A Multiscale Model Derivation and Simulation Tool for MEMS Arrays

B. Yang[1], W. Belkhir [2], M. Lenczner[1] and N. Ratier[1]
Email: bin.yang@femto-st.fr, walid.belkhir@inria.fr, michel.lenczner@utbm.fr and nicolas.ratier@femto-st.fr
[1]FEMTO-ST, Département Temps-Fréquence, University of Franche-Comté,
26 chemin de l'Epitaphe, 25030 Besançon Cedex, France
[2]INRIA Nancy - Grand Est, CASSIS project
54600 Villers-lés-Nancy, France

Abstract

We introduce a framework for computer-aided derivation of multi-scale models dedicated to arrays of microsystems. It relies on a combination of a asymptotic methods used in the field of partial differential equations with term rewriting techniques coming from computer science. In our approach, a multi-scale model derivation is characterized by the features taken into account in the asymptotic analyses. Its formulation consists in a derivation of a reference model associated to an elementary nominal model, and in a set of transformations to apply to this proof until it takes into account the wanted features. In addition to the reference model proof, the framework includes first order rewriting principles designed for asymptotic model derivations, and second order rewriting principles dedicated to transformations of model derivations. We apply the method to generate a family of homogenized models for second order elliptic equations with periodic coefficients that could be posed in multi-dimensional domains, with possibly multi-domains and/or thin domains. The transfer of asymptotic models into a finite element software package is illustrated through an example of a model of periodic cantilever arrays.

KEY WORDS: *Multi-scale modeling, Rewriting, Homogenization, Symbolic computation*

1. Introduction

We are currently developing MEMSALab a software package dedicated to asymptotic multiphysic model derivation applied to arrays of microsystems. It is based on asymptotic methods for systems governed by partial differential equations. Asymptotic analyses are applied for instance to large periodic structures, with small ratio of their cell size to their global size, or to thin structures, with small ratio of their thickness to their global size. Generally, the asymptotic models are computed much faster than the original ones. In this field, a lot of techniques have been developed. However, none of them have been implemented in a systematical approach to render it available to engineers as a design tool. In fact, most published paper focus on an asymptotic model for given geometry and physics and very few cover general cases. Considering the constant need for faster and more reliable simulations, as well as the variety of geometrical and physical features of microsystems, it becomes essential to find a general and systematic approach. Ours starts from a reference model derivation, and then extend it to adapt new features and physics. The reference model is a simple second order boundary value problem posed on an interval and with periodic coefficients. The derivation of a complex model involving complex geometries and physics is generated by a process of generalization. The techniques of model derivation are using the mathematical tool called the Two-Scale Transform or unfolding method, see [11], [4], [5]. The manner to express them and their generalization comes from formal specification techniques in computer science, namely term rewriting and rewriting strategies [1, 6]. The software is written in the symbolic computation language Maple™ and it is intended to be connected to the finite element software packages FreeFEM++ [8] and COMSOL™ for numerical simulation. In this paper we overview the mathematical and computer science principles of our method in its current state. First results of combination of elementary features are reported. A link with a finite element method software is illustrated on the multi-scale model of cantilever arrays derived in [11] and studied in [9].

The methodology used for the software package design is described in [13] and is now outlined. Its kernel is the rewriting based language *Symbtrans* [2]. A multi-scale model derivation is characterized by the features taken into account in the asymptotic analyses. Its formulation consists in a derivation of an elementary reference model and in a set of transformations to apply to this proof until it takes into accounts the wanted features. In addition to the reference model proof, the framework includes first order rewriting strategies designed for its derivation, and second order rewriting strategies, operating on first order strategies, for its transformations. Rewriting operates on expressions whose level of abstraction accurately reflects the mathematical framework. Their description follows a well defined grammar in order that they carry enough information allowing for the design of the rewriting rules and the strategies. Put together all these concepts can express and build proofs as first order strategies, and therefore provide a framework of symbolic computation.

2. Mathematical Formulation

The mathematical statement of the reference model and the corresponding asymptotic models are briefly recalled. We consider an interval $\Omega = \bigcup_{c=1}^{N(\varepsilon)} \Omega_c^{1,\varepsilon} \subset \mathbb{R}$ divided into $N(\varepsilon)$ periodic cells (or intervals) $\Omega_c^{1,\varepsilon}$, of size $\varepsilon > 0$ a small number tending to zero in the asymptotic analysis, indexed by c, and with center x_c. The translation and magnification $\Omega^1 = (\Omega_c^{1,\varepsilon} - x_c)/\varepsilon$ is the *unit cell*. The variables in Ω and in Ω^1 are denoted by x^ε and x^1. The two-scale transform T is an operator mapping functions defined in the physical domain Ω to functions defined in the two-scale domain $\Omega^\sharp \times \Omega^1$ where for the *reference model* $\Omega^\sharp = \Omega$. It is defined by

$$(Tu)(x_c, x^1) = u(x_c + \varepsilon x^1) \tag{1}$$

and then by extension $T(u)(x^\sharp, x^1) = u(x_c + \varepsilon x^1)$ for all $x^\sharp \in \Omega_c^{1,\varepsilon}$ and each c in $1, .., N(\varepsilon)$. We consider u^ε the solution of a linear boundary value problem posed in Ω,

$$\begin{cases} -\dfrac{d}{dx}(a^\varepsilon(x)\dfrac{du^\varepsilon(x)}{dx}) = f \text{ in } \Omega \\ u^\varepsilon = 0 \text{ on } \Gamma, \end{cases} \tag{2}$$

where the coefficient a^ε is $\varepsilon\Omega^1$-periodic. The weak formulation is obtained by multiplication of the differential equation by a test function v vanishing on the boundary and application of the Green formula,

$$\int_\Omega a^\varepsilon(x)\frac{du^\varepsilon}{dx}\frac{dv}{dx}\,dx = \int_\Omega f(x)v(x)\,dx. \tag{3}$$

We assume that for some functions $a^0(x^1)$ and $f^0(x^\sharp)$,

$$T(a^\varepsilon) = a^0 \text{ and } T(f) = f^0(x^\sharp) + O_w(\varepsilon). \tag{4}$$

The next proposition states the homogenized model and is the main result of the *reference proof*. For θ a solution to the microscopic problem

$$\int_{\Omega^1} a^0\frac{\partial\theta}{\partial x^1}\frac{\partial w}{\partial x^1}\,dx^1 = -\int_{\Omega^1} a^0\frac{\partial w}{\partial x^1}\,dx^1 \quad \text{for all } w\ \Omega^1\text{-periodic},$$

the homogenized coefficient and right-hand side are defined by

$$a^H = \int_{\Omega^1} a^0\left(1 + \frac{\partial\theta}{\partial x^1}\right)^2 dx^1 \text{ and } f^H = \int_{\Omega^1} f^0\,dx^1. \tag{5}$$

Proposition 1 *The limit u^0 of Tu^ε is solution to the weak formulation*

$$\int_{\Omega^\sharp} a^H\frac{du^0}{dx^\sharp}\frac{dv^0}{dx^\sharp}\,dx^\sharp = \int_{\Omega^\sharp} f^H v^0\,dx^\sharp \tag{6}$$

for all test function v^0 sufficiently regular on $\Omega^\sharp$ and vanishing on its boundary.

