Computer–Aided Multiscale Model Derivation for MEMS Arrays*

B. Yang[1,2], W. Belkhir[3], R. N. Dhara[2], M. Lenczner[2] and A. Giorgetti[3]
[1]Department of Applied Mathematics,Northwestern Polytechnical University
710129 Xi'an Shaanxi, China
[2]FEMTO-ST, Département Temps-Fréquence, University of Franche-Comté,
26 chemin de l'Epitaphe, 25030 Besançon Cedex, France
[3]LIFC, University of Franche-Comté,
16 route de Gray, 25030 Besançon Cedex, France
Email: bin.yang@femto-st.fr, michel.lenczner@utbm.fr, rajnarayan.dhara@femto-st.fr,
walid.belkhir@lifc.univ-fcomte.fr, alain.giorgetti@univ-fcomte.fr

Abstract

We are currently developing software dedicated to multiscale and multiphysics modeling of arrays of micro and nanosystems. Unlike traditional software that is based on models built once and for all, here this is the software that constructs models. It is based on the mathematical Two-Scale Transform, a technique for asymptotic methods, together with formal specification and verification techniques in computer science, combining formal transformations and term rewriting and type theory. We aim at taking into account a wide range of geometries combining thin structures, periodic structures with the possibility of multiple nested scales and any combination of these features. In this paper we present the principle of our methods and our first results.

1. Introduction

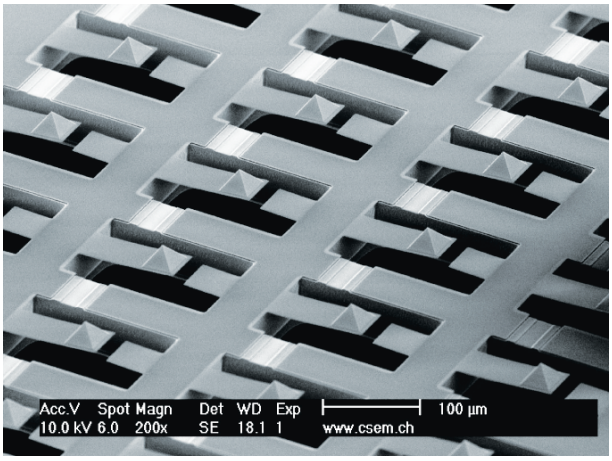


Fig. 1: An array of micro-levers for parallel AFM applications. A courtesy of A. Meister from CSEM Switzerland.

Simulation software available on the market does not take into account large arrays of micro and nanosystems such as that shown in Figure 1. From the point of view of geometric features, the software we develop aims at filling this gap. In fact, it will aid to design multiphysics systems with high contrast in equation coefficients and complex geometries including thin structures, periodic structures with possibly many nested scales, and combinations of these. The produced simulations should be light enough to be inserted into parameter identification and optimization loops. The software is based on multiscale models, especially on models obtained by asymptotic methods. An asymptotic model is derived from a system of partial differential equations (PDEs) when taking into account that at least one parameter is very small, as for a thin structure (small thickness) or for a periodic structure (small ratio of cell size to global size). The resulting model, another system of PDEs, is obtained by taking the mathematical limit of the nominal model, in some energy sense, when the small parameters are going to zero. This approach provides a reasonably good approximation and has the advantage to be rigorous and systematic which are factors of reliability. The resulting PDEs can be implemented in simulation software as for instance a finite element based simulator, and simulations turn to be fast as needed. We quote that the literature in this field is vast and that a large number of techniques have been developed for a large variety of geometric features and of physical phenomena. However, none of them have been implemented in a systematical approach to render it available to engineers as a design tool. In fact, each published paper focus on a special case regarding geometry or physics, and very few works are considering a general picture. In our software, we treat the problem of systematic implementation of asymptotic methods in the software by implementing the construction of models rather than the models themselves. This approach would cover many situations from a small number of bricks. It combines two types of tools, namely the mathematical Two-scale Transform originally introduced in [1, 2, 3] (also referred as the Unfolding Method) to model asymptotic periodic structures together with formal specification techniques in computer science, combining term rewriting, λ-calculus and types [4, 5, 6, 7, 8]. The software is written in the symbolic computation language Maple$^{\mathsf{TM}}$. Regarding the Two-Scale Transform technique, we have extended its domain of application to cover in the same time homogenization of periodic media, see for instance [9], and methods for asymptotic analysis for thin domains, see [10]. Compared to other techniques, ours requires more modular calculations and avoids any non-constructive proof.

