Towards an automatic tool for multi-scale model derivation illustrated with a micro-mirror array

Walid Belkhir, Nicolas Ratier, Duy Duc Nguyen, Bin Yang, Michel Lenczner, Frédéric Zamkotsian, Horatiu Cirstea

Towards an automatic tool for multi-scale model derivation illustrated with a micro-mirror array(⋆)(†)

Walid Belkhir[1,2], Nicolas Ratier[3], Duy Duc Nguyen[1], Bin Yang[1], Michel Lenczner [1], Frédéric Zamkotsian[4] and Horatiu Cirstea[5]

[1]FEMTO-ST, Time and Frequency Department, University of Franche-Comté, 25000, Besançon, France

[2]INRIA Nancy - Grand Est, CASSIS project, 54600 Villers-lès-Nancy,France.

[3]FEMTO-ST, Time and Frequency Department, Technical University of Belfort-Monbéliard, 90010 Belfort, France.

[4]LAM-CNRS, 38 rue Frédéric Joliot-Curie 13388, Marseille, France.

[5] University of Lorraine - LORIA, Campus scientifique - BP 239 - 54506 Vandœuvre-lès-Nancy, France.

Email: {duyduc.nguyen, nicolas.ratier, bin.yang}@femto-st.fr, walid.belkhir@inria.fr, michel.lenczner@utbm.fr, frederic.zamkotsian@lam.fr, Horatiu.Cirstea@loria.fr

***Abstract*—This paper reports recent advances in the development of a symbolic asymptotic modeling software package, called MEMSALab, which will be used for automatic generation of asymptotic models for arrays of micro and nanosystems. More precisely, a model is a partial differential equation and an asymptotic method approximate it by another partial differential equation which can be numerically simulated in a reasonable time. The challenge consists in taking into account a wide range of different physical features and geometries e.g. thin structures, periodic structures, multiple nested scales etc. The main purpose of this software is to construct models incrementally so that model features can be included step by step. This idea, conceptualized under the name "*by-extension-combination*", is presented for the first time. A user friendly language recently introduced is also shortly discussed. We illustrate the mathematical operations that need to be implemented in `MEMSALab` by an example of an asymptotic model for the stationary heat equation in a Micro-Mirror Array developed for astrophysics.**

Keywords. *Symbolic computation, computer-aided derivation of asymptotic models, rewriting strategies, homogenization, micro-mirror array*

I. Introduction

Many systems encountered in micro or nano-technologies are governed by differential or partial differential equations (PDEs) that are too complex to be directly simulated with general software. In a number of cases, the complexity of the simulation is due to a combination of many factors as several space scales or time scales, large coefficient heterogeneity or large aspect ratios. Many methods have been developed to overcome these difficulties, and in particular the asymptotic methods, also called perturbation techniques, constitute an active field of research in all fields of physics and mathematics for more than a century. Their application is based on a case-by-case approach so they are implemented only in specialized software. In all cases, we observe an important reduction in the simulation computation time with reasonable precision. We adopt an alternate approach by developing a software package called `MEMSALab` (for MEMS Array Lab) whose aim is to incrementally derive asymptotic models for input equations by taking into account their own features e.g. the scalar valued or vector valued solution, different estimates on the solutions and sources, thin structures, periodic structures, multiple nested scales etc.

Our approach of the software development is two-fold. On one hand we develop computer science concepts and tools allowing the software implementation and on the other hand we derive and implement asymptotic models to anticipate the introduction of related modeling concepts in the software library. This paper is written in this spirit, it reports our latest advances in the development of the kernel of `MEMSALab` and reports on an asymptotic model of a Micro-Mirror Array (MMA) introduced in [4] and dedicated to applications in astrophysics. The technique of model derivation relies on an asymptotic method taking into account the small ratio between the sizes of a cell and of the whole array [1]. It is not detailed since it is relatively long and technical, however we present some key facts giving an idea of the features playing a role in the derivation.

