A Tool for Aided Multi-Scale Model Derivation and its Application to the Simulation of a Micro Mirror Array

Walid Belkhir[1,2], Nicolas Ratier[1], Duy Duc Nguyen[1], Nguyen Nhat Binh Trinh[1], Michel Lenczner[1], and Frédéric Zamkotsian[3]

[1]FEMTO-ST institute, univ. Bourgogne Franche-Comté, CNRS, ENSMM Time and frequency dept. 26 Rue de l'Épitaphe, 25030 Besançon cedex, France.
[2]INRIA Nancy - Grand Est, Pesto project, 54600 Villers-lès-Nancy, France.
[3]LAM-CNRS, Marseille, France.
Email : walid.belkhir@inria.fr, nicolas.ratier@femto-st.fr, duyduc.nguyen@femto-st.fr, nhat-binh.trinh@femto-st.fr, michel.lenczner@femto-st.fr, frederic.zamkotsian@lam.fr

Abstract

Modeling the electric field in a matrix of micro-mirrors is presented as the first application of the MEMSALab software package. The latter is dedicated to semi-automated derivation of multiscale models by asymptotic methods and will complement simulation software as finite element software. It is designed according to a principle of reusability which is called the extension-combination method developed with techniques derived from the theory of rewriting.
Keywords: Rewriting Strategies, Periodic Homogenization, Micro-Mirror Array, Extension-Combination Method.

1. Introduction

Many microsystems and nanosystems exhibiting one or more multi-scale characteristics are impossible to fully simulate with standard tools. In this article, we present the recent developments of the MEMSALab software package whose aim is to make available the use of multiscale models to a large body of researchers and engineers.

A multi-scale phenomenon is generally characterized by one or more small (or large) parameters. These parameters constitute the starting point for the construction of reduced models by asymptotic or perturbation methods applied to Partial Differential Equations (PDE). Small or large parameters can originate from time phenomena (e.g. ratio of frequencies between low and high frequency components) or spatial phenomena (periodic structures with large number of cells, large coefficient variations, small ratio between lengths, etc). The outcome is a system of PDEs requiring much less computation time than the nominal model. These methods are grounded on a solid mathematical basis and offer a good compromise between the precision of the models and the computation time. Their big drawback is that the models must be built on a case-by-case basis and that their construction always requires a fairly large mathematical expertise in order to solve specific difficulties. Until now, this has been preventing their integration in general simulation software.

This integration is the goal of the MEMSALab project (for MEMS Array Lab) which recent advances are presented in this paper. They are illustrated by an example of asymptotic modeling of the electrostatic field in the Matrix of Micro-Mirrors (MMA) MIRA, well documented in [4] and [5]. Unlike a case-by-case method of construction, the adopted method is based on the concept of reusability. It combines mathematical principles of the two-scale approximation method established in [6] and concepts of the rewriting theory issued from theoretical computing. The first developed applications are in the field of micro and nanosystems.

The flowchart represented on Fig. 1 is a global view of the operations of the MEMSALab software package. A multi-scale model derivation starts with an input Partial Differential Equation (PDE) extracted from a PDE solver (FEM in the figure) and being expressed in the User Language. Together with the features to be taken into account for the asymptotic analysis, they are transmitted to the core that generates the asymptotic model. Then, the latter is sent to the PDE solver for simulation.

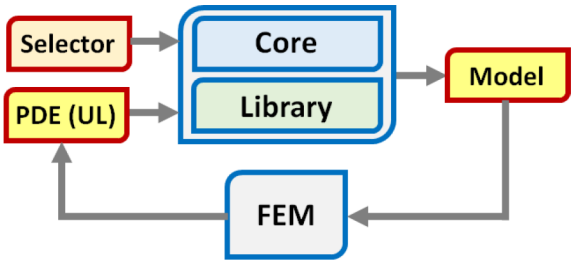

Fig. 1: flowchart of MEMSALab starting from an input partial differential equation and producing an output sent in a simulation tool for PDEs as a finite element method software package (FEM). The input can also be generated by a user thanks to the User Language (UL). The *Selector* is a graphic interface that selects the extensions to be combined for building the multi-scale model in the *Core*. The model construction is made by combining extensions of the library.

This flow of operations is established for what we call the reference case i.e. the periodic homogenization of a

978-1-5090-4344-6/17/$31.00 ©2017 IEEE

2017 18th International Conference on Thermal, Mechanical and Multi-Physics Simulation and Experiments in Microelectronics and Microsystems

second order PDE posed in a one-dimensional domain and with periodic coefficients. Then, it is complexified, we say extended, in several manners to take into account new features yielding new schemes. The input PDEs that come from a script or a software package has additional features. Accordingly, the reference proof is extended in different ways to cover the new features. Applying the *extended proofs* to the enriched PDEs yields new multi-scale models. Finally, a new scheme for an input PDE covering a group of new features is built by *combination*. It is worthwhile to underline that by construction, an extension is much smaller than the proof that it generates, and is minimal in the sense that it includes only what is specific to the features involved in the new PDE.