In the paper we present each aspect of our method as well as our first results. It is organized as follows. Sec-

*This work is partially supported by the European Territorial Cooperation Programme INTERREG IV A France-Switzerland 2007-2013.

tion 2. introduces the mathematical tools used in model derivation and the example under study. In Section 3. we detail our design methodology, the architecture of our software and its theoretical foundations. Section 4. explains the model derivation which has been validated by the software and presents a comparison between the numerical simulations of the starting model and the homogenized one. The errors estimation is calculated and discussed.

2. The mathematical tools used for model derivation

To date, the program covers only the elliptic second order PDE,

$$-\sum_{i,j=1}^{d} \frac{\partial}{\partial x_i}(a_{ij}(x)\frac{\partial}{\partial x_j}u(x)) = f(x), \qquad (1)$$

posed in a region Ω in $\mathbb{R}^d$ consisting of subregions each of which may have a periodic structure, be thin or combining these two features. The coefficients a_{ij} of the equation may also be periodic. The boundary conditions can be Dirichlet conditions $u = g$ on a part Γ_1 of the boundary $\partial\Omega$ for a given g, and Neumann conditions $\sum_{i,j=1}^{d} n_j a_{ij}(x)\frac{\partial}{\partial x_j}u = h$ on the other part Γ_2 of the boundary with outward unit vector $(n_1, .., n_d)$ for a given h. This partial differential equation appears in a number of physical models such as the stationary diffusion equation, the equation of electrostatics or a model for elastic beam in torsion. We assume that all small parameters of the problem are expressed in terms of a single one denoted by ε.

Example: *For the sake of illustration, we present the example of thermal equilibrium of the AFM array, shown in Figure 1, connected to a thin bar and placed in a vacuum, see Figure 2. The bar thickness is in the same range as the cell sizes. A heat source is placed in each cantilever to model the presence of actuations by thermo-elastic coupling. The heat flux at all boundaries is vanishing ($h = 0$) except at the bar end and at the left and bottom sides of the lever array where a temperature is imposed at $g = 0°C$. Modelling is done in two dimensions, so that the matrix a represents the two-dimensional thermal conductivity and f the two-dimensional density of thermal source.*

Here, we do not detail the mathematical steps of construction of asymptotic model, but we will just review the mathematical objects that are manipulated in order to explain, in Section 3, how they are treated in terms of formal computing. Several proofs have been published in [11], [12], [3], [13], and [14]. Here we adapt the proofs of the paper [13] where an effort has been made to formulate proofs in a modular form and to avoid any non-computational steps. The steps in this formal method are rigorously specified at a high level of generality, that make them independent of the domain geometry and applicable to other equations. The model derivation starts from the variational formulation associated to the classical formulation boundary value problem (1). Find $u \in V$ such that

$$\sum_{i,j=1}^{d} \int_\Omega a_{ij}\frac{\partial u}{\partial x_i}\frac{\partial v}{\partial x_j}\,dx = \int_\Omega f\,v\,dx \qquad (2)$$

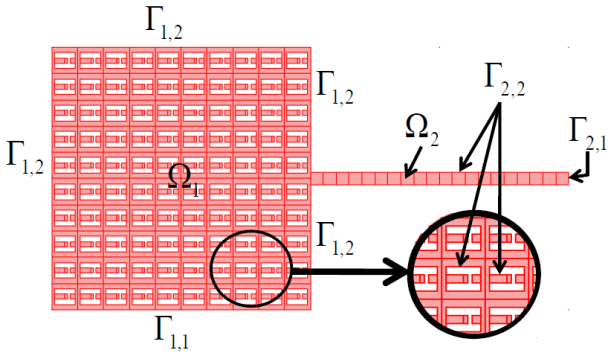


Fig. 2: Geometry and boundary conditions for an array of cantilevers connected to a beam