3. Rewriting strategies

In this section we recall the rudiments of rewriting, namely, the definitions of terms over a signature, of substitution and of rewriting rules. We introduce a strategy language that will allow to express all the useful rewriting strategies.

3.1 Term, substitution and rewriting rule.

We start with an example of rewriting rule. We define a set of rewriting variables $\mathcal{X} = \{x, y\}$ and a set of function symbols $\Sigma = \{f, g, a, b, c\}$. A term is a combination of elements of $\mathcal{X} \cup \Sigma$, for instance $f(x)$, $f(a)$ or $g(g(a, x), f(y))$. We denoted by $\mathcal{T}(\Sigma, \mathcal{X})$ the set of all terms. The rewriting rule $f(x) \rightsquigarrow g(x)$ applied to a term $f(a)$ is a two-step operation. First, it consists in matching the left term $f(x)$ with the input term $f(a)$ by matching the two occurences of the symbol of function f, and by matching the rewriting variable x with the symbol of function a. Then, the result $g(a)$ of the rewriting operation is obtained by replacing the rewriting variable x occuring in the right hand side $g(x)$ by the subterm a that have been associated to x. In case where any substitution is possible, as in the application of $f(b) \to g(x)$ to $f(a)$, we say that the rewriting rule fails.

We denote by Σ_n the subset of Σ of the function symbols of arity n. For instance in the example f and g belong to Σ_1 while a and b belong to Σ_0. Two other common examples of terms are the expression $Integral(\Omega, f(x), x)$ and *diff*$(f(x), x)$ which represent the expressions $\int_\Omega f(x)\,dx$ and $\dfrac{df(x)}{dx}$. Notice that $Integral \in \Sigma_3$, *diff* $\in \Sigma_2$, $f \in \Sigma_1$ and $x, \Omega \in \Sigma_0$. For the sake of simplicity we often keep the symbolic mathematical notation to express the rewriting rules. In the following we see a term as an oriented, ranked and rooted tree as it is usual in symbolic computation. We recall that in a ranked tree the child order is important. For instance the tree associated to the term $Integral(\Omega, f(x), x)$ has $Integral$ as its root which has three children in the order Ω, f, x and f has one child x.

A *substitution* is a function $\sigma : \mathcal{X} \to \mathcal{T}(\Sigma, \mathcal{X})$. The *application* of a substitution σ to a term t, denoted by $\sigma(t)$, simultaneously replaces all occurrences of variables in t by their σ-images.

A *rewriting rule*, is a pair (l, r) where l and r are terms in $\mathcal{T}(\Sigma, \mathcal{X})$; it will also be denoted by $l \rightsquigarrow r$. We observe that for any two terms s, t, there exists at most one substitution σ such that $\sigma(s) = t$. We mention that a rewriting rule stands for the rule application at the top position. It is more useful to be able to apply a rule at arbitrary position, and more generally to specify the way rules are applied. For this purpose we next present a strategy language that allows to built strategies out of basic constructors. To this end, we introduce strategy constructor symbols $;\,, \rightsquigarrow, \oplus, \mu, etc$ that do not belong to $\Sigma \cup \mathcal{X}$. Informally, the constructor ";" stands for the composition, "$\oplus$" for the left choice, $Some$ for the application of a strategy to the immediate subterms of the input term, $\eta(x)$ for the fail as identity constructor, $Child(j, s)$ applies the strategy s to the j^{th} immediate subterm, X is a fixed-point variable, and μ is the fixed-point or the iterator constructor, its purpose is to define recursive strategies. For example, the strategy $\mu X.(s; X)$ stands for $s; s; \ldots$, that is, it is the iteration of the application of s until a fixed-point is reached.

Definition 2 ***(Strategy)*** *A strategy is inductively defined by the following grammar:*

$$\begin{aligned} s \quad ::= \quad & l \rightsquigarrow r \ \mid \ s; s \ \mid \ s \oplus s \ \mid \ \eta(s) \ \mid \ Some(s) \\ & \mid \ Child(j, s) \ \mid \ X \ \mid \ \mu X.s \end{aligned} \tag{7}$$

where $j \in \mathbb{N}$. The set of strategies defined from a set of rewriting rules in $\mathcal{T} \times \mathcal{T}$ is denoted by $\mathcal{S}_\mathcal{T}$.

Example 3 *Out of the basic constructors of strategies given in Definition 2, we built up some useful strategies. The strategy $TopDown(s)$ applies the strategy s to an input term t in a top down way starting from the root, it stops when it succeeds. That is, if the strategy s succeeds on some subterm t' of t, then it is not applied to the proper subterms of t'. The strategy $OuterMost(s)$ behaves exactly like $TopDown(s)$ apart that if the strategy s succeeds on some subterm t' of t, then it is also applied to the proper subterms of t'. The strategy $BottomUp(s)$ (resp. $InnerMost(s)$) behaves like $BottomUp(s)$ (resp. $InnerMost(s)$) but in the opposite direction, i.e. it traverses a term t starting from the leafs. The strategy $Normalizer(s)$ iterates the application of s until a fixed-point is reached.*

3.2 Conditional rewriting

Rewriting with conditional rules, also known as conditional rewriting, extends the basic rewriting with the notion of condition. A conditional rewrite rule is a triplet:

$$(l, r, c)$$

where c is a logical formula expressed in some logic. The set of strategies defined over rewriting rules $(l, r, c) \in \mathcal{T} \times \mathcal{T} \times \mathcal{T}_c$ is denoted by $\mathcal{S}_{\mathcal{T},\mathcal{T}_c}$.

4. A Symbolic Computation Framework for Model Derivation

In this section we propose a framework for the two-scale model proofs where the latter are formulated as rewriting strategies. We notice that the following framework allows for the complete representation of the data. It does not rely on external structures such as hash tables. To this end, we define the syntax of the mathematical expressions by means of a grammar $\mathcal{G}$.

