfait avec la face intérieure de l'isolant. L'interprétation mathématique et algorithmique de cette phrase est connue dans l'environnement Motor-2 pour la simulation.

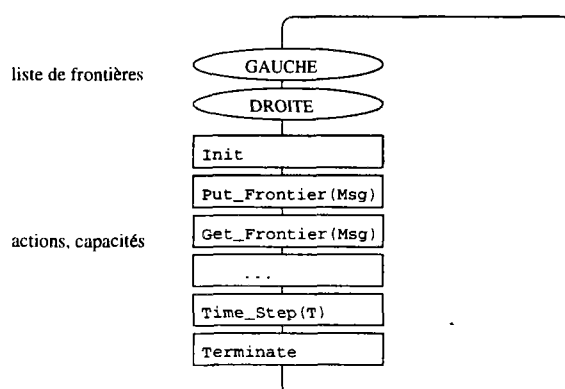


FIG. 5.1 - Les parties visibles d'un module : les frontières (en haut) spécifient les variables de communication, et les actions (en bas) que le module peut effectuer.

Dans le cadre de la simulation par Motor-2, les spécifications d'un module sont plus étendues. En utilisant la correspondance entre un composant du système et une file d'exécution séparée (*tâche*), nous ajoutons une qualité supplémentaire au module. Parfois on emploie le terme *acteur* pour désigner cette entité, parce que l'objet créé est une unité de calcul active, une sorte de programme indépendant. Nous avons donc dans la vue extérieure non seulement une partie statique des données (les frontières), mais aussi une partie dynamique. Il faut donc définir les actions qu'un module peut exécuter.

Tout type de module est dérivé d'un type de base abstrait (*Any_Module*), qui définit les actions nécessaires qu'un module doit être capable d'effectuer. Celle-ci sont par exemple l'initialisation et la terminaison de l'objet. Mais le développeur d'un modèle peut ajouter d'autres actions. Une extension à spécifier séparément est éventuellement la capacité du module à calculer des dérivées. Un utilisateur n'a pas besoin de s'occuper des capacités des *modules*, puisqu'elles sont liées au modèle utilisé pour le module. C'est le développeur de *modèle* qui définit les capacités. Tous les modules qui sont de ce type peuvent alors les exploiter.

Vu de l'extérieur, il n'y a pas de différence entre des modules composés et des modules terminaux. L'algorithme de résolution traite tous les modules de la même manière. Ceci est un principe de conception qui permet à Motor-2 de rester un environnement ouvert pour traiter des modèles *a priori* quelconques.

La vue extérieure d'un module étant définie, nous avons le choix entre plusieurs représentations internes. Nous pouvons facilement échanger les modèles associés sans besoin de modifier l'algorithme de raccordement. De cette manière, un modèle plus détaillé peut être exploité pour une partie du système, ou bien un modèle plus simple pour des composants jugés sans importance. L'utilisateur n'est pas obligé de modifier la structure du système. La description du système reste la même. On ne change que le modèle employé par un des composants.

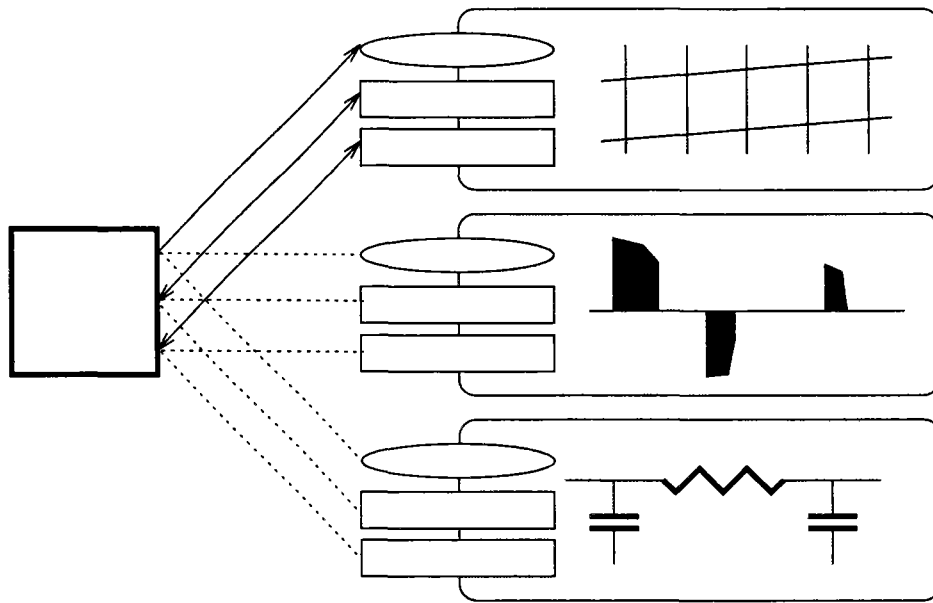


FIG. 5.2 - Pour une vue extérieure équivalente, on peut échanger les représentations internes des modules.