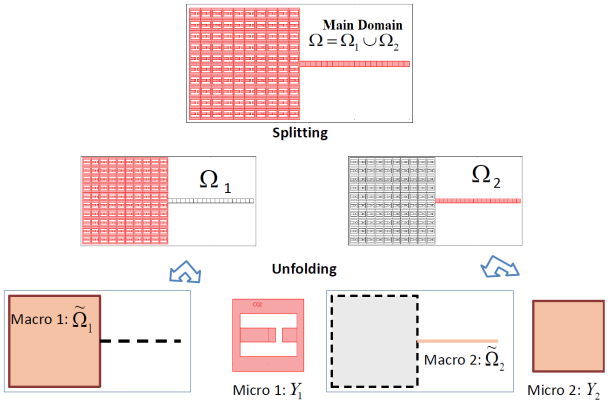


Fig. 3: Flowchart representing the Two-Scale Transformation of the region Ω

for all $v \in V$ where V represents the set of admissible functions

$$V = \{u \mid u = 0 \text{ on } \Gamma_1\}.$$

We observe that we have dropped the question of function regularity that should be taken into account in the set of admissible functions because this has not yet been taken into account in the implementation.

The variational formulation requires the definition of regions as Ω and their boundaries as Γ_1 and Γ_2, of variables as $x = (x_1, .., x_d)$, of functions as $w(x)$ and $v(x)$, of set of functions as V, and then of operations associated with them. Beyond simple operations such as addition or multiplication between functions, it is required to define formally the integral over a region and the sum over a set of indices. Once these mathematical objects have been defined, we also introduces properties as Green's formula, the natural extension of the integration by part formula,

$$\int_\Omega \frac{\partial w}{\partial x_i} v\,dx = -\int_\Omega w\frac{\partial v}{\partial x_i}\,dx + \int_{\partial\Omega} w\,v\,n_i\,dx,$$

that may be used during the proof.

On the other hand, the model derivation relates to the Two-Scale Transform whose principle is shown in Figure 3.

It consists of a phase of region splitting into several subregions ($\Omega_1 \cup \Omega_2$ in the example) requiring different treatments. If a subregion is not subject to any simplification when its processing stops. Otherwise, it is partitioned into a family of periodic cells, even if it does not undergo a simplification to the periodicity, and is transformed (by a process not detailed here) into the product of so-called *macroscopic* and *microscopic* regions. The latter contains all the information on the microstructure of the sub-region while the first is only used to locate the position of the cells so that in the case of a thin structure its dimension is that of the neutral axis or plane. Figure 3 illustrates this construction process on the example where Ω_1 is a periodic region and Ω_2 a thin region. Their image pairs are denoted by $(\widetilde{\Omega}_1, Y_1)$ and $(\widetilde{\Omega}_2, Y_2)$ respectively. By composition with the above operation, a function defined on a sub-region is transformed into a function defined on the product of its macroscopic and microscopic regions. This defines the linear operator T named the Two-Scale Transform. For demonstration purposes, we defined the adjoint T^* of T. Since T is a linear operator which transforms a function defined on a subdomain Ω_1 into a product $\widetilde{\Omega}_1 \times Y_1$, conversely its adjoint for the L^2-inner product is defined from $\widetilde{\Omega}_1 \times Y_1$ to Ω_1 by the equality

$$\int_{\Omega_1} T^* w \; v \; dx = \int_{\widetilde{\Omega}_1 \times Y_1} w \; Tv \; dxdy$$

for w and v respectively defined in $\widetilde{\Omega}_1 \times Y_1$ and Ω_1. To derive the approximate model, we require the concept of approximation of a function v by another function w in the sense of the small parameter ε, that we denote by $v = w + O(\varepsilon)$, together with the algebraic rules on $O(\varepsilon)$. Finally, the linearity of some operators is used repeatedly.

3. Software architecture

This section presents our software design methodology and its theoretical foundations. The software will implement the sequence of modelling steps depicted in Figure 4. It will be named MEMSALab, for "MEMS Arrays Laboratory". The software will transform a nominal multiphysic model, composed of a geometry and PDEs, into a multiscale (MS) asymptotic model approximating the initial one. This transformation depends on specifications about periodic and thin parts of the model, which are formulated by the designer.