4.1 A Grammar for Mathematical Expressions

The grammar includes four rules to built terms for mathematical functions $\mathcal{F}$, regions $\mathcal{R}$, mathematical variables $\mathcal{V}$, and boundary conditions $\mathcal{C}$. It involves Σ_{Reg}, Σ_{Var}, Σ_{Fun}, Σ_{Oper}, and Σ_{Cons} which are sets of names of regions, variables, functions, operators, and constants so subsets of Σ_0. Empty expressions in Σ_{Reg} and Σ_{Fun} are denoted by $\perp_{\mathcal{R}}$ and $\perp_{\mathcal{F}}$. The set of usual algebraic operations $\Sigma_{Op} = \{+, -, \times, /, \hat{}\}$ is a subset of Σ_2. The elements of $\Sigma_{Type} = \{$*Unknown*, *Test*, *Known*, $\perp_{Type}\} \subset \Sigma_0$, $\perp_{Type}$ denoting the empty expression, are to specify the nature of a function, namely an unknown function (as u^ε, u^0, u^1 in the proof), a test function (as v, v^0, v^1) in a weak formulation or another known function (as a^ε, f^ε, a^0, f^0 or n_{Γ^1}). The boundary conditions satisfied by a function are specified by the elements of $\Sigma_{BC} = \{d, n, pd, apd, t\} \subset \Sigma_0$ to express that it satisfies Dirichlet, Neuman, periodic, anti-periodic or transmission conditions. The grammar also involve the symbols of functions `Reg`, `Fun`, `IndexedFun`, `IndexedReg`, `IndexedVar`, `Oper`, `Var`, and `BC` that define regions, mathematical functions, indexed functions or regions or variables, operators, mathematical variables and boundary conditions. The grammar reads as

$$
\begin{aligned}
\mathcal{F} ::= \; & \circledast(\mathcal{F},\mathcal{F}) \;|\; d \;|\; \mathcal{V} \;| \\
& \texttt{Fun}(f, [\mathcal{V},\ldots,\mathcal{V}], [\mathcal{C},\ldots,\mathcal{C}], K) \;| \\
& \texttt{IndexedFun}(\mathcal{F},\mathcal{V}) \;| \\
& \texttt{Oper}(A, [\mathcal{F},\ldots,\mathcal{F}], [\mathcal{V},\ldots,\mathcal{V}], [\mathcal{V},\ldots,\mathcal{V}], [d,\ldots,d]) \;| \\
& \perp_{\mathcal{F}}, \\
\mathcal{R} ::= \; & \texttt{Reg}(\Omega, [d,\ldots,d], \{\mathcal{R},\ldots,\mathcal{R}\}, \mathcal{R}, \mathcal{F}) \;| \\
& \texttt{IndexedReg}(\mathcal{F},\mathcal{V}) \;| \\
& \perp_{\mathcal{R}}, \\
\mathcal{V} ::= \; & \texttt{Var}(x,\mathcal{R}) \;|\; \texttt{IndexedVar}(\mathcal{V},\mathcal{V}), \\
\mathcal{C} ::= \; & \texttt{BC}(c,\mathcal{R},\mathcal{F}),
\end{aligned}
$$

where the symbols Ω, d, $\circledast$, f, K, A, x and c hold for any function symbols in Σ_{Reg}, Σ_{Cons}, Σ_{Op}, Σ_{Fun}, Σ_{Type}, Σ_{Oper}, Σ_{Var}, and Σ_{BC}. The arguments of a region term are its region name, the list of its space directions (e.g. [1,3] for a plane in the variables (x_1, x_3)), the (possibly empty) set of subregions, the boundary and the outward unit normal. Those of a function term are its function name, the list of the mathematical variables that range over its domain, its list of boundary conditions, and its nature. Those for an indexed region or variable or function term are its function or variable term and its index (which should be discrete). For an operator term these are its name, the list of its arguments, the list of mathematical variable terms that it depends, the list of mathematical variable terms of its co-domain (useful e.g. for T when the image cannot be deduced from the initial set), and a list of parameters. Finally, the arguments of a boundary condition term are its type, the boundary where it applies and an imposed function if there is one. For example, the imposed function is set to 0 for an homogeneous Dirichlet condition and there is no imposed function in a periodicity condition. We shall denote by $\mathcal{T}_{\mathcal{R}}(\Sigma,\emptyset)$, $\mathcal{T}_{\mathcal{F}}(\Sigma,\emptyset)$, $\mathcal{T}_{\mathcal{V}}(\Sigma,\emptyset)$, and $\mathcal{T}_{\mathcal{C}}(\Sigma,\emptyset)$ the set of terms generated by the grammar starting from the non-terminal $\mathcal{R}$, $\mathcal{F}$, $\mathcal{V}$, and $\mathcal{C}$. The set of all terms generated by the grammar (i.e. starting from $\mathcal{R}$, $\mathcal{F}$, $\mathcal{V}$, or $\mathcal{C}$) is denoted by $\mathcal{T}_{\mathcal{G}}(\Sigma,\emptyset)$. Finally, we also define the set of terms $\mathcal{T}_{\mathcal{G}}(\Sigma,\mathcal{X})$ where each non-terminal $\mathcal{R}$, $\mathcal{F}$, $\mathcal{V}$, and $\mathcal{C}$ can be replaced by a rewriting variable in $\mathcal{X}$. Equivalently, it can be generated by the extension of $\mathcal{G}$ obtained by adding " | x" with $x \in \mathcal{X}$ in the definition of each non-terminal term. Or, by adding $N ::= x$, with $x \in \mathcal{X}$ for each non-terminal N.

4.2 Short-cut Terms

For the sake of conciseness, we introduce shortcut terms that are constantly used in the end of the paper: $\underline{\Omega} \in \mathcal{T}_{\mathcal{R}}(\Sigma,\mathcal{X})$, $\underline{x} \in \mathcal{T}_{\mathcal{V}}(\Sigma,\mathcal{X})$ defined in $\underline{\Omega}$, $\underline{I} \in \mathcal{T}_{\mathcal{R}}(\Sigma,\mathcal{X})$ used for (discrete) indices, $\underline{i} \in \mathcal{T}_{\mathcal{V}}(\Sigma,\mathcal{X})$ used as an index defined in $\underline{I}$, $\underline{u} \in \mathcal{T}_{\mathcal{F}}(\Sigma,\mathcal{X})$ or $\underline{u}(\underline{x}) \in \mathcal{T}_{\mathcal{F}}(\Sigma,\mathcal{X})$ to express that it depends on the variable $\underline{x}$ and $\underline{u}_{\underline{i}}$ the indexed-term of the function $\underline{u}$ indexed by $\underline{i}$. Similar definitions can be given for the other notations used in the proof as $\underline{\Omega}^\sharp$, $\underline{x}^\sharp$, $\underline{\Omega}^1$, $\underline{x}^1$, $\underline{\Omega'}$, $\underline{x'}$, $\underline{v}(\underline{x}^\sharp, \underline{x}^1)$ etc.