5.1.2 Vue intérieure

La partie intérieure d'un module est définie par le modèle qui est associé à celui-ci. L'environnement Motor-2 traduit les appels qui arrivent aux frontières d'un module par des appels correspondants aux fonctions définies par le modèle. Si par exemple le module MUR est de type DIFFÉRENCES FINIES 1D, et reçoit des nouvelles entrées aux frontières, ces valeurs sont automatiquement attribuées aux nœuds extrêmes, et un nouvel état peut être calculé à l'intérieur.

Les modules terminaux se voient attachés leurs propres méthodes locales (contenues dans le modèle). Nous ne nous en occupons pas pour l'instant. Le développement des modèles de calcul est la tâche d'un modélisateur. Ce sujet est traité plus en détail dans le chapitre 7.3.1.

La résolution du système consiste à déterminer les évolutions d'état des modules terminaux ainsi que la résolution locale du couplage. C'est la résolution des raccordements entre les modules. Le raccordement est situé à l'intérieur des modules composés. La vue interne d'un module composé contient donc les vues externes des sous-modules et les contraintes de raccordement. Pour exprimer ces raccordements dans le projet SYMBOL nous utilisons la notion d'*interface*. Le *type* d'une interface décrit la nature du raccordement. Une interface du type CONTACT PARFAIT vérifie l'égalité des températures et la compatibilité des flux de chaleur. Au contraire de ce qui est en usage dans le langage courant, il faut remarquer ici que nous utilisons le terme *interface* pour la communication entre modules et non pour la spécification d'un module qui est une *frontière*.

Aussi les modules composés peuvent être des sous-modules et doivent communiquer avec l'extérieur à travers des frontières. Qu'est ce qu'est une frontière

pour un module composé ? Dans un cas simple et général, on peut déclarer une frontière d'un sous-module comme équivalent à la frontière du module composé dans lequel il s'intègre (voir fig. 5.3). Nous extrayons la frontière d'un module fils, et nous la déclarons visible au niveau du père. La face intérieure du BÉTON est en même temps la face intérieure du MUR. Néanmoins il est quelque fois souhaitable de présenter plusieurs frontières internes comme une seule frontière externe. Le mur pourrait avoir deux parties, en béton en bas et en bois en haut par exemple. L'ensemble des frontières intérieures du BOIS et du BÉTON constituent maintenant la frontière INTÉRIEURE du MUR. Dans ce cas, il se pose la question pour l'utilisateur de savoir, quelles variables choisir pour la vue extérieure.

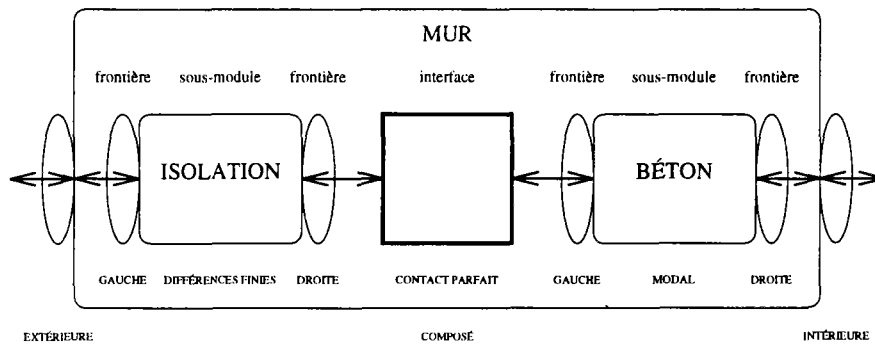


FIG. 5.3 - Le raccordement entre deux modules : la frontière DROITE du module ISOLATION est raccordée à la frontières GAUCHE du BÉTON. Entre les deux se trouve une interface du type CONTACT PARFAIT. Pour la communication au prochain niveau supérieur, la frontière DROITE du BÉTON constitue la frontière INTÉRIEURE du module composé MUR et de la même manière GAUCHE de l'ISOLATION constitue la frontière EXTÉRIEURE du MUR.