The software will be composed of modules. The modules composing the software kernel are represented in the grey part of Figure 4. The Data Structure Constructor is in charge of importing the nominal model from a finite element analysis and simulation software package of physics and engineering applications, e.g. COMSOL. Conversely, a Geometry & Equation Generator will export the derivated multiscale model towards COMSOL after being adapted so that the simulation with COMSOL can be done effectively. Finally the results of this simulation will be brought back to the physical domain by means of a Inverse MS Transform module. The modelling step named Multiscale Data Structure Constructor is essentially devoted to the extension of the internal data structure with additional multiscale data. Its input is a symbolic representation of the geometry produced by the Data Structure Constructor from the numerical data from COMSOL. Data structures are detailed in Section 3.1. The purpose of the module named Multiscale (MS) Model Constructor is to automatically construct the asymptotic multiscale model from the multiscale data structures. Yet this purpose is not completely fulfilled since it requires a unification of multiscale methods. However two parts of this module are already built and tested, namely a symbolic transformation package and a type-checker. Sections 3.2 and 3.3 respectively detail these two basic blocks and their theoretical foundations.

3.1 Data structures

The multiscale data structures are designed and implemented to be complete, flexible and able to describe complex geometries, but without redundancies. Four kinds of data structures with increasing complexity have been defined.

The simplest ones are formal expressions in the host language Maple. These formal expressions are composed of symbols representing mathematical operations (such as sum, product, ...) or operators (such as partial derivation or integration) that are applied in a hierarchical way on similar expressions and on atomic symbols, such as region and (mathematical) variable names. For instance, Eq. (2) is represented by a single Maple expression with $=$ as top symbol. Most of these symbols are usefully predefined in Maple, but we adapt some of them and define new symbols (e.g. the two-scale operators). For instance, the Maple-defined symbol `int` for integration is not general enough to be used. We introduce a distinct integration symbol `Integral` for integration over a region and encode its semantics by equalities (data structures of the fourth kind, see below for details).

The second kind of data structures describes geometrical regions and their boundaries, mathematical functions and admissible functional spaces. They are stored in Maple arrays. Functional spaces are also formalized in the multiscale data structures and can be used and modified by the Multiscale Model Constructor.

Data structures of the third level are formal equalities (pairs of Maple expressions) encoding equational assertions such as boundary hypotheses. The fourth level of data structures represents the algebraic properties of the new operators (two-scale operators, integration over a region, etc) as systems of oriented equalities, a.k.a. rewriting strategies, as detailed in Section 3.2.

3.2 Symbolic transformation package

Most of the transformations performed by multiscale methods are based on quasi-systematic derivations of equalities. The classical way to automate these derivations is to consider mathematical expressions as symbolic ones, called *terms*, and to give a computational meaning to mathematical equations. The idea is to orient the equality $x = y$ into a *rewrite rule* $x \to y$ which states that every occurrence of an instance of x can be replaced with the corresponding instance of y. Consequently equational derivations are reduced to a series of term rewritings.

Algebraic computation and term rewriting are two research domains with strong similarities. Both are sep-