Before to state the contributions of this paper, let us review the recent progresses. The language *Symbtrans* of rewriting strategies for coding the proofs has been developed as a Maple® package [2]. An early formulation of the idea of model derivation by *Extension-Combination* was introduced and was partially implemented in Maple® as reported in [10]. For efficiency reasons the program has been rewritten in Ocaml, a functional language well suited for implementing rewriting techniques. It is associated to a Matlab® interface for launching compilation, applications of proofs or extensions and for displaying. Then, a *User Language* has been built to define input PDEs as scripts. An important contribution was a better formalization of the principle of model derivation by the *Extension-Combination* method. These progresses have been reported in [8] and illustrated with an homogenized model of the heat transfer in a micro-mirror array. However, this was just an illustration but still not an implementation.

Since then, two classes of strategies of extensions have been studied with the goal to find a class of extension strategies that is closed through combination and in the same time that is transposable in the user language. Two studies that also include formulae of combinations of extensions have been published in the report [3]. Here, we present the second one that has been successfully turned into a specific language of extensions and added to the user language.

The second novelty of this paper is the presentation of a library of extensions written in the strategy language and its use for the derivation by the *Extension-Combination* method of the multi-scale model of the electric field in a micro-mirror array presented in [7].

Finally, we also report on original displaying and debugging tools that are of great help for program writing.

The next section goes into the concept of extension-combination in slightly more details. Then, in the rest of the paper, each aspect of this method and of its application to the MMA model derivation is detailed from the points of view of input model, asymptotic methods, computing language and user language.

2. The Extension-Combination Method

For the construction of an asymptotic model taking into account several characteristics of the PDE and/or of the proof, the extension-combination method begins with the construction of extension operators associated with each characteristics, cf Ext_1 and Ext_2 on Figure 2. When applied to the reference pair $(\mathrm{PDE}_{\mathrm{Ref}}, \mathrm{Proof}_{\mathrm{Ref}})$ they lead to new pairs $(\mathrm{PDE}_1,\mathrm{Proof}_1)$ and $(\mathrm{PDE}_2,\mathrm{Proof}_2)$. It is assumed that the extensions are *correct* in the sense that for each pair $(\mathrm{Proof}_n, \mathrm{PDE}_n)$ the application of Proof_n to PDE_n provides an asymptotic model Model_n taking correctly into account the n^{th} characteristics.

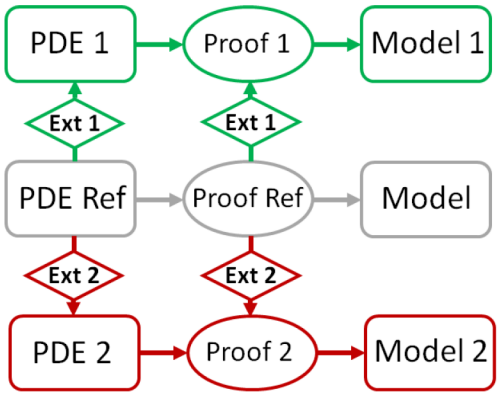

Fig. 2: Schematic view of the scheme of asymptotic model generation for the pair $(\mathrm{PDE}_{\mathrm{Ref}}, \mathrm{Proof}_{\mathrm{Ref}})$ and their extensions $(\mathrm{PDE}_1,\mathrm{Proof}_1)$ and $(\mathrm{PDE}_2,\mathrm{Proof}_2)$ by Ext_1 and Ext_2.

Fig. 3: Schematic view of the combination of Ext_1, Ext_2 and the complementary extension Ext'_{12} built to generate a correct extension Ext_{12}.

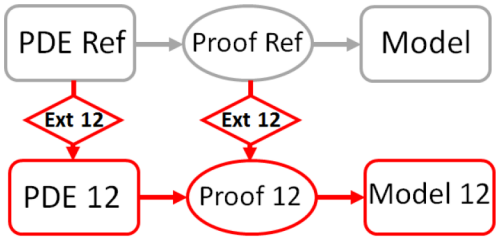

Fig. 4: Schematic view of the pair $(\mathrm{Proof}_{12}, \mathrm{PDE}_{12})$ built by the combination Ext_{12} and the asymptotic model Model_{12}.

By construction, the combination of several extensions, e.g. $\mathrm{Ext}_1 + \mathrm{Ext}_2$, is another extension covering all the characteristics of the involved extensions but which is not necessarily *correct* in the above sense. To reach correctness, it is generally, but not always, necessary to combine $\mathrm{Ext}_1 + \mathrm{Ext}_2$ with a complementary extension, e.g. Ext'_{12} in Figure 3. Figure 4 shows the final step of the method, i.e. the pair $(\mathrm{Proof}_{12}, \mathrm{PDE}_{12})$, built by application of the extension Ext_{12}, generating the exptected model Model_{12}.