5.2 Moyens de communication des modules : frontières et pattes

Regardons un module au niveau mathématique plus en détail. Dans ce qui suit le nom d'un module sera indiqué par une lettre latine majuscule. Comme indice, il est mis dans le coin supérieur droit d'une variable.

L'état abstrait d'un module A est représenté par le vecteur η^A . Nous essayons de ne pas utiliser l'état d'un module, car l'état fait partie de la vue interne et nous ne voulons pas dépendre des représentations internes dans nos algorithmes. La communication du module avec l'extérieur passe à travers des *pins* (pattes). Les pattes sont structurées et regroupées en frontières². Une frontière Φ est signifiée par le vecteur de ses pattes Φ_i^A plus un *type* et éventuellement des paramètres supplémentaires. Une frontière regroupe un vecteur

2. Plusieurs frontières peuvent être rassemblées dans un *link*. Dans la plupart des cas il n'y a qu'une seule frontière dans un *link* et le *link* est obsolète. C'est une structure optionnelle qui est nécessaire pour traiter les cas de connexion par recouvrement où on raccorde plusieurs frontières d'un module à plusieurs frontières d'un autre module.

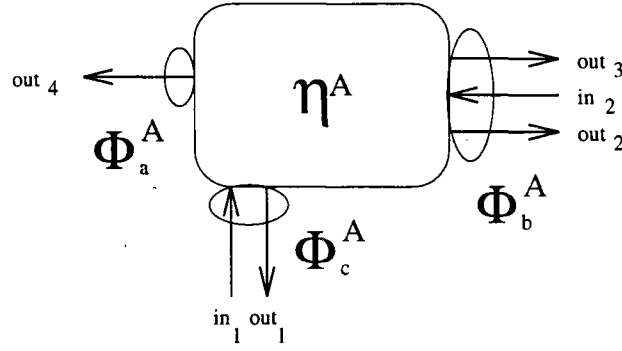


FIG. 5.4 - Les variables liées au module A.

local de variables d'entrées e_i^A et un vecteur local de variable de sorties s_i^A . Ces variables sont incluses dans le vecteur $\Phi_i^A = \begin{bmatrix} e_i^A \\ s_i^A \end{bmatrix}$. Pour des raisons de cohérence, nous associons toujours un *type* à une frontière. Ce type décrit le nombre et la fonction de ses pattes. Par exemple le type FIRST KIND exige une température en entrée et un flux de chaleur en sortie.

Si l'on ne regarde plus les frontières, mais directement les pattes de connexion, l'ensemble de toutes les pattes d'entrée d'un module sont décrites par le vecteur

$$\mathbf{E}^A = \begin{bmatrix} in_1 \\ in_2 \end{bmatrix}, \quad \text{les sorties par le vecteur} \quad \mathbf{S}^A = \begin{bmatrix} out_1 \\ out_2 \\ out_3 \\ out_4 \end{bmatrix}. \quad (5.1)$$

On peut également réintégrer la notion de frontière et voir les vecteurs d'entrées et de sorties composés des sous-vecteurs de frontières.

$$\mathbf{E}^A = \begin{bmatrix} \mathbf{e}_1^A \\ \mathbf{e}_2^A \\ \mathbf{e}_3^A \end{bmatrix} \quad \text{et} \quad \mathbf{S}^A = \begin{bmatrix} \mathbf{s}_1^A \\ \mathbf{s}_2^A \\ \mathbf{s}_3^A \end{bmatrix} \quad (5.2)$$

Parmi les capacités de base d'un module, il y a la possibilité de calculer les sorties en fonction des entrées. Il existe pour un module A une fonction du type

$$\mathbf{S}^A = \hat{R}^A(\eta^A, \mathbf{E}^A, t) \quad (5.3)$$

Les sorties $\mathbf{S}^A$ sont calculées en fonction de toutes les entrées du module $\mathbf{E}^A$, de l'état du module η^A et du temps t . Souvent on ne regarde qu'une frontière d'un module et on attribue à chacune des frontières d'un module A une fonction R_i^A de la forme

$$s_i^A = R_i^A(\eta^A, \mathbf{E}^A, t) \quad (5.4)$$

Nous appelons cette fonction la *réponse* de la frontière Φ_i du module A à l'ensemble des excitations extérieures $\mathbf{E}^A$. La réponse de l'équation 5.3 est alors composée de l'ensemble des réponses des frontières de l'équation 5.4.