In [2] we presented a transformation language implemented as a Maple package. It relies on the paradigm of rule-based programming and rewriting strategies as well as their combination with standard Maple code. We used this language to encode "by hand" the homogenized model of the stationary heat equation with periodic coefficients. Then, in [3] we introduced a theoretical framework for computer-aided derivation of multi-scale models. It relies on a combination of an asymptotic method with term rewriting techniques. In the framework [3] a multi-scale model derivation is characterized by the features taken into account in the asymptotic analysis. Its formulation consists in a derivation of a *reference proof* associated to a *reference model*, and in a set of *extensions* to be applied to this proof until it takes into account the wanted features. The reference model covers a very simple case i.e. the periodic homogenization model of a scalar second-order elliptic equation posed in a one-dimensional domain. The re-

(⋆) A detailed report can be found at https://hal.inria.fr/hal-01223141

(†) This work was supported by LABEX ACTION ANR-11-LABX-0001-01.

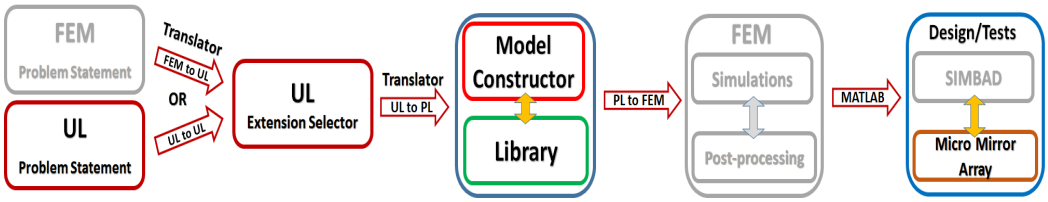

Fig. 1: Flow of a MEMSALab Application.

lated reference proof is a series of derivations that turn a partial differential equation into another one. An extension transforms the tree structure of the proof as long as the corresponding feature is taken into account, and many extensions are composed to generate a new extension. The composition of several existing elementary extensions instead of the development of new extension transformations has the advantage of reducing the development effort by avoiding doing complex changes manually. This method has been applied 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. However, it is limited to extension not operating on the same positions of the tree.

This limitation is due to an unsufficient formalization of the concept of extension. The present paper fills the gap, so the "*by-extension-combination*" method specifies what is meant by extension, also called generalization, and how it is implemented in terms of added context and/or parametrization. The clear statement allows defining rigorously the combination of extensions. Key implementation aspects are discussed. The symbolic transformation language, also called "*Processing Language*", previously written in the Maple package is now in Ocaml to gain in development flexibility, to reduce the programming errors and to take advantage of a free environment. A "*User Language*" is now available for the specification of the proofs and the extensions.

The MMA introduced in [4], see figure 9, that illustrates the application of asymptotic methods is used for instance as a field selector for multi-object spectroscopy since it allows individual selection of objects by preventing overlapping of spectra and remove spoiling sources and background emission. The full modeling of the MMA should cover mechanical, electrical and thermal effects, however this paper focuses on the heat diffusion only. Although we have not yet implemented the MMA model in the MEMSALab software, this sophisticated application illustrates complex features, see subsection V-B, that we want to take into account by means of the by-extension-combination method.

Finally, we mention that our purpose is not to fully formalize the multi-scale model proofs as with a proof assistant e.g. Coq [5], but to devise a methodology for an incremental construction of complex model proofs, as well as a tool that comes with such methodology. It is worth mentioning that the concept of proof reuse by means of abstraction/generalization and modification of formal proofs was investigated in many works e.g. [6]. Although the notion of unification is at the heart of our formalism as well as the works on the reuse of formal proofs by generalization, these works do not consider the combination of proofs. Finally, this approach is new at least in the community of multi-scale methods where asymptotic models are not derived by computer-aided combinations.

II. MEMSALab

We describe the main operations of MEMSALab and the processing and specification languages.