3. The Reference PDE

The choice of the pair $(\mathrm{PDE}_{\mathrm{Ref}}, \mathrm{Proof}_{\mathrm{Ref}})$ was done in [10] and the mathematical justifications in [6]. The ref-

erence equation is the second order elliptic posed in the an interval $\Omega = (a, b)$,

$$-\frac{d}{dx}(a\frac{d}{dx}u) = f \text{ in } \Omega$$

with mixed Dirichlet and Neuman boundary conditions

$$u = h \text{ on } \Gamma^D \text{ and } a\frac{d}{dx}u = g \text{ on } \Gamma^N.$$

Here, Γ^D and Γ^N are two complementary parts of the boundary of Ω.

3.1 Representation in the User Language

The User Language allows descriptions of equations, boundary conditions, initial conditions and geometry characteristics. As the script of Reference PDE in Figure 5 shows, the sections Constant, Index, Region, Variable, Function, Operator, Expression and PDE are to define constants, indices, domains, mathematical variables, functions, operators operating on functions, mathematical expressions and finally a PDE. The scripts are written in Unicode UTF8 format to allow the use of conventional mathematical symbols such as ∂, $\int$, $\sum$, etc and thus to express the equations in a natural and readable form.

3.2 Representation in the Computation Language

The user language programs are converted into a computational language which is implemented in OCaml. In particular, the grammar of PDEs, of proofs and extensions are implemented as OCaml types allowing to reduce the programming errors. Besides, there is a unique abstract OCaml type to which all the previous three types are converted. All the computational processing, such as the symbolic transformations, the display and the debugging operate on the abstract OCaml type in the *Core* represented in Figure 1.

3.3 Displaying Tools for Degugging and Reporting

Displaying PDEs, but also of proofs and extensions, can be done in the Matlab command window with the possibility of hiding all the terms of one or more of the types as Index, Region etc. For instance, the internal fields of all functions can be hidden. Another interface in an html window offers more flexibility to hide or show subterms in an expression. By selection with a mouse, it is possible to hide or show all the terms of one or several types and to hide internal fields or context of a term. Both tools are used for program debugging. For report editing, expressions are converted to Latex files and then to PDF files.

4. Extension to the Electrostatic Equation in the MMA

This section is devoted to the extension of PDE_{Ref} to equation PDE_{Elec} that governs the electrostatic field of the MMA. We begin with a brief description of the structure of the MMA, the boundary value problem that governs the electrical potential and the extension $\text{PDE}_{\text{Ref}} \mapsto \text{PDE}_{\text{Elec}}$. Then, we provide details on the principle of extensions and combinations that are used in the kernel.

```
Index
  i_BC : "i_BC" ["D","N"] ["Given"]
  i_BC_D : "i_BC" ["D"] ["Given"]
  i_BC_N : "i_BC" ["D"] ["Given"]
Region
  gamma      : "gamma" [i_BC] [] [] __ __
  gamma_D    : "gamma" [i_BC_D] [] [] __ __
  gamma_N    : "gamma" [i_BC_N] [] [] __ __
Variable
  xg     : "xg" [] gamma
  xg_D   : "xg" [] gamma_D
  xg_N   : "xg" [] gamma_N
Function
  normal : "normal" [] [xg] [] ["Given"]
Region
  omega : "omega" [] [] [] gamma normal
Variable
  x : "x" [] omega
Function
  a : "a" [] [x]     [] ["Given"]
  f : "f" [] [x]     [] ["Given"]
  h : "h" [] [xg_D]  [] ["Given"]
  g : "g" [] [xg_N]  [] ["Given"]
  u : "u" [] [x]     [] ["Unknown"]
Expression
  equation : - ∂ (a • ∂ u / ∂ x) / ∂ x = f
PDE
  pde : "pde" [] equation [u, h, xg_D
                , a • ∂ u / ∂ x, g, xg_N] [] [0]
  pde : "pde_set" [] 0 [] [] [pde]
```

Fig. 5: Script defining the reference boundary value problem in the user language. The functions a, f and u are depending on the variable x which is defined on the domain omega. The latter has for boundary gamma and for outward unit normal n. The functions h and g are functions of xg_D or xg_N two variables defined on two parts of the boundary.

4.1 Electrostatic Equation in the MMA as an Extension

The structure and operation mode of MIRA, made with 64x32 cells of size 100 x 200 μm, is detailed in [4] and [5]. Figure 6 shows the components of its elementary cell that is divided into the mirror and the electrode blocks. The mirror block is composed of the mirror itself, two stopper beams with two landing beams on their tips and a suspending beam. The electrode block is composed of an electrode, two landing pads and two pillars. Each cell is indivudually addressable and is tilted due to the electrostatic force on the mirror surface. At rest, when no voltage is applied, the micro-mirror is held in a flat position by the suspended beams. When a voltage is applied between the micro-mirror and the electrode, an electrostatic force is generated, resulting in the attraction of the micro-mirror toward the fixed electrode. This yields tilting and therefore provides a restoring force. For voltages below the pull-in voltage, the mirror angle is set to a few degrees. At the pull-in voltage, the electrostatic force increases dramatically and the mirror snaps down toward the electrode until it is stopped when the mirror touches the stopper beam on one side and the landing pads in the other side. Therefore,