Example 4 *For instance,* $\underline{\Omega} = \texttt{Reg}(\Omega, [2], \emptyset, \underline{\Gamma}, \underline{n})$, *where* $\underline{\Gamma} = \texttt{Reg}(\Gamma, [], \emptyset, \perp_{\mathcal{R}}, \perp_{\mathcal{F}})$, $\underline{n} = \texttt{Fun}(n, [\underline{x'}], [], \texttt{Known})$,

$\underline{x}' = \mathtt{Var}(x, \underline{\Omega}')$ *and* $\underline{\Omega}' = \mathtt{Reg}(\Omega, [2], \emptyset, \underline{\Gamma}, \perp_{\mathcal{F}})$ *refers to a region-term representing a one-dimensional domain named* Ω*, oriented in the direction* x_2*, with boundary* $\underline{\Gamma}$ *and with outward unit normal* $\underline{n}$*. The shortcut* $\underline{\Gamma}$ *refers to a region term representing the boundary named* Γ*. An unknown function* $u(x)$ *defined on* $\underline{\Omega}$ *satisfying homogeneous Dirichlet boundary condition* $u(x) = 0$ *on* $\underline{\Gamma}$ *is represented by the function-term,* $\underline{u}(\underline{x}) = \mathtt{Fun}(u, [\underline{x}], \mathtt{Cond}(d, \underline{\Gamma}, 0), \mathtt{Unknown})$ *where* $\underline{x} = \mathtt{Var}(x, \underline{\Omega})$*.*

The operators necessary for the proof are the integral, the derivative, the restriction operator tr (restricting a function defined on a region to the boundary), the two-scale transform T and other detailed in [13]. In addition, for most of the extensions of the reference proof we shall use the discrete sum operator. Instead of writing operator-terms as defined in the grammar, we prefer to use the usual mathematical expressions. The table below establish the correspondance between them,

$$\int \underline{u}\, d\underline{x} \equiv \mathtt{Oper}(\mathtt{Integral}, \underline{u}, [\underline{x}], [], []),$$
$$\frac{\partial \underline{u}}{\partial \underline{x}} \equiv \mathtt{Oper}(\mathtt{Partial}, \underline{u}, [\underline{x}], [\underline{x}], []),$$
$$tr(\underline{u}, \underline{x})(\underline{x}') \equiv \mathtt{Oper}(\mathrm{Restriction}, \underline{u}, [\underline{x}], [\underline{x}'], []),$$
$$T(\underline{u}, \underline{x})(\underline{x}^\sharp, \underline{x}^1) \equiv \mathtt{Oper}(T, \underline{u}, [\underline{x}], [\underline{x}^\sharp, \underline{x}^1], [\varepsilon]),$$
$$\sum\nolimits_{\underline{i}} \underline{u}_{\underline{i}} \equiv \mathtt{Oper}(\mathtt{Sum}, \underline{u}_{\underline{i}}, [\underline{i}], [], []).$$

The multiplication and exponentiation involving two terms f and g are written fg and f^g as usual in mathematics. All these conventions has been introduced for terms in $\mathcal{T}(\Sigma, \emptyset)$. For terms in $\mathcal{T}(\Sigma, X)$ as those encoutered in rewriting rules, the rewriting variables can replace any of the above short cut terms.

Example 5 *The rewriting rule associated to the Green rule reads*

$$\int \frac{\partial u}{\partial \underline{x}} v\; d\underline{x} \rightsquigarrow -\int u \frac{\partial v}{\partial \underline{x}}\; d\underline{x} + \int tr(u)\; tr(v)\; n\; d\underline{x}'.$$

with the short-cuts $\underline{\Gamma} = \mathtt{Reg}(\Gamma, d1, \emptyset, \perp_{\mathcal{R}}, \perp_{\mathcal{F}})$*,* $\underline{\Omega} = \mathtt{Reg}(\Omega, d2, \emptyset, \underline{\Gamma}, n)$*,* $\underline{x} = \mathtt{Var}(x, \underline{\Omega})$ *and* $\underline{x}' = \mathtt{Var}(x, \underline{\Gamma})$*. The other symbols* u*,* v*,* x*,* Ω*,* Γ*,* $d1$*,* $d2$*,* n *are rewriting variables, and for instance*

$$\frac{\partial u}{\partial x} \equiv \mathtt{Oper}(\mathtt{Partial}, u, x, [], []).$$

Applying this rule according to an appropriate strategy, say the top down strategy, to a term in $\mathcal{T}(\Sigma, \emptyset)$ *like*

$$\Psi = \int \frac{\partial \underline{f}(\underline{z})}{\partial \underline{z}} \underline{g}(\underline{z})\; d\underline{z},$$

for a given variable term $\underline{z}$ *and function terms* $\underline{f}$*,* $\underline{g}$*. As expected, the result is*

$$-\int \underline{f}\; \frac{\partial \underline{g}}{\partial \underline{z}}\; d\underline{z} + \int \underline{f}\; \underline{g}\; \underline{n}\; d\underline{z}'$$

with evident notations for $\underline{n}$ *and* $\underline{z}'$*.*

4.3 A Variable Dependency Analyzer

The *variable dependency analyzer* Θ is related to *effect systems* in computer science [12]. It is a function from $\mathcal{T}_{\mathcal{F}}(\Sigma, \emptyset)$ to the set $\mathcal{P}(\mathcal{T}_{\mathcal{V}}(\Sigma, \emptyset))$ of the parts of $\mathcal{T}_{\mathcal{V}}(\Sigma, \emptyset)$. When applied to a term $t \in \mathcal{T}_{\mathcal{F}}(\Sigma, \emptyset)$, it returns the set of mathematical variables on which t depends.

Example 6 *For*

$$\Psi = \int_{\underline{\Omega}^\sharp} \int_{\underline{\Omega}^1} T(\underline{u}(\underline{x}), \underline{x})(\underline{x}^\sharp, \underline{x}^1) \frac{\partial \underline{v}(\underline{x}^\sharp, \underline{x}^1)}{\partial \underline{x}^1} d\underline{x}^1 d\underline{x}^\sharp$$

in $\mathcal{T}_{\mathcal{F}}(\Sigma, \emptyset)$*, the set* $\Theta(\Psi)$ *of mathematical variables on which* Ψ *depends is hence inductively computed as follows:*

$$\Theta(\underline{u}(\underline{x})) = \{\underline{x}\},\ \Theta(T(\underline{u}(\underline{x}), \underline{x})(\underline{x}^\sharp, \underline{x}^1)) = \{\underline{x}^\sharp, \underline{x}^1\},$$
$$\Theta(\underline{v}(\underline{x}^\sharp, \underline{x}^1)) = \{\underline{x}^\sharp, \underline{x}^1\},\ \Theta(\frac{\partial \underline{v}(\underline{x}^\sharp, \underline{x}^1)}{\partial \underline{x}^1}) = \{\underline{x}^\sharp, \underline{x}^1\},$$
$$\Theta(T(\underline{u}(\underline{x}), \underline{x})(\underline{x}^\sharp, \underline{x}^1) \frac{\partial \underline{v}(\underline{x}^\sharp, \underline{x}^1)}{\partial \underline{x}^1}) = \{\underline{x}^\sharp, \underline{x}^1\},$$
$$\Theta(\int_{\underline{\Omega}^1} T(\underline{u}(\underline{x}), \underline{x})(\underline{x}^\sharp, \underline{x}^1) \frac{\partial \underline{v}(\underline{x}^\sharp, \underline{x}^1)}{\partial \underline{x}^1} d\underline{x}^1) = \{\underline{x}^\sharp\},$$