A. Operation principle of MEMSALab

The main components are the Library of elementary extensions and the core, or Model Constructor. The latter consists in an engine of rewriting strategies and an engine of extensions and combinations. It operates on expressions written in the Processing Language (PL). A user specifies its problem using the User Language (UL), a language close to the usual mathematical language. Translators convert the PL into the UL and vice-versa. The expected software operation flow is represented in Figure 1. It starts from an Input Model specified either in a specification file written in UL or using a FEM software. In the second case, the specification is extracted and translated into UL through a FEM to UL translator. The elementary extensions are pre-defined in the EC-Library. They are combined and applied to the reference proof to generate a complete generalization applied to an input model yielding a final asymptotic model output in UL format and subsequently sent to FEM. Finally, calibration and optimizations are done thanks to SIMBAD, a dedicated home made software package, through its connection to FEM.

B. The processing language

We describe the grammar of expressions in which the problem is processed leaving out other structures as proofs, lemmas, propositions, etc.

$$
\begin{aligned}
\mathcal{E} ::= & \ \texttt{Plus}(\mathcal{E},\mathcal{E}) \mid \texttt{Mult}(\mathcal{E},\mathcal{E}) \mid \\
& \ \texttt{Minus}(\mathcal{E}) \mid \texttt{Inverse}(\mathcal{E}) \mid \texttt{Power}(\mathcal{E},\mathcal{E}) \mid \mathcal{F} \\
\mathcal{F} ::= & \ \texttt{Fun}(f,[\mathcal{I};\ldots;\mathcal{I}],[\mathcal{V};\ldots;\mathcal{V}],[(\mathcal{R},\mathcal{E});\ldots;(\mathcal{R},\mathcal{E})],K) \mid \\
& \ \texttt{Oper}(A,[\mathcal{I};\ldots;\mathcal{I}],[\mathcal{E};\ldots;\mathcal{E}],[\mathcal{V};\ldots;\mathcal{V}],[\mathcal{V};\ldots;\mathcal{V}] \\
& \qquad [d;\ldots;d]) \mid \\
& \ \mathcal{V} \mid \texttt{MathCst}(d) \mid \texttt{Nil} \\
\mathcal{V} ::= & \ \texttt{MathVar}(x,[\mathcal{I};\ldots;\mathcal{I}],\mathcal{R}) \\
\mathcal{R} ::= & \ \texttt{Reg}(\Omega,[\mathcal{I};\ldots;\mathcal{I}],[d,\ldots,d],[\mathcal{R};\ldots;\mathcal{R}],\mathcal{R},\mathcal{E}) \mid \texttt{Nil}, \\
\mathcal{I} ::= & \ \texttt{Ind}(i,[d,\ldots,d])
\end{aligned}
$$

It describes mathematical expressions built up by the arithmetic operations "+" (`Plus`), "·" (`Prod`), etc as well as the mathematical function constructor `Fun` and the operator constructor `Oper`. The latter allows one to build expressions for mathematical operators such as the integration operator $\int$, the derivative operator ∂, the summation operator $\sum$, the multi-scale operators T, B, etc. Besides, a mathematical expression can contain mathematical variables (`MathVar`), regions (`Reg`) and discrete variables (`Ind`).

We shall sometimes write and depict lists in the prefix notation using the constructor `list` and `nil` (empty list). For instance, if e_1 and e_2 are two expressions, we shall write $\texttt{list}(e_1, \texttt{list}(e_2, \texttt{nil}))$ instead of $[e_1; e_2]$. The symbol `Nil` in the grammar above represents an "empty expression".

C. The user language

The user language is used for specification and also for examples in papers. Instead of writing full expressions in a single term as in the processing language, the User Language allows the definition of expressions from shortcuts. For instance, we shall simply write x_i instead of $\texttt{MathVar}(x, [\texttt{Ind}(i, [n])], \texttt{Reg}(\Omega, \ldots))$ provided that the domain $\texttt{Reg}(\Omega, \ldots)$ and the dimension n are already defined, except in papers were we often ommits unwanted details. We shall also write $\partial_x v(x)$ instead of $\texttt{Oper}(\texttt{Partial}, \texttt{Fun}(v, \ldots), \ldots)$. Besides, these expressions can contain *rewriting variables*. A rewriting variable, denoted by $x\_, y\_$, etc, is a particular term that can match any expression. For instance, the shortcut expression $\Omega\_$, which abbreviates the expression $\texttt{Reg}(\Omega\_, \ldots)$, stands for any domain. However, it is worth mentioning that the notion of mathematical variable (e.g. x) should not be confused with the one of rewriting variable (e.g. $x\_$). We implemented a parser of the user language and the transformer into the processing language.