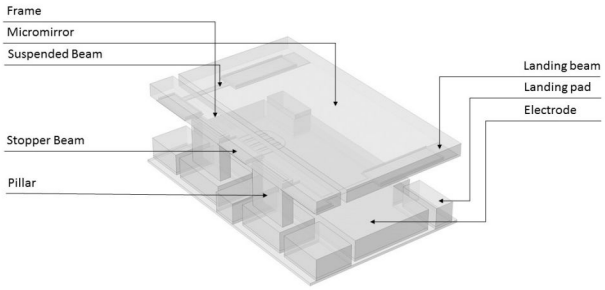

Fig. 6: The components of a cell of MIRA.

after pull-in the mirror is fixed at a precise tilt angle. During the phase of voltage decrease, the mirror remains in the same position until it detaches from the stopper beam and the tilt angle increases. Finally, when the spring force of the suspended beams overcome the electrostatic force, the landing beams detach from the landing pads and the mirror returns to its rest position.

The model of this paper is for a one-dimensional array as represented on Figure 7 but with a large number of cells.

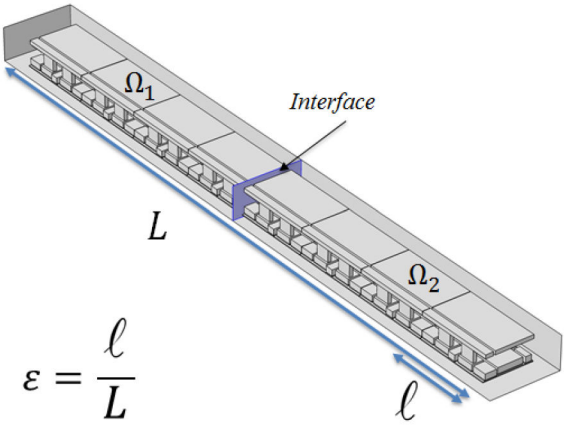

Fig. 7: The one-dimensional MMA modeled in [7] with different voltage sources in the two subdomains Ω_1 and Ω_2.

From the mathematical point of view, the MMA is located in a three-dimensional domain Ω and the field of electric potential u is governed by t he PDE,

$$-\sum_{i,j} \frac{\partial}{\partial x_i}\left(a_{ij}\frac{\partial}{\partial x_j}u\right) = 0 \text{ in } \Omega,$$

where a is the permittivity matrix which is generally reduced to a single coefficient. The electrical potential satisfies Dirichlet conditions on the electrodes expressed as $u = h$ on Γ^D and an homogeneous Neuman condition $\sum_{i,j} a_{ij}\frac{\partial}{\partial x_j}u\, n_i = 0$ on the complementary part Γ^N of the boundary. We also assume that $h = \pm V_k$ in the top and bottom electrodes of the two subdomains Ω_k of $\Omega = \Omega_1 \cup \Omega_2$.

To pass from the Reference PDE to the electrostatic PDE one uses an extension which transforms the domain Ω into a three-dimensional domain. This is done by adding indices to the space variables x and xg, replacing the scalar coefficient a by a matrix, introducing indices to the derivatives and summations behind the internal equation and the Neuman condition. Figure 8 shows the fragment of the extension that specifies the vector character of the variable $x \in \Omega$, the boundary variable $xg \in \Gamma$, and that changes the functions a, u and f accordingly.

```
Constant
  space_dim : "3"
Index
  i_dim  : "i_dim"   [space_dim] ["Given"]
Region
  gamma'ndim    : "gamma" [i_BC] [] [] __ __
Variable
  xg'ndim   : "xg" [i_dim] gamma'ndim
Function
  normal'ndim : "normal" [i_dim] [xg'ndim] [] ["Given"]
Region
  omega'ndim : "omega" [] [] [] gamma'ndim normal'ndim
Variable
  x'ndim : "x" [i_dim]   omega'ndim
  x_var  : "x" []        region_
Function
  f_var          : name_ [] [x]       []  type_
  f_var'ndim     : name_ [] [x'ndim] []  type_
Rule
  f_ext : "func_dim_ext" f_var ⇉ f_var'ndim
Extension
  pde'ndim : OuterMost ("ndim", Id (f_var) ; f_ext )
```

Fig. 8: Script of a part of the PDE extension that transforms a function $f(x)$ into a function $f(\mathbf{x})$ which variable is a vector $\mathbf{x} = (x_1, x_2, x_3)$. All internal fields of $\mathbf{x}$ are changed accordingly.