$$\mathbf{S}^A = \begin{bmatrix} s_1^A \\ s_2^A \\ s_3^A \end{bmatrix} = \begin{bmatrix} R_1^A(\boldsymbol{\eta}^A, \mathbf{E}^A, t) \\ R_2^A(\boldsymbol{\eta}^A, \mathbf{E}^A, t) \\ R_3^A(\boldsymbol{\eta}^A, \mathbf{E}^A, t) \end{bmatrix} \quad (5.5)$$

Éventuellement, le module peut savoir calculer localement les sorties d'une frontière en fonction des entrées locales. Si la réponse de la frontière ne prend en compte que les entrées locales nous écrivons

$$s_i^A = \tilde{R}_i^A(\boldsymbol{\eta}^A, \mathbf{e}_i^A, t) \quad (5.6)$$

et on l'appelle la *réponse locale*. Il faut remarquer ici qu'on demande au module de calculer les sorties locales sans connaître toutes les entrées, ou en supposant que les autres entrées n'influencent pas s_i^A . En pratique cela signifie que les autres entrées conservent les valeurs qu'elles acquièrent par ailleurs, lorsqu'on observe l'effet de $\mathbf{e}_i^A$ sur $\boldsymbol{\eta}^A$ et s_i^A . $\tilde{R}$ ne dépend que de $\mathbf{e}$, et non de $\mathbf{E}$. Cette fonction est comparable avec l'évolution insensible φ_{ins} du FET (voir page 29). Généralement un module ne peut calculer la réponse locale que s'il suppose que les autres entrées gardent leur valeur actuelle ou s'il suppose une évolution donnée. Cette réponse locale est un concept important dans la réalisation de l'algorithme de résolution et également du parallélisme. Nous y revenons plus loin.

Rappelons que nous avons défini trois types de réponses, qu'un module peut être capable de fournir : calculer toutes les sorties en fonction de toutes les entrées ($\mathbf{S} = \hat{R}(\mathbf{E})$, éq. 5.3), calculer des sorties spécifiques en fonction de toutes les entrées ($s = R(\mathbf{E})$, éq. 5.4), et plus particulièrement calculer les sorties locales en fonction des entrées locales ($s = \tilde{R}(\mathbf{e})$, éq. 5.6).

Une frontière sert de lien entre un module et une interface. Sa description complète contient donc

- un *nom*, qui sert d'identificateur,
- ses deux vecteurs de *pattes* pour les entrées et les sorties,
- son *type* pour le contrôle,
- le *module* auquel elle appartient et ...
- l'*interface* à laquelle elle est connectée.

À l'intérieur d'une frontière, on trouve des *pattes*. Cette entité est également décrite par plusieurs attributs :

- un *nom* pour l'identification,

70 Modules et Raccordement : les deux entités majeures de Motor-2

- un *type* (température, débit massique, ...) qui peut être utilisé pour la vérification d'un raccordement,
- une *direction* qui indique la nature de la variable comme sortie ou entrée³,
- une valeur (généralement un nombre réel),
- une unité physique qui est liée au type de la variable.

Pour améliorer le contrôle de raccordement, et aussi de calcul interne, on pourrait ajouter aussi des valeurs minimale et maximale associées au type de la patte. Cela n'est pas implémenté pour l'instant dans Motor-2.

Comme nous l'avons déjà dit pour le type d'une frontière, le type d'une patte est un moyen pour mieux contrôler la cohérence de la description du système. Nous décrivons par ailleurs ainsi les variables à un niveau plus abstrait (plutôt technique que mathématique) qui permet de les gérer plus facilement. Dans le fichier SYMBOL, on peut ainsi vérifier à première vue le type des variables utilisées.

Ces informations sont utilisées dans le processus de la communication entre les modules. Elle est réalisée dans Motor-2 par les *interfaces* que nous décrivons plus en détail dans les paragraphes suivants.

5.3 Raccordement

5.3.1 Présentation

Comme déjà dit dans les chapitres précédents, nous utilisons une approche dans laquelle on découpe le système et construit un arbre de dépendances. Chaque *embranchement* de l'arbre correspond à des liens entre les modules. Ces liens sont à reconstituer lors de la simulation du système et on les appelle *raccordements*.

Découper un système en sous-systèmes nécessite d'écrire les conditions nécessaires que doivent respecter les composants séparés afin qu'ils représentent le comportement des éléments dans le système. Les comportements des composants sont couplés. Une étape importante est alors la reconstitution du système par *raccordement* des composants. Nous avons défini les *interfaces* comme objets abstraits qui rétablissent les conditions de raccordement.