Example 1: We give an example of the Green rule

$$\int_{\Omega\_} u\_ \frac{dv\_}{dx\_}\,\mathrm{d}x\_ \to \int_{\Gamma\_} tr\,(u\_)\,tr\,(v\_)\,n\_\,\mathrm{d}s\,(x\_) - \int_{\Omega\_} v\_ \frac{du\_}{dx\_}\,\mathrm{d}x\_ \tag{1}$$

written in the user language as,

```
Model "Green formula for one-dimensional domain
and scalar functions" :
Operator
 trace_a : "Trace" [trace_Ind_] [a_] [x_]
           [x_.Region.Boundary] [trace_Pa_]
 trace_b : "Trace" [trace_Ind_] [b_] [x_]
            [x_.Region.Boundary] [trace_Pa_]
Rule
 "green_rule" : ∫∂a_/∂x_ • b_ dx_
        → ∫trace_a•trace_b•∂x_.Region.Boundary.Normal
          dx_.Region.Boundary-∫a_•∂b_/∂x_ dx_
```

This specification contains two declaration blocks for the declaration of operators and rewriting rules, where "$\cdot \to \cdot$" denotes a rewriting rule, • stands for the product, and `.Region` and `.Boundary` and `.Normal` are predefined functions allowing the access to the fields "Region", "Boundary" and "Normal", respectively, of a given expression. Notice also that the derivative operator ∂ and $\int$ are already predefined, and the user does not need to declare them.

III. The by-extension-combination method

This section is the main contribution of the paper. After stating the concept of the *by-extension-combination* method, it is expressed in terms of added context at positions. Then, it is restated in a form based on usual rewriting strategies. This is also the form chosen for implementation yielding the preliminary results.

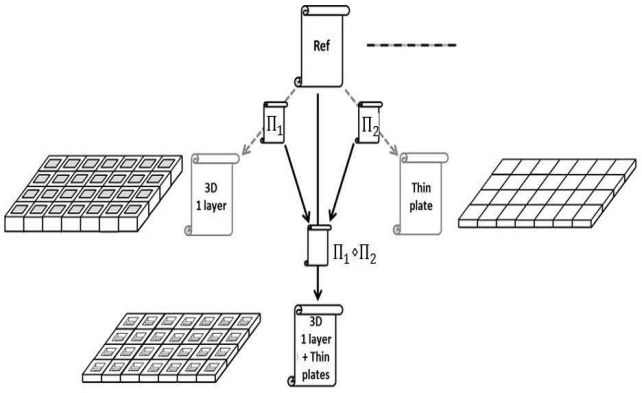

Fig. 2: By-extension-combination principle illustrated on a reference proof (top). Left: a one-layer periodic problem. Right: a thin layer with homogeneous coefficients. The combination of these two extensions yields a thin layer with periodic coefficients (bottom).

A. General principle

The idea of the extension can be viewed as a generalization of a proof. It is based on two concepts: *mathematical equivalence* and *parametrisation*. Consider the expression $\partial_x v$ that we want to generalize to $\partial_{x_i} v$ where $i \in \{1, \ldots, n\}$. We proceed in two steps. First a mathematical equivalence consists in introduction a discrete variable i, ranging from 1 to 1, to the expression $\partial_x v$, yielding the expression $\partial_{x_i} v$, where $i \in \{1, \ldots, 1\}$. Notice that this transformation does not change the mathematical meaning. Secondly, the step of parametrisation consists in replacing the upper bound 1 by a variable n, yielding the expression $\partial_{x_i} v$, where $i \in \{1, \ldots, n\}$. We propose next an implementation of the notion of generalization by the by-extension-combination method. This method relies on three key principles. Firstly, we introduce a reference model together with its derivation. This derivation is called the reference proof, it is depicted on the top of Fig. 2. It is based on the derivation approach of [1] and was implemented and presented in details in [3]. Although the reference model covers a very simple case, its proof is expressed in a sufficiently general way. A number of basic algebraic *properties* are formulated as *rewriting rules*, they are considered as the building blocks of the proofs. The full derivation of the model is formulated as a sequence of applications of these rules. The proof of some

properties is also performed by a sequence of applications of mathematical rules when the others are admitted e.g. the Green rule.