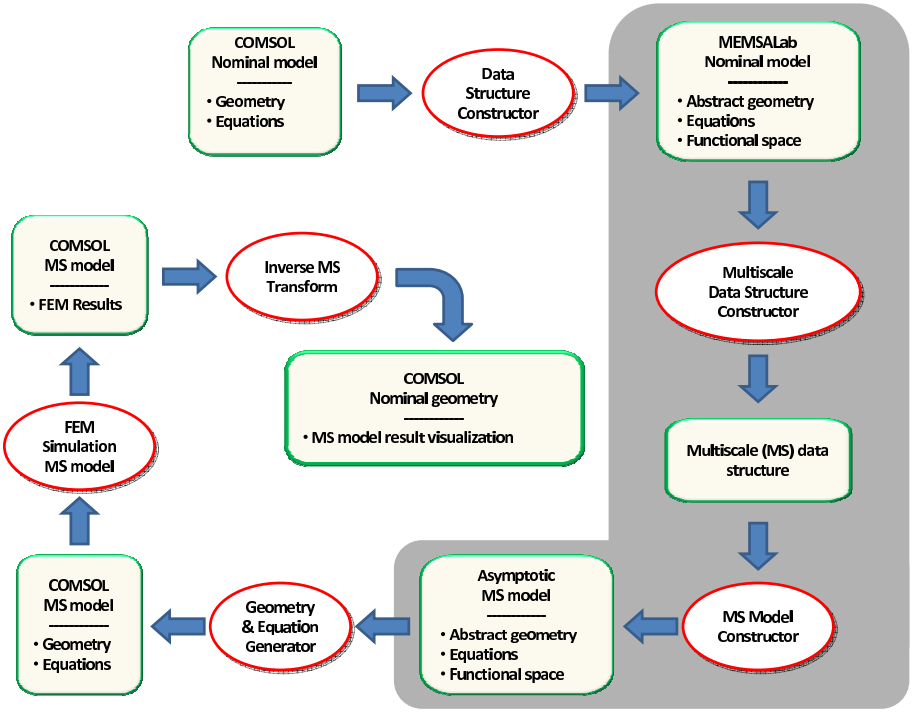


Fig. 4: Flowshart of the modelling steps

arately well-studied but there are only few works about the combination of algebraic computation and term rewriting [4, 15, 16, 6].

Term rewriting provides a theoretical and computational framework which is very useful to express, study and analyze a wide range of complex dynamic systems [17, 18]. It is characterized by the repeated transformation of a data object such as a word, term or graph. Transformations are combinations of rules specifying how to transform an object into another one when it follows a specific pattern. Rule application can be restricted by additional conditions and rules can be combined by specifying strategies controlling the order and the way rules are repeatedly applied. Term rewriting is used in formal semantics to describe the meaning of programming languages and more generally in computer science to describe program transformations and to perform automated reasoning. It is central in systems where the notion of *rule* is explicit such as expert systems, algebraic specifications, etc.

The computer algebra system Maple™ is widely used in the symbolic computation community and we have retained it for a prototypal implementation of the algorithmic aspects of multiscale methods. Maple is a suitable language for combining function-based and rule-based symbolic transformations. Unfortunately, its rewriting function does not provide the user with control on how and where rules must be applied. We have implemented a Maple package, named `symbtrans`, extending Maple with *rewriting-based programming* where rules, strategies and usual Maple functions can be freely combined. The reader is invited to see [6] for details. Let us briefly explain the theoretical foundations of this package.

The transformation language provided by the `symbtrans` package is deliberately an adaptation for Maple of popular strategy languages such as ρ-log [5] or TOM [19]. The common theoretical basis for all these works is the ρ-cube [7], an extension of the λ-cube where rewriting rules extend function abstractions, with the same hierarchy of type systems. The current version of our transformation language corresponds to the untyped version of the ρ-cube, called the ρ-calculus.

Among other advantages our transformation language smoothly integrates with standard Maple functions. Consequently, the Maple programmer learns it quickly and is free to mix function- and rule-based programming styles. Moreover all the features of her development environment (such as refactoring, code completion, dependency analyses, etc) are preserved for free.

Each mathematical tool introduced in Section 2., namely the integral and PDE calculus, the algebraic rules on $O(\varepsilon)$ and the algebraic properties of the two-scale operators, is expressed in the software by means of the notion of strategy. Roughly speaking, a strategy is a combination of rewriting rules. It expresses the way the rules are applied. Thanks to the notion of strategy the formal proof is close to the mathematical one in terms of size and nature of steps.

3.3 Type-checker

This section shows the interest of a typing discipline for the formalization of multiscale methods. In computer science type systems complete grammatical descriptions of programming languages with additional contextual conditions of *well-formedness*. For example, the fact that a mathematical expression under an integration sign has to be integrable is a contextual information that is better detected

by well-formedness rules than by grammatical rules. The set of rules that relate the type of a mathematical expression with the one of its sub-expressions is called a *type system*. The reader is invited to see [8] for more information on type theory.