and $\Theta(\Psi) = \emptyset$*.*

4.4 Formulation of the Symbolic Framework for Model Derivation

Now we are ready to define the framework for two-scale model derivation by rewriting. To do so, the rewriting rules are restricted to left and right terms $(l, r) \in \mathcal{T}_{\mathcal{G}}(\Sigma, \mathcal{X}) \times \mathcal{T}_{\mathcal{G}}(\Sigma, \mathcal{X})$. Their conditions c are formulas generated by a grammar, not explicited here, combining terms in $\mathcal{T}_{\mathcal{G}}(\Sigma, \mathcal{X})$ with the usual logical operators in $\Lambda = \{\vee, \wedge, \rceil, \in\}$. It also involves operations with the dependency analyzer Θ. The set of terms generated by this grammar is denoted by $\mathcal{T}_{\mathcal{L}}(\Sigma, \mathcal{X}, \mathcal{G}, \Theta, \Lambda)$.
A model derivation is divided into several intermediary lemmas. Each of them is intended to produce a new property that can be expressed as one or few rewriting rules to be applied in another part of the derivation. Since dynamical creation of rules is not allowed, a strategy is covering one lemma only and is operating with a fixed set of rewriting rules. The conversion of a result of a strategy to a new set of rewriting rules is done by an elementary external operation that is not a limitation for generalizations of proofs. The following definition summarizes the framework of symbolic computation developed in this paper.

Definition 7 *The components of the quintuplet* $\Xi = \langle \Sigma, \mathcal{X}, E, \mathcal{G}, \Theta \rangle$ *provide a framework for symbolic computation to derive multi-scale models. A two-scale model derivation is expressed as a strategy* $\pi \in \mathcal{S}_{\mathcal{T}_{\mathcal{G}}(\Sigma, \mathcal{X}), \mathcal{T}_{\mathcal{L}}(\Sigma, \mathcal{X}, \mathcal{G}, \Theta, \Lambda)}$ *which is applicable to an initial expression* $\Psi \in \mathcal{T}(\Sigma, \emptyset)$*.*

In the end of this section we argue that this framework is in the same time relatively simple, it covers the *reference model* derivation and it allows for the extensions presented in the next section.
The grammar of terms is designed to cover all mathematical expressions occuring in the proof of the *reference model*

as well as of their generalizations. A term produced by the grammar includes locally all useful information. This avoids the use of external tables and facilitates design of rewriting rules, in particular to take into account the context of subterms to be transformed. It allows also for local definitions, for instance a same name of variable x can be used in different parts of a same term with different meaning, which is useful for instance in integrals.
Each step in the proof consists in replacing parts of an expression according to a known mathematical property. This is well done, possibly recursively, using rewriting rules together with strategies allowing for precise localization.

5. Transformation of Strategies as Second Order Strategies

For a given rewriting strategy representing a model proof, one would like to transform it to obtain a derivation of more complex models. Transforming a strategy $\pi \in \mathcal{S}_{\mathcal{T}(\Sigma,\mathcal{X})}$ is achieved by applying strategies to the strategy π itself. For this purpose, we consider two levels of strategies: the first order ones $\mathcal{S}_{\mathcal{T}(\Sigma,\mathcal{X})}$ as defined in Definition 2, and the strategies of second order in such a way that second order strategies can be applied to first order ones. That is, the second order strategies are considered as terms in a set $\mathcal{T}(\overline{\Sigma},\overline{\mathcal{X}})$ of terms where $\overline{\Sigma}$ stands for the set of second order function symbols and $\overline{\mathcal{X}}$ stands for the set second order rewriting variables. We notice that the (first order) rule $l \rightsquigarrow r$ can be viewed as the term $\rightsquigarrow (l,r)$ with the symbol $\rightsquigarrow$ at the root, and the (first order) strategy $\mu X.s$ viewed as the term $\mu(X,s)$. The set of second order strategies will be denoted by $\overline{\mathcal{S}}_{\mathcal{T}(\overline{\Sigma},\overline{\mathcal{X}})}$; it is built up out of the second order strategy constructors: $\overline{\rightsquigarrow}, \overline{;}, \overline{\oplus}, \overline{Some}, \overline{Child}, \overline{\eta}, \ldots$.
The behavior of the strategies in $\overline{\mathcal{S}}_{\mathcal{T}(\overline{\Sigma},\overline{\mathcal{X}})}$ is similar to the one of first order strategies. In addition, we assume that second order strategies transform first order strategies, to which they are applied, into first order strategies.
In the following example, based on a mathematical property of T stated in [13], we illustrate the extension of an elementary strategy which is a rewriting rule.

Example 8 *For the set $\mathcal{X} = \{i, j, x, x^\sharp, x^1, u, \varepsilon\}$ we define s_1, s_2, s_3, and s_{23} four rewriting rules,*

$$s_1 := T(\frac{\partial u}{\partial x}, x)(x^\sharp, x^1) \rightsquigarrow \frac{1}{\varepsilon}\frac{\partial T(u,x)(x^\sharp, x^1)}{\partial x^1}$$

for $x \in \Omega$ and $(x^\sharp, x^1) \in \Omega^\sharp \times \Omega^1$,

$$s_2 := T(\frac{\partial u}{\partial x_i}, x)(x^\sharp, x^1) \rightsquigarrow \frac{1}{\varepsilon}\frac{\partial T(u,x)(x^\sharp, x^1)}{\partial x_i^1}$$

for $x \in \Omega$ and $(x^\sharp, x^1) \in \Omega^\sharp \times \Omega^1$,

$$s_3 := T(\frac{\partial u}{\partial x}, x)(x^\sharp, x^1) \rightsquigarrow \frac{1}{\varepsilon}\frac{\partial T(u,x)(x^\sharp, x^1)}{\partial x^1}$$

for $x \in \Omega_j$ and $(x^\sharp, x^1) \in \Omega_j^\sharp \times \Omega_j^1$,

$$s_{23} := T(\frac{\partial u}{\partial x_i}, x)(x^\sharp, x^1) \rightsquigarrow \frac{1}{\varepsilon}\frac{\partial T(u,x)(x^\sharp, x^1)}{\partial x_i^1}$$

for $x \in \Omega_j$ and $(x^\sharp, x^1) \in \Omega_j^\sharp \times \Omega_j^1$.