Then, an *elementary extension* is obtained by an application of an elementary transformation, called also an *extension operator*, to the reference proof. In Fig. 2 the extension operators are Π_1 and Π_2. They respectively cover the extension to the 3-D setting and the thinness setting. We notice that, in practice, when a single feature is taken into account, only a small change occurs in a relatively long proof. In other words, while considering an elementary extension, most of the existing rules could be reused by operating a small change on them, and, on the other hand, only a small number of new rules has to be manually introduced.

Finally, we make possible the combination of two extension operators to produce a new extension operator that takes into account the features covered by each initial extension operator. In the example of Figure 2, the combination of the extension operators Π_1 and Π_2 is the extension operator $\Pi_1 \diamond \Pi_2$. By iterating this process, many extension operators can be combined together giving rise to complex extensions that cover many features.

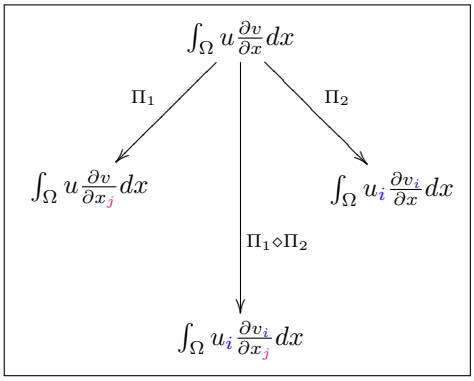

Fig. 3: An example of the by-extension-combination method applied to the mathematical expression $\int_\Omega u\frac{\partial v}{\partial x}dx$ that corresponds to the left hand side of Green formula (1), where Π_1 stands for the extension operator of the *multi-dimension setting*, Π_2 stands for the extension operator to the *vector-valued setting*, and $\Pi_1 \diamond \Pi_2$ stands for the combination of Π_1 and Π_2.

Figure 3 shows how the extension operators and their combination operate on the mathematical expression $\int_\Omega u\frac{\partial v}{\partial x}dx$, which is the left hand side of Green formula (1).

B. Implementation through added contexts at positions

An extension operator consist in the operation that adds *contexts* or replace terms by rewriting variables for parametrization at given positions of a term. For the sake of shortness, we do not take term replacement into account in the rest of the paper. The context $\tau = \texttt{list}(\Box, j)$ depicted in Figure 4 captures the idea that the extension would add a discrete variable to an expression. The application of $\Pi_{(p,\tau)}$ to the term $t = \partial_x v(x)$ at the position of p of the variable x (the parameter of the derivative operator ∂) yields the term $\Pi_{(p,\tau)}(t) = \partial_{x_j} v(x)$. Similarly, Figure 5 illustrates the extension operator $\Pi_{(q,\tau')}$ and its application to the term $t = \partial_x v(x)$ at the position of the function v which yields the term $\Pi_{(q,\tau')}(t) = \partial_x v_i(x)$.

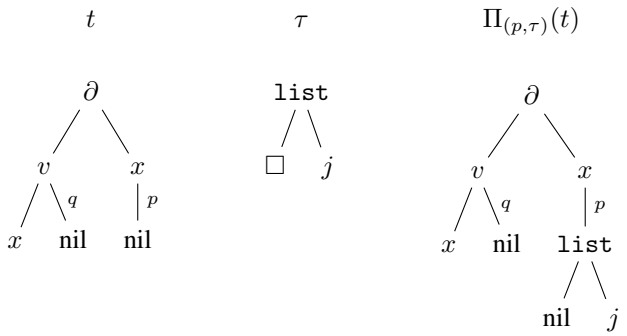

Fig. 4: Application of the extension operator $\Pi_{(p,\tau)}$ (with the extension constructor τ) to the term $t = \partial_x v(x)$ at the position p, yielding the term $\partial_{x_j} v(x)$.