4.2 Principles of Extension-Combination in the Core

In addition to the specification of PDEs, the user language allows specifications of extensions. Roughly speaking, an extension is a symbolic transformation that operates on the tree-like structure of a PDE or a proof defined below. The design of the extensions is inspired by the formalism of μ-calculus [1]. When applied to a PDE, an extension yields a new extended PDE. An extension is composed of two parts: a navigation part that searches and locates a series of patterns, and an insertion part that inserts a new expression to a part of the PDE. The insertion part of an extension consists of adding one or many expressions to an expression. This is done by the extension rule `Exp` $\rightrightarrows$ `Exp'`. For instance the extension rule `f(x)` $\rightrightarrows$ `f(x)`, from Figure 8, inserts an index `i` to `x`, to the normal of `x` and to `xg`. The navigation part of an extension can be viewed as a particular rewriting strategy which is built up of the following constructors. The extension, which is a restricted form of the composition, `Id(u);S` tries to match the expression `u` with the input expression, if it does then the extension `S` is applied to the input expression, otherwise the extension `Id(u);S` fails. The left-choice extension `S1 Or S2` applies the extension `S1` to an expression, if this application fails, then `S2` is applied to the expression. The conditional extension `If (S1 and ... and Sn) Then S` applies the extension `S` to an input expression after checking that the application of each `Si` to the expression does not fail. If one of the

Si fails, then the full extension fails. The one-step extension Inside(S) applies the extension S to all the immediate subexpressions of the input expression. The iterative extension is composed of the introduction of a label L to an extension S by Label L Of S, together with a jump operator GoTo L that appears in S. This extension iterates the application of the extension S starting from the input expression. The one-step constructor Inside together with the iterative constructor allows one to encode many navigation strategies such as OuterMost.

For example, the extension $\texttt{OuterMost}(\texttt{L1}, \texttt{Id}(\texttt{f}(\texttt{x})); \texttt{f}(\texttt{x}) \rightrightarrows \texttt{f}(\texttt{x}))$, from figure 8, consists in a search of all subexpressions of the form f(x) starting from the top of the PDE, then each subexpression is transformed into f(x). In fact, a more refined extension would be $\texttt{OuterMost}\big(\texttt{L1}, \texttt{Id}(\texttt{f}(\texttt{x})); \texttt{OuterMost}(\texttt{L2}, \texttt{Id}(\texttt{x}); \texttt{x} \rightrightarrows \texttt{x})\big)$. This extension searches for all subexpressions of the form f(x) starting from the top of the PDE, then in each subexpression it makes a second search of all expressions of the form x, the latter being transformed into x.

It is worth noting that the extension OuterMost(L, S) is not a rudimentary one and can be defined in the User Language in terms of the rudimentary extensions mentioned above by OuterMost(L, S) := Label L Of (S Or Inside(GoTo L)). This means that either the extension S is applied successfully, or it fails and in this case we go into each subexpression of the input expression and we apply the full extension again.

Two such extensions can be combined to generate a third new extension. The combination of two extensions amounts to merge their navigation parts and their insertion part. More precisely, the combination of two extensions S and S′ is the extension S″ Or S Or S′, where S″ is the extension that results from the merging of S with S′. For instance, the merging of the extension Id(u); S with Id(u′); S′ yields the extension Id(u″); S″ where u″ is the expression that results from unification of u and u′, and S″ is the merging of S with S′. The merging of Inside(S) and Inside(S′) is the extension If (S and S′) Then Inside(S″), where S″ is the merging of S with S′. The merging of the extension If (S1 and ... and Sn) Then S with the extension If (S1′ and ... and Sm′) Then S′ is the extension If (S1 and ... and Sn and S1′ and ... and Sm′) Then S″, where S″ is the merging of S with S′. The merging of the insertions of two expressions is simply their successive insertions. Finally, we devised a criterion for the correctness of the operation of combination of extensions, and we have proved that the combination is indeed correct.

5. Reference Model Derivation

We sketch the model derivation and the grammar of strategies for its specification.

5.1 The Two-Scale Model and its Derivation

The derivation of the homogenized model presented in [6] is based on the assumption of periodicity of the coefficient a and on the hypothesis of a large number N of periods present in the interval Ω. It consists of a succession of property applications grouped into seven blocks. The properties can be general as the Green formula, or specific to the technique of two-scale approximation. For a domain Ω of dimension 1 the Green formula corresponds to the integration by part formula, but it is preferable to write it directly in the general form

$$\int_\Omega \frac{d}{dx}u\, v\, dx = -\int_\Omega u\, \frac{d}{dx}v\, dx + \int_\Gamma \mathrm{tr}(u)\ \mathrm{tr}(v)\ n\, ds$$

where tr(.) represents the trace operator of a function on the boundary. A green_rule implementation is in figure 9, it is accompanied with the OuterMost strategy discussed in Section 5.2.

```
Expression
  yg         : y_.Region.Normal.Variable
  n_y_name   : y_.Region.Normal.Name
Function
  v          : "v"        [] [x]   [] type_
  normal_y   : n_y_name   [] [yg] [] ["Given"]
Operator
 tr_u    : "Trace" [] [u_] [y_] [yg] []
 tr_v    : "Trace" [] [v_] [y_] [yg] []
Rule
  green_rule :  "green_rule" ∫  ∂ u_ / ∂ y_ • v_ đ x_
        →  - ∫ u_ • ∂ v_ / ∂ y_ đ x_
        + ∫ tr_u • tr_v • normal_y đ x_
        if y_.Name = x_.Name
Strategy
  green_rule_st : green_rule ↧
```

Fig. 9: Script of the Green Rule and of its application with the strategy "OuterMost".