The rule s_1 is encountered in the reference proof, s_2 is a (trivial) generalization of s_1 in the sense that it applies to multi-dimensional regions Ω^1 referenced by a set of variables $(x_i^1)_i$, and s_3 is a second (trivial) generalization of s_1 on the number of sub-regions $(\Omega_j)_j$, $(\Omega_j^\sharp)_j$ and $(\Omega_j^1)_j$ in Ω, $\Omega^\sharp$ and Ω^1. The rule s_{23} is a generalization combining the two previous generalizations. First, we aim at transforming the strategy s_1 into the strategy s_2 or the strategy s_3. To this end, we introduce two second order strategies with $\overline{\mathcal{X}} = \{v, z\}$ and $\overline{\Sigma} \supset \{i,\ j,\ \Omega,\ \Omega^\sharp,\ \Omega^1,$ $Partial, IndexedFun, IndexedVar, IndexedReg\}$,

$$\bar{\Pi}_1 := \overline{OuterMost}(\frac{\partial v}{\partial z} \bar{\rightsquigarrow} \frac{\partial v}{\partial z_i})$$

$$\bar{\Pi}_2 := \overline{OuterMost}(\Omega \bar{\rightsquigarrow} \Omega_j); \overline{OuterMost}(\Omega^\sharp \bar{\rightsquigarrow} \Omega_j^\sharp);$$
$$\overline{OuterMost}(\Omega^1 \bar{\rightsquigarrow} \Omega_j^1).$$

Notice that $\bar{\Pi}_1$ (resp. $\bar{\Pi}_2$) applies the rule $\frac{\partial v}{\partial z} \bar{\rightsquigarrow} \frac{\partial v}{\partial z_i}$ (resp. $\Omega \bar{\rightsquigarrow} \Omega_j$, $\Omega^\sharp \bar{\rightsquigarrow} \Omega_j^\sharp$, and $\Omega^1 \bar{\rightsquigarrow} \Omega_j^1$) at all of the positions [1] of the input first order strategy so that

$$\bar{\Pi}_1(s_1) = s_2 \text{ and } \bar{\Pi}_2(s_1) = s_3.$$

Once $\bar{\Pi}_1$ and $\bar{\Pi}_2$ have been defined, they can be composed to produce s_{23} :

$$\bar{\Pi}_2\bar{\Pi}_1(s_1) = s_{23} \text{ or } \bar{\Pi}_1\bar{\Pi}_2(s_1) = s_{23}.$$

6. Implementation and Experiments

The framework presented in Section 4.4 has been implemented in Maple™. The derivation of the reference model has been implemented using the language *Symbtrans* of strategies presented in [2]. It starts from an input term which is the weak formulation (3) of the physical problem,

$$\int \underline{a}\frac{\partial \underline{u}}{\partial \underline{x}}\frac{\partial \underline{v}}{\partial \underline{x}}\, d\underline{x} = \int \underline{f}\ \underline{v}\ d\underline{x}, \tag{8}$$

where $\underline{a} = \mathtt{Fun}(a, [\underline{\Omega}], [\ ], Known)$, $\underline{u} = \mathtt{Fun}(u,\ [\underline{\Omega}],$ $[\underline{Dirichlet}],\ Unknown)$, $\underline{v} = \mathtt{Fun}(u,\ [\underline{\Omega}],\ [\underline{Dirichlet}],$ $Test)$, $\underline{\Omega} = \mathtt{Reg}(\Omega,\ [1],\ \emptyset,\ \underline{\Gamma},\ n_\Omega)$, $\underline{\Gamma} = \mathtt{Reg}(\Gamma,\ [\ ],\ \emptyset,$ $\perp_{\mathcal{R}},\ \perp_{\mathcal{F}})$, $\underline{Dirichlet} = \mathtt{BC}(Dirichlet,\ \underline{\Gamma}, 0)$ and where the short-cuts of the operators are those of Section 4.2.
The proof is divided into strategies corresponding to blocks of the proof, each ending by some results transformed into rewriting rules used in the following blocks. The rewriting rules used in the strategies are FO-rules and can be classified into the three categories.

- *Usual mathematical rules:* that represents the properties of the derivation and integration operators, such as the linearity, the chain rule, the Green rule, etc,
- *Specialized rules:* for the properties of the two-scale calculus,
- *Auxiliary tools:* for transformations of expressions format that are not related to operator properties such as the rule which transforms $\psi = \varphi$ into $\psi - \varphi = 0$.

[1] Notice the difference with $\overline{TopDown}$ which could not apply these rules at any position.

Table 1 summarizes the number of first order (FO) rules, used in the reference model, by categories.
The reference model has been extended to cover three different kinds of configurations. To proceed to an extension, the new model derivation is established in a form that is as close as possible of the *reference proof.* The grammar and the dependency analyzer should be completed. Then, the initial data is determined, and second order (SO) strategies yielding the generalized model derivation are found and optimized. As it has been already mentioned, $\mathcal{G}$ and Θ have already been designed to cover the three extensions.
The first generalization is to cover multi-dimensional regions, i.e. $\Omega \subset \mathbb{R}^n$ with $n \geq 1$. When $n = 2$, the initial term is

$$\sum_{\underline{i}=1}^{n} \sum_{\underline{j}=1}^{n} \int \underline{a_{ij}} \frac{\partial \underline{u}}{\partial \underline{x_i}} \frac{\partial \underline{v}}{\partial \underline{x_j}} \, d\underline{x} = \int \underline{f} \, \underline{v} \, d\underline{x},$$

where $\underline{\Omega} = \texttt{Reg}(\Omega,\ [1,2],\ \emptyset,\ \underline{\Gamma},\ n_\Omega)$, $\underline{a_{ij}} = \texttt{Indexed}(\texttt{Indexed}(\underline{a},\ \underline{j}),\underline{i})$, $\underline{i} = \texttt{Var}(i,\ \underline{I})$, $\underline{I} = \texttt{Reg}(I,\ [1,2],\ \emptyset,\ \perp_{\mathcal{R}},\ \perp_{\mathcal{F}})$ and the choice of the test function is trivially deduced. Then, the model derivation is very similar to this of the *reference model*, see [11], so much so it is obtained simply by applying the SO strategy $\bar{\Pi}_1$ defined in Example 8.
The second generalization transforms the *reference model* into a model with several adjacent one-dimensional regions (or intervals) $(\Omega_k)_{k=1,..,m}$ so that Ω is still an interval i.e. $\Omega \subset \mathbb{R}$. For $m = 2$, the initial term is the same as (8) but with $\underline{\Omega} = \texttt{Reg}(\Omega, [1],\ \{\underline{\Omega_1}, \underline{\Omega_2}\},\ \underline{\Gamma}, n_\Omega)$, $\underline{\Omega_1} = \texttt{Reg}(\Omega_1, [1], \emptyset,\ \underline{\Gamma_1}, n_{\Omega_1})$, and $\underline{\Omega_2} = \texttt{Reg}(\Omega_2,\ [1], \emptyset,\ \underline{\Gamma_2}, n_{\Omega_2})$. The two-scale geometries, all variables, all kind of functions and also the two-scale operators are defined subregion by subregion. All definitions and properties apply for each subregion, and the proof steps are the same after spliting the integrals over the complete region Ω into integrals over the subregions. The only major change is in the fourth step where the equality $u_1^0 = u_2^0$ at the interface between Ω_1 and Ω_2 which is encoded as transmission conditions in the boundary conditions of u_1^0 and u_2^0.
The third extension transforms the multi-dimensional model obtained from the first generalization to a model related to thin cylindrical regions, in the sense that the dimension of Ω is in the order of ε in some directions $i \in I^\natural$ and of the order 1 in the others $i \in I^\sharp$ e.g. $\Omega = (0,1) \times (0,\varepsilon)$ where $I^\natural = \{2\}$ and $I^\sharp = \{1\}$. The boundary Γ is split in two parts, the lateral part Γ_{lat} and the other parts Γ_{other} where the Dirichlet boundary conditions are replaced by homogeneous Neuman boundary conditions i.e. $\frac{du^\varepsilon}{dx} = 0$. In this special case the integrals of the initial term are over a region which size is of the order of ε so it is required to multiply each side of the equality by the factor $1/\varepsilon$ to work with expressions of the order of 1. Moreover, the macroscopic region differs from Ω, it is equal to $\Omega^\sharp = (0,1)$ when the microscopic region remains unchanged. With these main changes in the definitions and the preliminary properties, the proof steps may be kept unchanged.
The mathematical formulation of the second and third extensions has been derived. This allows for the determination of the necessary SO-strategies, but they have not been implemented nor tested. To summarize the results about the principle of extension of strategies, we show its benefit through some statistics. In particular the main concerned is the reusability and the extensibility of existing strategies. Table 2 shows an estimate of the number of new FO-rules for the three extensions in each category and for the first four blocks.
Table 3 shows the number of SO-strategies used in each extension. Finally, Table 4 shows, the ratio of the modified FO-rules and the ratio of the modified FO-strategies. The reusability ratio is high since most of the FO-strategies defined in the skeleton model are reused. Besides very little number of SO-strategies is used in the extensions. This systematic way of the generation of proofs is a promising path that will be further validated within more complex configurations for which the proofs can not obtained by hand. In the future, we plan to introduce dedicated tools to aid in the design of composition of several extensions.