We propose a first type system which basically corresponds to the Basic Extended Simple Type Theory [20]. In this system region names (Ω, Ω_1, Ω_2, $\widetilde{\Omega}_1$, $\widetilde{\Omega}_2$, Y_1 and Y_2 in the example) are considered as *type constants*, i.e. atomic type names. The other type constants are $*$ for Booleans and $\mathbb{R}$ for real numbers. The notation $e : A$ asserts that e is an expression of type A. The type $A \to B$ denotes the set of functions whose domain is A and co-domain is B. Moreover, any Cartesian product of region names also forms a type. For instance, the functions w defined on $\widetilde{\Omega}_1 \times Y_1$ in Section 2. are expressions of type $\widetilde{\Omega}_1 \times Y_1 \to \mathbb{R}$. It is denoted by $w : \widetilde{\Omega}_1 \times Y_1 \to \mathbb{R}$.

For any region Ω, any variable x such than $x : \Omega$ and any expression e such that $e : A$ or $e : A \to B$, where A is a Cartesian product composed of other regions than Ω, the expression $\int_\Omega e\, u\, dx$ (where u is any expression) can be rewritten as $e \int_\Omega u\, dx$ and the expression $\partial e/\partial x$ as 0. Such simplification rules of major importance are made possible thanks to the typing information. The type checking process requires that the user declares the type of all the mathematical objects used in the proof (via the data structures described in Section 3.1). These objects are the constants, variables, functions, $O(\varepsilon)$ terms, etc. For all the possible software inputs the types of the standard mathematical operators such as derivation and integration are predefined in the type system.

For each region Ω we can define the type of the Two-Scale Transform T_Ω of functions defined on Ω as $(\Omega \to \mathbb{R}) \to (((\nu(\Omega) \times \mu(\Omega)) \to \mathbb{R})$ where $\nu(\Omega)$ (resp. $\mu(\Omega)$) denotes the macroscopic (resp. microscopic) region of Ω. But it is more generic and compact to consider a single Two-Scale Transform $T : (\alpha \to \mathbb{R}) \to (((\nu(\alpha) \times \mu(\alpha)) \to \mathbb{R})$, where α is a *type variable* [20]. In the present case, this variable can only be replaced by any region name and this replacement produces an instance of this *polymorphic* operator. Its polymorphic adjoint is $T^* : (((\nu(\alpha) \times \mu(\alpha)) \to \mathbb{R}) \to (\alpha \to \mathbb{R})$.

For multiscale methods, regularity conditions and splitting operations could additionnally be checked by a richer type system. Regularity conditions for a mathematical function (i.e. derivability and integrability) are usually described by its membership to an adequate functional space. Splitting operations introduce subregions and unions of regions. In the present type system regions are considered as types, but simple type theories neither support subtyping nor type union. Two more complex type theories could be good candidates for an extension, namely dependent type theory [8, Section 6] and subtyping theories. We are currently working on these extensions.

4. Validation

When applying our software to Eq. (2), we specify that the Two-Scale Transforms to be used are T^1 and T^2 corresponding to Fig. 3. The two-scale model provided by the software includes the microscopic equation posed in the Y_1 cell with the coefficient $a^1 = T^1 a$, the function space $W^1 = \{v \text{ defined in } Y_1 \mid v \text{ is } Y_1\text{-periodic}\}$ and the variational formulation: Find $w_i^1 \in W^1$ such that

$$\sum_{k,\ell=1}^{2} \int_{Y_1} a^1_{k\ell} \frac{\partial w_i^1}{\partial y_k} \frac{\partial v}{\partial y_\ell}\, dy = -\sum_{k,\ell=1}^{2} \int_{Y_1} a^1_{k\ell} \delta_{ik} \frac{\partial v}{\partial y_\ell}\, dy \quad (3)$$

for all $v \in W^1$. It also includes a problem posed in the microscopic cell of the thin part Y_2 even if the latter has not the periodicity feature. This problem is written in a similar manner not detailed here and leads to the definition of a^2 and w_i^2. It follows the definition of homogenized thermal diffusion coefficient and density of heat source per unit area in the first macroscopic part $\widetilde{\Omega}_1$,