Le couplage de modules dans un système ne s'exprime pas toujours par des équations de connectique simples. Dès qu'un couplage est autre chose que l'égalité de paires de variables, il y a raccordement et non plus connexion. Les interfaces sont alors des instances de *modèles de couplage*. Un modèle de couplage associé à une interface est par exemple le contact parfait qui vérifie

3. une troisième « direction » *observation* a également été définie comme sortie non-raccordable, mais elle n'est pas utilisée dans Motor-2.

une température égale dans toutes les frontières connectées et une annulation de la somme de flux de chaleur provenant des frontières.

Deux objets-phénomènes différents peuvent être complètement, ou en partie, topologiquement superposés. On n'aura cependant jamais à raccorder deux modules représentant le même phénomène en un même lieu. Si les deux modules concernent le même espace, c'est qu'ils représentent deux phénomènes différents. Cela peut être le rayonnement d'une part et l'échange convectif d'autre part, dans une pièce. Si deux modules sont du même type, les objets sont différents. Par exemple, les couches d'une paroi sont représentées par des modèles de type différences finies.

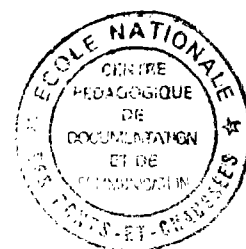
5.3.2 Topologies de raccordement

Le raccordement peut se réaliser sur des topologies très différentes. Les travaux de BLANC SOMMEREUX ont traité cette problématique pour la construction des *modèles couplés* [8]. Nous le rappelons ici en vue d'une simulation qui exploite des géométries de raccordement différentes. Cela peut éventuellement nous donner la possibilité de ne plus dépendre d'un arbre de découpage, mais plutôt d'un graphe plus général (voir § 4.2).

Si l'on ne regarde que le raccordement de modules par paire, nous pouvons distinguer ces couplages :

- Couplage bord à bord. On raccorde une frontière naturelle de l'un des modèles avec une frontière de l'autre modèle, comme on l'a déjà vu dans l'exemple d'une paroi bicouche.
- Couplage par recouvrement d'une partie commune. Dans la modélisation des bâtiments, on rencontre des situations où un raccordement par recouvrement peut être utile, par exemple pour le couplage de deux zones qui partagent un mur commun. Nous avons mentionné ce problème dans le chapitre « découpage » (§ 4.2). Pour ce type de raccordement les modèles doivent fournir des frontières qui ne se trouvent pas aux extrémités géométriques des composants. Des précaution spéciales sont à prendre, parce que la partie commune est souvent surdéterminée.
- Couplage par recouvrement avec élimination de la partie commune dans l'un des deux modules ou même dans les deux. Ce type de raccordement doit également être prévu pendant la construction du modèle, car il nécessite aussi une frontière interne sur laquelle on peut imposer des sollicitations et il doit fournir toutes les variables de la frontière externe.

En ce moment, seul le couplage bord à bord est implémenté et testé dans le simulateur Motor-2. Les autres topologies de couplage ne sont disponibles que pour la création de *modèles composés*. C'est à dire, avec la description SYMBOL, nous pouvons spécifier la synthèse de *modèles* basés sur des modèles plus élémentaires (*synthèse modale*, voir [8] et [36]).



5.3.3 Styles de description de raccordement

Un des objectifs du projet SYMBOL est de fournir un environnement de travail où on peut exprimer et vérifier le couplage entre les composants au niveau le plus haut, le plus naturel possible. L'idéal serait certainement de pouvoir dessiner deux modules, l'un à côté de l'autre et que le logiciel comprenne qu'il s'agit d'un raccordement de composants.

Dans le § 2.3 nous avons déjà vu quelques méthodes pour décrire le couplage entre les composants d'un système. TRNSYS utilise un fichier où on spécifie les indices des variables de sorties et d'entrée qui sont connectées. Par exemple, la sortie numéro 2 de l'unité 5 est l'entrée numéro 3 dans l'unité 7.

Dans la méthode des *Bondgraphs*, la description des connexions entre les composants est essentielle. Comme nous l'avons vu, une flèche indique la direction d'un « flux de puissance » qui est échangé entre deux composants. De plus, une marque existe pour spécifier la « causalité » et de cette façon un ordre de calcul préférentiel. Les flux d'information peuvent être ajoutés à ce schéma. Les *Bondgraphs* donnent une très bonne description des dépendances à l'intérieur d'un système. Leur représentation informatique dans le logiciel TUTSIM passe – similairement à TRNSYS – par des indices. Une connexion correcte entre les composants est vérifiée par le fait que l'ensemble du système d'équations est calculable.