	Usual Rules	Special Rules	Aux. Tools
Skeleton	53	14	28

Table 1: The number of first order rules used in the reference model.

	Usual Rules	Special Rules	Aux. Tools
Multi-Dim	6	0	4
Thin-Reg	2	0	0
Multi-Reg	3	0	0

Table 2: The number of first order rules used in the three extensions.

	Usual Rules	Special Rules	Aux. Tools
Multi-Dim	9	2	3
Thin-Reg	0	0	0
Multi-Reg	1	0	0

Table 3: The number of second order strategies used in the extension of proofs.

7. Connection to FEM Software through an Example

In [11], a model of the mechanical behavior of elastic periodic cantilever arrays was established and studied in [9], [10], [7]. Its derivation follows the same steps as in the reference proof but in addition, it takes into account several features: multi-dimensional regions, thin regions, vector valued solutions, asymptotic scaling of the solution and strongly heterogeneous coefficients. So, it can be derived by composing those elementary extensions. Its complete composition have not yet been build in MEMSALab, so here we only present its transfer into a finite element

Input model	Resulting mod.	% FO-rules	% FO-strateg.
Reference	Multi-Dim.	16.6%	5%
Multi-Dim.	Thin	0	0
Thin	Multi-Reg.	0	2.5%

Table 4: The ratio of modified FO-rules and FO-strategies.

software in a manner that will be generalized in the future. We adopt the formulation of Section 2 in [7], that is posed in a rectangle $(x_1^\sharp, x_2^1) \in (0, L_B) \times (0, L_C^*)$ where the parameters L_B and L_C^* represent respectively the base length in the macroscale direction $x_1^\sharp$ and the cantilever scaled length in the microscale variable x_2^1. For simplicity, we denote the coordinates $x_1^\sharp$ and x_2^1 by x and y. The base is modeled by the line $x \in (0, L_B)$ located at $y = 0$. At each x corresponds a clamped cantilever oriented in the direction $y \in (0, L_C^*)$. Only the bending displacement is considered in the system motion. The base is governed by an Euler-Bernoulli beam equation with two kinds of distributed forces. One is exerted by the attached cantilevers and the other is an external force denoted by f_B. The bending displacement, the bending coefficient and the width being denoted by w_B, R^B and ℓ_C, the base governing equation states as

$$R^B \partial_x^4 w + \ell_{\mathbf{C}} R^C \partial_y^3 w_C = f_B. \tag{9}$$

The base is assumed to be clamped, i.e. with boundary conditions,

$$w_B = \partial_x w_B = 0 \tag{10}$$

at both ends. The motion of the cantilevers is governed by an infinite number of Euler-Bernoulli equations distributed along the x-direction,

$$R^C \partial_{y\cdots y}^4 w_C = f_C \tag{11}$$

where w_C, f_C and R^C represent the bending displacement, the distributed force and the bending coefficient. The cantilevers are clamped in their base and free at their other end, so the boundary conditions are

$$\begin{cases} w_C = w_B \text{ and } \partial_y w_C = 0 & \text{at } y = 0 \\ \partial_{yy}^2 w_C = \partial_{yyy}^3 w_C = 0 & \text{at } y = L_C^*. \end{cases} \tag{12}$$

In total, the model is governed by (9-12). In the following, we describe its implementation, for N cantilevers, in a FEM software package using a two-dimensional geometry, see Figure 1. To do so, the fields w_B and w_C are taken independent of y and x respectively, and the problem (11, 12) is solved for the N cantilevers. The contribution $\partial_y^3 w_C$ in the base equation is therefore added as an internal term. The implementation is described for a software package as COMSOL™ which is based on a mixed finite element method with lagrangians on the form

$$\mathcal{L}(u, \mu) = \frac{1}{2} a(u, u) - l(u) + b(u, \mu).$$

We refer to [3] for a presentation of this method built with two bilinear forms $a(u, u)$, $b(u, \mu)$ and a linear form $l(u)$ defined for $u \in U$ and $\mu \in M$ where U and M are two functional spaces. Here, they are restricted to the form necessary for the above model posed on a bidimensional domain Ω with boundary Γ,

$$\begin{aligned} a(u, u) &= \int_\Omega L.\nabla u \, dx, \; l(u) = \int_\Omega F \, u \, dx \\ \text{and } b(u, \mu) &= \int_\Omega R_\Omega \, \mu_\Omega \, dx + \int_\Gamma R_\Gamma \, \mu_\Gamma \, dx \end{aligned} \tag{13}$$

but for broad possible choices of vector valued functionals $L(\nabla u, u)$, constraints $R_\Omega(\nabla u, u)$, $R_\Gamma(\nabla u, u)$ and right-hand sides $F(\nabla u, u)$. The functions μ_Ω, μ_Γ are lagrange multipliers associated to the constraints. The associated equations are

$$\begin{aligned} -\operatorname{div}(L(u)) + (\partial_u R_\Omega(u))^*(\mu) = F \text{ in } \Omega \\ \text{and } L(u).n = 0 \text{ or } R_\Gamma(\nabla u, u) = 0 \text{ in } \Gamma \end{aligned} \tag{14}$$

where n is the outward unit normal to Γ.