$$a^H_{ij} = \sum_{k,\ell=1}^{2} \frac{1}{|Y_1|} \int_{Y_1} a^1_{k\ell} (\delta_{ik} + \frac{\partial w_i^1}{\partial y_k})(\delta_{j\ell} + \frac{\partial w_j^1}{\partial y_\ell})\, dy$$

$$\text{and } f^H = \frac{1}{|Y_1|} \int_{Y_1} T^1 f\, dy.$$

Similar formula yields a^H and f^H in $\widetilde{\Omega}_2$ the second macroscopic part. The two-scale model also includes the macroscopic problem posed on $\widetilde{\Omega}_1 \cup \widetilde{\Omega}_2$ the macroscopic region with $V^0 = \{v \text{ defined in } \widetilde{\Omega}_1 \cup \widetilde{\Omega}_2 \mid v = 0 \text{ on } \Gamma_1 \cap (\widetilde{\Omega}_1 \cup \widetilde{\Omega}_2)\}$ the macroscopic admissible function space and the variational formulation: Find $u^0 \in V^0$ such that

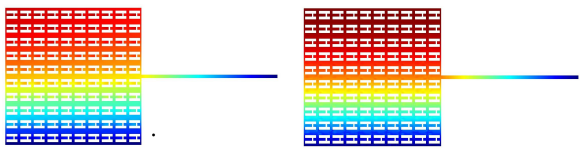
$$\sum_{i,j=1}^{2} \int_{\widetilde{\Omega}_1} a^H_{ij} \frac{\partial u^0}{\partial \widetilde{x}_i^1} \frac{\partial u^0}{\partial \widetilde{x}_j^1}\, d\widetilde{x}^1 + \int_{\widetilde{\Omega}_2} a^H_{11} \frac{\partial u^0}{\partial \widetilde{x}_1^2} \frac{\partial u^0}{\partial \widetilde{x}_1^2}\, d\widetilde{x}_1^2 \quad (4)$$

$$= \int_{\widetilde{\Omega}_1} f^H v^0\, d\widetilde{x}^1 + \int_{\widetilde{\Omega}_2} f^H v^0\, d\widetilde{x}_1^2$$

for all $v \in V^0$. The final approximation in the physical region Ω is

$$u \approx B^1(u^0 + \varepsilon \sum_{i=1}^{2} (y_i + \overline{w}_i^1) \frac{\partial u^0}{\partial \widetilde{x}_i^1}) \text{ in } \Omega_1, \quad (5)$$

where $\overline{w}_i^1 = w_i^1 - \frac{1}{|Y_1|} \int_{Y_1} w_i^1\, dy$ and B^1 is the approximate inverse two-scale transform defined for any Y_1-periodic function $v(\widetilde{x}^1, y)$ by $(B^1 v)(x) = v(\widetilde{x}^1, \frac{x - \varepsilon/2}{\varepsilon})$. A similar approximation is built in Ω_2.



(a) (b)

Fig. 5: Results for u^0 by FEM simulations for (a) the nominal model (b) the two-Scale model

Now we compare the results obtained with the nominal model and the simplified model, see Fig. 5. To obtain

sufficient accuracy of calculations with the software COMSOL we used meshes with 459,840 elements to model the complete problem, of 7664 elements for both microscopic problems and with 305 elements for the macroscopic problem. The relative quadratic error

$$\frac{\left(\int_\Omega |u-u^0|^2 dx\right)^{1/2}}{\left(\int_\Omega |u^0|^2 dx\right)^{1/2}}$$

between u, the model nominal solution, and u^0, this of the homogenized model, is 6.85%. The computing time on a laptop for the three problems are respectively 3.734s, 0.046s and 0.015s, giving a ratio of computation time of the nominal model and simplified model of 0.029.