Le format de fichier NMF qui est utilisé dans IDA (et aussi en partie dans Spark) permet non seulement la description de modules élémentaires, mais aussi de modules composés.

Une autre manière de spécifier les dépendances entre les composants d'un système consiste à créer une grande matrice qui a autant de lignes et de colonnes qu'il y a de composants. Les cases de la matrice contiennent les équations qui décrivent la dépendance du module de la colonne en fonction de celui de la ligne.

Dans le cadre du projet SYMBOL nous utilisons des modules composés pour décrire les connexions entre les composants. Le langage utilisé est détaillé au § 7.2. En bref, nous déclarons des interfaces comme points de communication à l'intérieur des modules composés. Ces interfaces ont un type et une liste des frontières qui sont raccordées par cette interface.

5.3.4 Raccordement dans Motor-2 aux différents niveaux d'abstraction

Dans le chapitre 4.1 les différents niveaux d'abstraction ont été présentés. Ces niveaux sont utilisés non seulement pour la description des modules mais aussi pour expliciter les relations entre les modules. Pour aboutir à une structuration pertinente, nous allons suivre l'évolution de la description à l'aide de notre exemple du mur bicouche.

- Dans le *monde technique* le raccordement entre deux objets est une connexion réelle. Une couche de laine de roche est clouée sur un mur en béton.

Dans d'autres cas cela peut être une fenêtre dans un mur ou un tuyau entre un capteur et une pompe.

- Si on passe au *monde physique*, les deux couches sont modélisées par une conduction monodimensionnelle. Pour ce qui concerne la connexion, nous simplifions les singularités des clous (!) et nous négligeons l'inévitable résistance thermique entre les couches. Le raccordement est décrit par le modèle du CONTACT PARFAIT ce qui veut dire qu'il y a une température commune à la surface de contact et le flux de chaleur qui sort du béton entre sans modification dans l'isolation.
- Au *monde mathématique* les modules sont représentés par des équations. En conséquence les relations entre les modules s'expriment soit par des variables communes et des variables de transfert, soit par des équations supplémentaires (prévoir des variables communes une simplification des équations supplémentaires). Pour le contact parfait nous écrivons :

$$T_{l,iso} = T_{0,béton} \quad \text{et} \quad \phi_{l,iso} = -\phi_{0,béton}$$

- Des solutions analytiques existent pour notre exemple si simple. Pour démontrer le cas général nous poursuivons la description au niveau du *monde algorithmique*. Nous utilisons la définition de réponse (éq. 5.4) pour une sorte de raccordement aveugle. Si les deux frontières du raccordement sont du type DIRICHLET, les réponses des modules contiennent le flux de chaleur en fonction de la température imposée. Nous construisons donc une fonction $f = f(R^{ISO}, R^{BÉT})$. Dans notre cas, f a la forme

$$f = R^{ISO}(T) + R^{BÉT}(T)$$

Nous faisons varier la température jusqu'à ce que la somme des flux de chaleur s'annule. L'algorithme pour trouver la solution est par exemple la méthode de NEWTON-RAPHSON.

5.3.5 Interface

Dans l'environnement Motor-2, nous avons choisi l'*interface* comme lieu (fictif) où s'effectue le couplage entre modules. On y décrit les conditions de raccordement. À l'intérieur d'un module composé on ne spécifie pas seulement le fait qu'il y a des sous-modules, mais on décrit aussi les connexions entre les sous-modules à l'aide des interfaces. Une interface est un point de communication entre des sous-modules fils d'un même module composé parent. C'est donc un regroupement d'un ensemble de connexions vers des frontières des sous-modules (voir fig. 5.3).

Pour pouvoir connecter des frontières très différentes aux interfaces, nous leur associons un *type* qui détermine un *modèle de couplage*. Le type d'une interface spécifie donc la nature du raccordement entre les sous-modules. Pour Motor-2 une interface n'a pas d'épaisseur, c'est à dire qu'elle n'a pas de capacité

physique et qu'il ne se produit aucun phénomène physique. Tout phénomène physique se trouve dans les modules. Comme nous l'avons déjà vu, une interface peut par exemple être du type CONTACT PARFAIT pour un raccordement entre la couche ISOLATION et la couche BÉTON qui constituent ensemble le module composé MUR.

Une interface est un ensemble de relations permettant de « brancher » les parties visibles des modules. C'est l'interface qui sert à *contraindre* les informations des frontières à être compatibles au sens du lien physique entre les modules. Elle même ne porte pas de phénomène physique. Nous regardons le raccordement plus en détail au niveau mathématique et algorithmique – les méthodes de résolution des contraintes – au chapitre 6.