The homogenized model consists of two boundary problems, the macroscopic problem posed on the macroscopic structure equal to Ω in this case, and the microscopic problem posed on a dilated reference cell which is in this case the interval $\Omega^1 = (0, 1)$. The approximation for small $\varepsilon = 1/N$ of the solution u by the solution of the two-scale model is of the form $u(x) \approx u^0(x) + \varepsilon\widetilde{u}^1(x, x/\varepsilon) + \varepsilon O(\varepsilon)$. The mean behavior is governed by u^0 solution of the macroscopic problem, while $u^1(x, x^1)$ is a small rapidly oscillating perturbation defined on $\Omega \times \Omega^1$ and is Ω^1-periodic in the variable x^1. It is constructed from u^0 and the solution of the microscopic problem. Given its rapid variations, the role of $\widetilde{u}^1$ is of the same order of magnitude as that of u^0 in the approximation of the derivative $\frac{du}{dx}(x) \approx \frac{du^0}{dx}(x) + \frac{d\widetilde{u}^1}{dx^1}(x, x/\varepsilon) + O(\varepsilon)$.

The derivation of this model is based on the two-scale transformation T that maps any function $u(x)$ defined in the physical domain Ω into a function $(Tu)(x, x^1)$ defined in the two-scale domain $\Omega \times \Omega^1$. According to [6], the construction of the homogenized model consists of seven lemmas that are derived from the asymptotic expansion hypothesis of the two-scale transformation

$$Tu \approx u^0 + \varepsilon u^1 + \varepsilon O(\varepsilon),$$

(i) u^0 is independent of x^1, (ii) $\dfrac{du}{dx} \approx \dfrac{du^1}{dx^1} + O(\varepsilon)$, (iii) $\widetilde{u}^1 = u^1 - x^1\dfrac{du^0}{dx^\sharp}$ is Ω^1-periodic, (iv) u^0 satisfies the

boundary conditions at the extremities of Ω, (v) formulation of the coupled equations of u^0 and $\widetilde{u}^1$, (vi) the relation between u^0 and $\widetilde{u}^1$: $\dfrac{d\widetilde{u}^1}{dx^1} = \dfrac{du^0}{dx}\dfrac{d\theta}{dx^1}$ and the equation satisfied by θ, and (vii) the homogenized equation satisfied by u^0.

5.2 A Grammar of Strategies

The User Language that can specify PDEs and their extensions is also able to specify proofs. A proof is formalized in the User Language as a rewriting strategy that is a symbolic transformation that describes the way basic strategies are applied, see the survey [9]. The User Language is built up out of basic strategies, traversal strategies, an iterative strategy and structuring strategies. A basic strategies is a rewriting rule `Exp1` $\rightarrow$ `Exp2` corresponding to a mathematical rule. When applied to an input expression, it transforms it into `Exp2` provided `Exp1` matches with the input expression. Otherwise, this rewriting rule fails. The left-choice strategy `S1|S2` applies the strategy `S1` to the input expression, and if this fails, then it applies the strategy `S2`. The sequential composition strategy `S1;S2` applies `S1` followed by `S2`. The traversal strategy $\mathtt{S}\overline{\downarrow}$, standing for the outermost strategy, applies the strategy `S` to the input expression, if this fails then $\mathtt{S}\overline{\downarrow}$ is applied again to each sub-expression of the input expression. The strategy $\mathtt{S}\circlearrowleft$ iterates the application of `S` until it fails or a fixed-point is reached. Finally, there are strategy constructors allowing one to structure proofs such as `Step(name,S)` and `Lemma(name,S)`.

Proofs can be involved in several operations: a proof can be applied to a PDE, displayed, and extended by means of extensions similar to those described for PDEs. For debugging operation, two proofs can be compared with a dedicated tool that locates the differences modulo the associativity of the left-choice "|" and the sequential composition ";" constructors. Moreover, during a proof application an option allows to display the number of times that a rewriting rule has been applied successfully and unsuccessfully.

6. Extension to the Two-Scale Model of MMA

All aspects of the extension of the reference proof to the asymptotic model of electrostatics in the MMA are shortly described: the features to be covered by the proof, the changes in the steps of the proof, their implementation in the language and in the core, the principle of the extension-combination method in the core and performance indicators for extensions. The implementation of the model in a finite element software has been done manually, see Figure 12, but is not discussed here as it is the object of [7].