5. Conclusion

We presented the basic principles of a new software dedicated to the generation of multiscale models for applications to MEMS arrays and their illustration on an example. The symbolic transformation package and the type-checker have been successfully applied together to a couple of models. In particular, the asymptotic model (4) from the running example has been produced from model (2) with the assistance of these tools. We underline that there are a number of interests with our approach. Clearly, the final model is "correct-by-construction", human errors are avoided, and the model derivation effort is dramatically reduced. In the point of view of a MEMS designer, this software will generate simplified analytical and numerical models corresponding to a design and to a set of requested simplifications. Since the computation will be fast enough, he will test a variety of simplifications and retain the one satisfying the best tradeoff between simulation time and precision.

References

1. M. Lenczner. Homogénéisation d'un circuit électrique. *C. R. Acad. Sci. Paris Sér. II b*, 324(9):537–542, 1997.
2. J. Casado-Díaz. Two-scale convergence for nonlinear Dirichlet problems in perforated domains. *Proc. Roy. Soc. Edinburgh Sect. A*, 130(2):249–276, 2000.
3. D. Cioranescu, A. Damlamian, and G. Griso. Periodic unfolding and homogenization. *C. R. Math. Acad. Sci. Paris*, 335(1):99–104, 2002.
4. 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.
5. M. Marin and F. Piroi. Rule-based programming with mathematica. In *In Sixth Mathematica Symposium (IMS 2004)*, pages 1–6, 2004.
6. W. Belkhir, A. Giorgetti, and M. Lenczner. Rewriting and symbolic transformations for multiscale methods. December 2010. Submitted, http://arxiv.org/abs/1101.3218v1.
7. H. Cirstea, C. Kirchner, and L. Liquori. The rho cube. In F. Honsell and M. Miculan, editors, *FoSSaCS*, volume 2030 of *Lecture Notes in Computer Science*, pages 168–183. Springer, 2001.
8. H. Geuvers. Introduction to type theory. In A. Bove, L. Barbosa, A. Pardo, and J. Pinto, editors, *Language Engineering and Rigorous Software Development*, volume 5520 of *Lecture Notes in Computer Science*, pages 1–56. Springer, 2009.
9. A. Bensoussan, J-L. Lions, and G. Papanicolaou. *Asymptotic analysis for periodic structures*, volume 5 of *Studies in Mathematics and its Applications*. North-Holland Publishing Co., Amsterdam, 1978.
10. P. G. Ciarlet. *Mathematical elasticity. Vol. I*, volume 20 of *Studies in Mathematics and its Applications*. North-Holland Publishing Co., Amsterdam, 1988.
11. M. Lenczner and G. Senouci-Bereksi. Homogenization of electrical networks including voltage-to-voltage amplifiers. *Math. Models Methods Appl. Sci.*, 9(6):899–932, 1999.
12. J. Casado-Díaz. Two-scale convergence for nonlinear Dirichlet problems in perforated domains. *Proc. Roy. Soc. Edinburgh Sect. A*, 130(2):249–276, 2000.
13. 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.
14. D. Cioranescu, A. Damlamian, and G. Griso. The periodic unfolding method in homogenization. *SIAM Journal on Mathematical Analysis*, 40(4):1585–1620, 2008.
15. S. Fèvre and D. Wang. Combining algebraic computing and term-rewriting for geometry theorem proving. In *Proceedings of the International Conference on Artificial Intelligence and Symbolic Computation*, pages 145–156, London, UK, 1998. Springer-Verlag.
16. R. Bündgen. Combining computer algebra and rule based reasoning. In *Integrating Symbolic Mathematical Computation and Arti Intelligence. Proceedings of AISMC-2, volume 958 of LNCS*, pages 209–223. Springer, 1995.
17. P. Baldan, C. Bertolissi, H. Cirstea, and C. Kirchner. A rewriting calculus for cyclic higher-order term graphs. *Mathematical Structures in Computer Science*, 2006.
18. H. Cirstea, G. Faure, and C. Kirchner. A rho-calculus of explicit constraint application. *The journal of Higher-Order and Symbolic Computation*, 2005.
19. E. Balland, P. Brauner, R. Kopetz, PE. Moreau, and A. Reilles. Tom: Piggybacking Rewriting on Java. In *RTA*, pages 36–47, 2007.
20. W. M. Farmer. A basic extended simple type theory. SQRL report 14, McMaster University, 2003.