Définition mathématique de l'interface

Le raccordement entre des modules est effectuée par des interfaces. Nous indiquons les interfaces par des petites lettres latines et lui associons également un type et des paramètres.

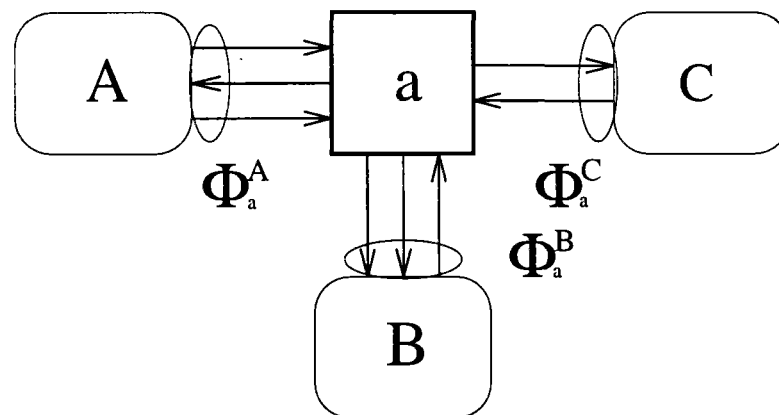


FIG. 5.5 - Vue de l'interface a.

Dans la figure 5.5 on voit le raccordement des trois modules A , B et C . Ils sont connectés à l'interface a à travers ses frontières Φ_a^A , Φ_a^B , et Φ_a^C . Cette interface rassemble

- les relations (en général non-linéaires) entre les entrées et les sorties d'une frontière pour chacune des frontières. Ce sont les équations des modules R_a^X .
- une relation entre l'état de l'interface η_a et les variables connectées (les pattes des modules). (L'état provient de l'algorithme de résolution (§ 6.5), il n'a pas de sens physique.)
- des relations contraignant le raccordement et donc l'évolution des modules. Elles sont définies par le *type* de l'interface. Ce type est appelé un *modèle de couplage*. Par exemple le type CONTACT PARFAIT exige égalité

des variables intensives (la température) et compatibilité des variables extensives (flux de chaleur))

Prenons maintenant l'exemple de la conduction monodimensionnelle dans un mur bicouche composé d'une couche A et d'une couche B . Les conditions aux limites de deux couches étant de première espèce, elles sont raccordées par l'interface a de type CONTACT PARFAIT. Le type de l'interface détermine les relations de raccordement.

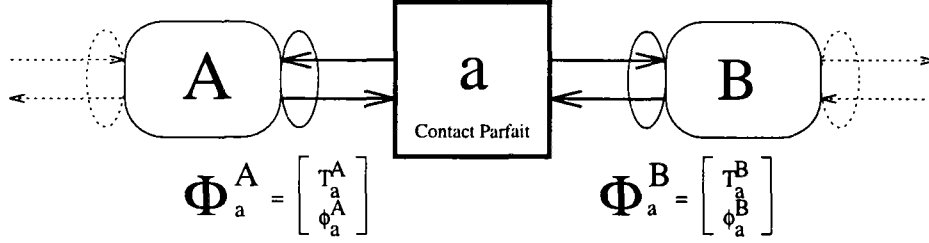


FIG. 5.6 - Exemple d'un mur bicouche.

Les contraintes du contact parfait sont une température commune et le bilan nul de flux de chaleur. Cela s'exprime par

$$T^A = T^B \quad \text{et} \quad \phi^A + \phi^B = 0. \quad (5.7)$$

Les frontières des modules A et B vers l'interface a sont composées d'une température et du flux correspondant

$$\Phi_a^A = \begin{bmatrix} T^A \\ \phi^A \end{bmatrix}, \quad \Phi_a^B = \begin{bmatrix} T^B \\ \phi^B \end{bmatrix}. \quad (5.8)$$

Les relations entre les variables d'une frontière sont les équations des modules. Les conditions aux limites étant de première espèce par exemple, on peut exprimer les flux en fonction des températures.

$$R^A(T^A, \eta^A, t) = \phi^A, \quad R^B(T^B, \eta^B, t) = \phi^B. \quad (5.9)$$

On peut rassembler ces équations dans une écriture symbolique de forme

$$\begin{array}{l} R_a^A(T^A, \eta^A, t) = \phi^A \\ R_a^B(T^B, \eta^B, t) = \phi^B \\ \text{température commune} \\ \text{égalité de flux} \end{array} \quad \begin{bmatrix} R_a^A & 0 & -1 & 0 \\ 0 & R_a^B & 0 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} T^A \\ T^B \\ \phi^A \\ \phi^B \end{bmatrix} = \mathbf{0} \quad (5.10)$$

R^A et R^B signifient ici les relations non-linéaires entre les températures imposées et les flux de chaleur des modules.