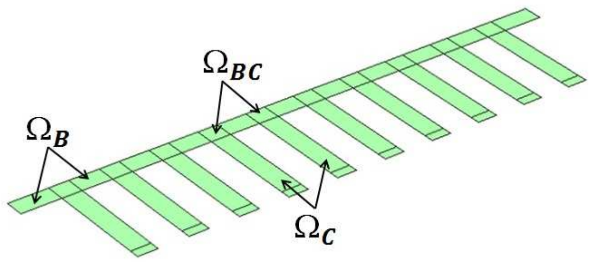

Fig. 1: Geometry of an array of cantilevers

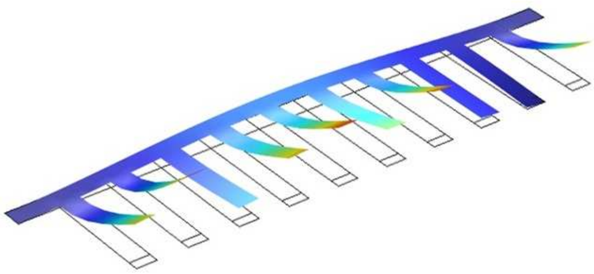

Fig. 2: Simulation results for an array of cantilevers

Here, the domain Ω is split into three pieces $\Omega_B \cup \Omega_{BC} \cup \Omega_C$, see Figure 1. The fourth order equations (9, 11) cannot be directly written in the above framework. They are implemented as a system of first order equations

$$\begin{gathered} \partial_x w_B = m_1, \; \partial_x m_1 = m_2, \; \partial_x m_2 = m_3, \\ \text{and } \partial_x m_3 + \ell_{\mathbf{C}} R^C n_3 = f_B \text{ in } \Omega_B \cup \Omega_{BC}, \\ \partial_y w_C = n_1, \; \partial_y n_1 = n_2, \; \partial_y n_2 = n_3, \; \partial_y n_3 = f_C \text{ in } \Omega_C, \\ \text{and } \partial_y n_3 = 0 \text{ in } \Omega_{BC}. \end{gathered}$$

The fields w_B, w_C are defined in Ω, m_1, m_2, m_3 are defined in $\Omega_B \cup \Omega_{BC}$, n_3 is defined in $\Omega_C \cup \Omega_{BC}$ and n_1, n_2 are defined in Ω_C. The corresponding equations are $L = 0$ in all parts and for all fields. Posing $\Gamma_B = \{(x,y) \in \Omega \mid x \in \{0, L_B\}\} \times (0, L_C^*)$, $\Gamma_{BC} = \{(x,y) \in \Omega \mid y = 0\}$, $\Gamma_C = \{(x,y) \in \Omega_C \mid y = L_C^*\}$, and removing the indices Ω and Γ of R since there is no ambiguity, the related equations are

$$F_{w_B,w_C} = \begin{pmatrix} \partial_x w_B - m_1 \\ w_C \end{pmatrix}, \ R_{w_B} = \partial_y w_B \text{ in } \Omega_B \cup \Omega_{BC},$$
$$F_{w_B,w_C} = \begin{pmatrix} \partial_y w_B \\ \partial_y w_C - n_1 \end{pmatrix}, R_{w_C} = \partial_x w_C \text{ in } \Omega_C,$$
$$R_{w_B} = w_B \text{ at } \Gamma_B, R_{w_C} = w_C \text{ at } \Gamma_{BC},$$

$$F_{m_1,m_2,m_3} = \begin{pmatrix} \partial_x m_1 - m_2 \\ \partial_x m_2 - m_3 \\ \partial_x m_3 + \ell_{\mathbf{C}} R^C n_3 - f_B \end{pmatrix},$$
$$R_{m_1,m_2,m_3} = \begin{pmatrix} \partial_y m_1 \\ \partial_y m_2 \\ \partial_y m_3 \end{pmatrix} \text{ in } \Omega_B \cup \Omega_{BC},$$
$$R_{m_1} = m_1 \text{ at } \Gamma_B,$$

$$F_{n_1,n_2} = \begin{pmatrix} \partial_y n_1 - n_2 \\ \partial_y n_2 - n_3 \end{pmatrix}, R_{n_1,n_2} = \begin{pmatrix} \partial_x n_1 \\ \partial_x n_2 \end{pmatrix} \text{ in } \Omega_C,$$
$$R_{n_1} = n_1 \text{ at } \Gamma_{BC}, R_{n_2} = n_2 \text{ at } \Gamma_C,$$
$$F_{n_3} = \partial_y n_3 - f_C, R_{n_3} = \partial_x n_3 \text{ in } \Omega_C, \ R_{n_3} = n_3 \text{ on } \Gamma_C,$$
$$F_{n_3} = \partial_y n_3, R_{n_3} = \partial_x n_3 \text{ in } \Omega_{BC}.$$

Figure 2 illustrates this implementation for forces applied to lines closed to the cantilever ends. It is noticed that this model simulation dramatically reduces computation time and memory use compared to direct three-dimensional simulations.

8. Acknowledgements

This work is partially supported by the European Territorial Cooperation Programme INTERREG IV A France-Switzerland 2007-2013 and by the French National Programme LABEX ACTION.

References

1. F. Baader and T. Nipkow. *Term rewriting and all that.* Cambridge University Press, 1998.
2. W. Belkhir, A. Giorgetti, and M. Lenczner. A symbolic transformation language and its application to a multiscale method. Submitted. December 2010, http://arxiv.org/abs/1101.3218v1.
3. F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods*. Springer-Verlag, 1991.
4. J. Casado-Díaz. Two-scale convergence for nonlinear Dirichlet problems in perforated domains. *Proc. Roy. Soc. Edinburgh Sect. A*, 130(2):249–276, 2000.
5. D. Cioranescu, A. Damlamian, and G. Griso. The periodic unfolding method in homogenization. *SIAM Journal on Mathematical Analysis*, 40(4):1585–1620, 2008.
6. H. Cirstea and C. Kirchner. The rewriting calculus — Part I and II. *Logic Journal of the Interest Group in Pure and Applied Logics*, 9(3):427–498, May 2001.
7. S. Cogan, H. Hui, M. Lenczner, E. Pillet, N. Ratier, and Y. Yakoubi. Modelling, identification and control of a micro-cantilever array. *Micro, Nanosystems and Systems on Chips: Modeling, Control, and Estimation*, pages 56–63, 2010.
8. O. Pironneau F. Hecht and K. Ohtsuka. *Freefem++: Manual*. 2012.
9. H. Hui, M. Lenczner, E. Pillet, and S. Cogan. A two-scale model for one-dimensional arrays of cantilevers and its verification. *Mechatronics*, 22(5):538 – 543, 2012.
10. M. Lenczner. Multiscale model for atomic force microscope array mechanical behavior. *Applied Physics Letters*, 90(9):091908–091908, 2007.
11. M. Lenczner and R. C. Smith. A two-scale model for an array of AFM's cantilever in the static case. *Mathematical and Computer Modelling*, 46(5-6):776–805, 2007.
12. D. Marino and T Millstein. A generic type-and-effect system. In *Proceedings of the 4th international workshop on Types in language design and implementation*, TLDI '09, pages 39–50, New York, NY, USA, 2009. ACM.
13. Bin Yang, Walid Belkhir, and Michel Lenczner. Computer-aided derivation of multi-scale models: A rewriting framework. *arXiv preprint arXiv:1302.2224*, 2013.