6.1 Features in the Proof of the MMA Model

In addition to the transition from a one-dimensional domain to a three-dimensional domain which constitutes the first feature (F1), the construction of the asymptotic model of MMA presents three others features that are taken into account in four extensions of the proof. (F2) The presence of imposed values of the electrical potential in each cell prevents the macroscopic variations of the asymptotic solution u^0. Therefore the main variations of u^0 are inside the cells only, the approximation of the derivative is of the form $\varepsilon\nabla u \approx \nabla_{x^1} u^0 + O(\varepsilon)$ and u^0 is Ω^1-periodic. (F3) Due to the periodicity, u^0 cannot satisfy the nominal boundary conditions at the ends of the network this causes boundary layer effects i.e. additional contributions u_0^{BL} and u_L^{BL} at the two ends of the network. (F4) Due to the two constant sources of voltages V_1 and V_2 in Ω_1 and Ω_2 and the periodicity of the solution u^0 in each of these parts there exists also a boundary layer solution u_{int}^{BL} at the interface which allows to recover the continuity of the model solution.

6.2 Extensions for the Proof

The electric potential solution of the asymptotic electrostatic model has the form

$$u^{BL,0} \approx T^{-1}u^0 + T_{BL,0}^{-1}u_0^{BL} + T_{BL,L}^{-1}u_L^{BL} + T_{BL,\text{int}}^{-1}u_{\text{int}}^{BL}.$$

It is noteworthy that the construction of the equations satisfied by u^{BL} is performed in a mathematical framework harmonized with the two-scale approximation method but replacing the two-scale transformation by an operator for each boundary layer which transforms Ω into a semi-infinite domain.

The four features F1-F4 are taken into account independently by four extensions which are applied to the construction of the reference model. Taking into account F1 is done in a similar way to what has been done to transform the reference equation. To take F2 into account, the construction of the model of u^0 seems very different from that of the reference model. However, a precise inspection shows that it can be carried out by first transforming the weak formulation used in the reference construction into a very weak formulation (ie by transfering all the derivatives on the test function) and then by applying Step (v) which leads directly to the equation of u^0 in very weak form which remains to interpret in terms of strong form. For F3, it is enough to introduce an indexation of the subdomains. Finally, the boundary layer equations of the case F4 are obtained in a manner similar to that of the case F2 but using the boundary layer operators instead of the two-scale transformation. A fragment of the script of the extension that covers F2 is shown in Figure 10. It consists of adding the steps `step0` (transforming the weak formulation to the very weak formulation), `step1_1` (a complement in the definition of specific test functions for the very weak formulation), and step5 (transforming the very weak formulation into the strong formulation and interpretation) without making any change inside Lemma 5 of the reference proof. The extension associated with F3 is of the same nature. These two extensions are carried out in indifferent order. The extension associated with F1 brings modifications to several levels of the reference Proof as well as in the extensions associated to F2 and F3. It is therefore applied after F2 and F3.

6.3 Principle of Extensions-Combinations in the Core

The principle of extensions of proofs is the same as the one of extension of PDEs presented in subsection 4.2. That is, there is only one extension language in which the user writes his extensions and can apply them to both PDEs and proofs. it follows that the combination of extensions in this case are the same as the one explained before in subsection 4.2.

```
Step
  step1_var  :  "step1" set_test_function_
  step2_var  :  "step2" opB_to_opTS_
  step3_var  :  "step3" opTS_to_opT_
  step4_var  :  "step4" pass_to_the_limit_
Step
  step0  : "step0"   weak_to_very_weak_form
  step1_1 : "step1_1" set_test_function
  step5  : "step5"   very_weak_to_strong_form
Lemma
  block5_var : "lemma5"
               step1_var
               ; step2_var
               ; step3_var
               ; step4_var
  block5 : "lemma5"
               step0
               ; step1_var
               ; step1_1
               ; step2_var
               ; step3_var
               ; step4_var
               ; step5
SORule
  block5_ext : "block5_ext" block5_var ⇉ block5
Extension
  proof'largrad_horiz : block5_ext
```

Fig. 10: Fragment of the script of the extension for the characteristics F2.

6.4 Implementation of the Extensions for the MMA Model

The derivation of the asymptotic MMA model was implemented with the extension-combination method using the three features F1, F2 and F3[1]. Four indicators of complexity and performance were used for the reference and extended proofs: the number of rewriting rules used without counting the repetitions, same but counting the repetitions, the size of a proof counted in kilo-nodes (knodes) without counting the size of the rules therein, and the size of the set of rules without counting the repetitions. In order to evaluate the extensions, we use the relative difference of these indicators between the extended and reference proofs. Finally, a global indicator of complexity, called full proof size, is the sum of the size of a proof regardless of the size of the rules and the size of the rules without repetition.