Généralisation

À ce stade il n'y a pas d'intérêt particulier à distinguer entre entrée et sortie d'un module. C'est l'ensemble des variables d'une frontière Φ qui est traité par le module. L'équation (5.6) qui calcule la réponse locale

$$\tilde{R}_i^A(\boldsymbol{\eta}^A, \mathbf{e}_i^A, t) - \mathbf{s}_i^A = 0 \quad (5.11)$$

devient alors

$$\tilde{f}_i^A(\Phi^A) = 0. \quad (5.12)$$

De la même façon nous pouvons aussi exprimer l'équation 5.4 qui calcule la sortie locale en fonction de toutes les entrées d'un module

$$R_i^A(\boldsymbol{\eta}^A, \mathbf{E}_i^A, t) - \mathbf{s}_i^A = 0 \quad (5.13)$$

Elle devient

$$f_i^A(\Phi^A) = 0 \quad (5.14)$$

Ici nous ne mentionnons plus la dépendance entre la frontière Φ et l'état $\boldsymbol{\eta}^A$ et le temps t . Ces dépendances sont « cachées » dans le module et ne sont pas « visibles » pour l'interface. L'algorithme n'a pour résoudre le système d'équations de l'interface que les variables présentes dans la frontière.

Un module A calcule $f_a^A(\Phi^A)$ dans sa frontière connectée à l'interface a . Les équations de contraintes relient les pattes des modules raccordés à l'intérieur de l'interface. Dans le cas général, les équations des contraintes peuvent être également non-linéaires, et peuvent inclure une dépendance de l'état de l'interface $\boldsymbol{\eta}_a$, mais elles ne sont jamais discontinues⁴. Dans le § 6.5, nous voyons comment l'état d'une interface peut être utilisé dans la simulation. L'ensemble des équations d'une interface est alors un système d'équations non-linéaires (SENL) qu'il faut résoudre. Les algorithmes pour résoudre des SENL cherchent à trouver des valeurs de variables pour lesquelles toutes les fonctions s'annulent.

$$\begin{array}{ll} \text{équations des modules} & \begin{array}{l} f^A(\Phi_a^A, 0, 0, \dots, 0) = 0 \\ f^B(0, \Phi_a^B, 0, \dots, 0) = 0 \\ f^C(0, 0, \Phi_a^C, \dots, 0) = 0 \\ \vdots \\ f_1^a(\Phi_a^A, \Phi_a^B, \Phi_a^C, \dots, \boldsymbol{\eta}_a) = 0 \\ \vdots \end{array} \\ \text{équations des contraintes} & \begin{array}{l} f_n^a(\Phi_a^A, \Phi_a^B, \Phi_a^C, \dots, \boldsymbol{\eta}_a) = 0 \end{array} \end{array} \quad (5.15)$$

Le système (5.15) a essentiellement deux parties. L'une contient les équations indépendantes R_a des modules. Ici on ne trouve pas de connexion entre

4. Toute éventuelle discontinuité est dans la réponse du module R .

les frontières et leurs variables associées. Comme nous ne regardons que des variables « visibles » pour l'interface, les dépendances des états des modules ne sont pas mentionnées dans cette écriture. On remarque déjà qu'il n'y a des éléments que dans la diagonale principale. L'algorithme de résolution doit prendre en compte ce fait afin d'améliorer l'efficacité numérique. La deuxième partie contient les équations de raccordement qui sont propre à l'interface et qui dépendent de son type. Ces équations relient les frontières entre elles, et éventuellement avec un état de l'interface.

Nous cherchons alors la solution d'un système d'équations non-linéaires (SENL). Pour des connexions purement physiques, nous savons qu'une solution existe dans le monde réel. Dans le monde mathématique ou algorithmique le problème peut être mal posé. Cela peut nous compliquer la tâche ou même empêcher de trouver la solution.

Le chapitre suivant traite les différentes méthodes qui ont été implémentées dans le simulateur Motor-2 pour résoudre les systèmes d'équations qui apparaissent dans un module composé. L'influence d'une parallélisation de l'exécution sur les calculs a une place importante dans ce contexte.

78 Modules et Raccordement : les deux entités majeures de Motor-2