7. Conclusion

We presented our recent advances in the development of MEMSALab software dedicated to the computer-aided generation of multi-scale models. The main idea behind MEMSALab is the incremental generation of families of multiscale models by integrating their features instead of the case-by-case model construction. We described the core of MEMSALab software, namely the extension-combination method together with its theoretical foundations, and language for the specification of PDEs, proofs and their extensions. This design method was partially but successfully applied to a micro-mirror array, leading to the simulation of this complex multi-scale multi-physics component as a whole. Although the extension-combination idea was announced and partially implemented in previous works in the early stages of the development of MEMSALab, it reaches its maturity in this work. The novelty of this paper is to devise a class of rewriting strategies to formulate extensions together with an internal operation that corresponds to the intended combination operation. We implemented a part of the reference proof in the User Language and its extension to several features.

The extension-combination method is not specific to the multi-scale applications, it is a general method that can be used in the design of complex systems as far as the reusability and the incremental development are concerned. Besides, the User Language, grounded on the transformation by rewriting strategies, is not narrowly tied to our application domain, it could be of general use in the transformation of mathematical formulas, data-structures, programs etc

We aim to generate more complex models with MEMSALab software and to integrate them in a library for industrial use. We plan to automate the transfer to the finite element method software. We plan to automate the elimination of non-accessible code in the extensions obtained

[1]The implementation of F4 is not yet complete at the moment of the paper submission but does not present additional difficulties.

	# Rules without repetitions	# Rules with Repetitions	Proof Size (# kNodes)	Size of the Rules without Repetitions (# kNodes)	Full Proof Size (# kNodes)
Raw values					
ref	57	295	0.77	4.6	5.4
ndim (F1)	62	331	0.79	5.1	5.9
largrad (F2)	84	521	1.3	8.0	9.3
bndlay (F3)	72	485	1.2	7.1	8.3
Ratios (%)					
ndim (F1)	8%	11%	3%	10%	8%
largrad (F2)	30%	40%	40%	40%	40%
bndlay (F3)	21%	39%	38%	35%	35%

Fig. 11: Measures of the complexity and performances of the extension-combination method for MMA modeling.

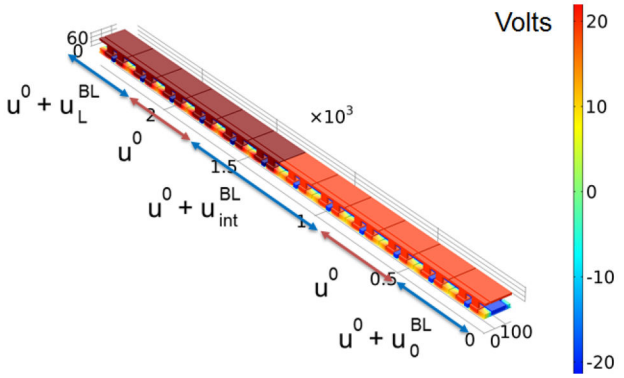

Fig. 12: Simulation of the asymptotic MMA model of the electrostatic field in a 12 cells array with the superimposition of the three solutions u^0, u_0^{BL}, u_L^{BL} and u_{int}^{BL}. It is worth noting that the simulation time of this model is independent of the size of the array.

by combination. The would drastically reduce of the size of the extensions. A more challenging problem is the automatic generation of parts of the extensions.

References

1. André Arnold and Damian Niwinski. *Rudiments of μ-calculus*. Studies in logic and the foundations of mathematics. London, Amsterdam, 2001.
2. W. Belkhir, A. Giorgetti, and M. Lenczner. A symbolic transformation language and its application to a multiscale method. *Journal of Symbolic Computation*, 65:49–78, 2014.
3. W. Belkhir, N. Ratier, D. D. Nguyen, and M. Lenczner. Closed combination of context-embedding iterative strategies. *Report hal-01277395*, 2016.
4. MD Canonica, F Zamkotsian, P Lanzoni, W Noell, and N De Rooij. The two-dimensional array of 2048 tilting micromirrors for astronomical spectroscopy. *Journal of Micromechanics and Microengineering*, 23(5):055009, 2013.
5. Michael David Canonica. *Large Micromirror Array Based on a Scalable Technology for Astronomical Instrumentation*. PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2012.
6. 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):776–805, 2007.
7. D. D. Nguyen, N. N. B. Trinh, M. Lenczner, F. Zamkotsian, and S. Cogan. A multi-scale model of the electric field surrounding a one-dimensional micro-mirror array and robust design optimization of a cell. In *submitted to EuroSimE*, 2017.
8. D.D Nguyen, W. Belkhir, N. Ratier, B. Yang, M. Lenczner, F. Zamkotsian, and H. Cirstea. A multi-scale model of a micro-mirror array and an automatic model derivation tool. In *EurosimE*, 2015.
9. Eelco Visser. A survey of strategies in rule-based program transformation systems. *Journal of Symbolic Computation*, 40(1):831 – 873, 2005.
10. B. Yang, W. Belkhir, and M. Lenczner. Computer-aided derivation of multiscale models: A rewriting framework. *International Journal for Multiscale Computational Engineering*, 12(2), 2014.