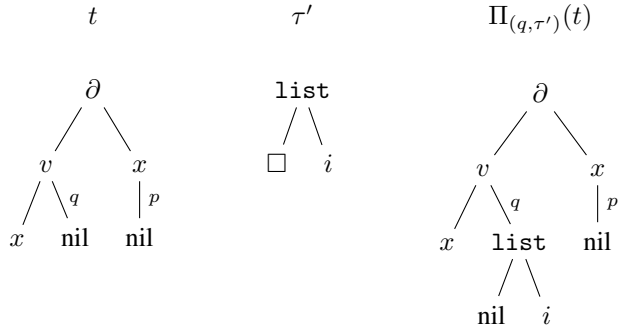

Fig. 5: Application of the extension operator $\Pi_{(q,\tau')}$ (with the extension constructor τ') to the term $t = \partial_x v(x)$ at the position q, yielding the term $\partial_x v_i(x)$.

When an extension operator $\Pi_{(p,\tau)}$, where p is a position, is applied to a term t at the position p, the context τ is inserted at the position p of t, and the subterm of t at the position p is inserted at $\Box$. The general schema of the application of an extension operator is depicted in Figure 6.

Figure 7 shows the combination of the two extension operators $\Pi_{(p,\tau)}$ and $\Pi_{(q,\tau')}$. In what follows $\mathcal{P}os$ stands for the set of positions, $t_{|p}$ stands for the subterm of t at the position p, and $t[t']_p$ stands for the replacement of the subterm of t at the position p with the term t'. If τ is a context, we shall denote by $\tau[t]$ the term obtained from τ by replacing the $\Box$ by t. In particular, if τ and τ' are contexts, we call $\tau[\tau']$ the composition of τ and τ'. Formally, a parameter of an extension operator is of the form $\mathcal{P} = [(p_1, \tau_1), \ldots, (p_n, \tau_n)]$, where p_i are positions and τ_i are contexts and each position p_i occurs at most once in $\mathcal{P}$. The extension operator $\Pi_{(p,\tau)}$ is defined as the function $u \mapsto u[\tau[u_{|p}]]_p$. And the extension operator $\Pi_{[(p_1,\tau_1),\ldots,(p_n,\tau_n)]}$ is defined as the composition $\Pi_{(p_1,\tau_1)}; \ldots; \Pi_{(p_n,\tau_n)}$. Let $\mathcal{P} = [(p_1, \tau_1), \ldots, (p_n, \tau_n)]$ and

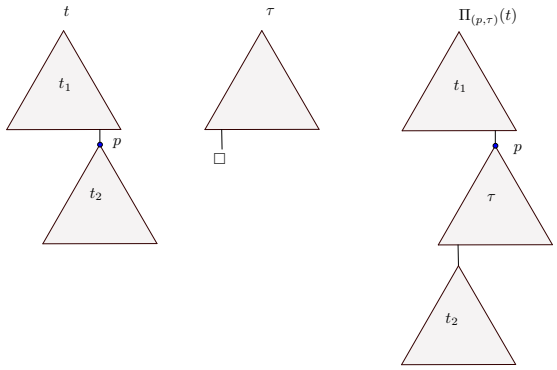

Fig. 6: Schematic diagram of the application of an extension operator $\Pi_{(p,\tau)}$ (with a context τ) to a term t at the position p.

$\mathcal{P}' = [(p'_1, \tau'_1), \ldots, (p'_m, \tau'_m)]$. We define the combination $\Pi_{\mathcal{P}} \diamond \Pi_{\mathcal{P}'}$ by $\Pi_{\mathcal{P}\diamond\mathcal{P}'}$ with $\mathcal{P} \diamond \mathcal{P}' = [(p''_1, \tau''_1), \ldots, (p''_r, \tau''_r)]$, where, for all i, either *(i)* $p''_i = p_j$, for some j, and in this case $\tau''_i = \tau_j$, or *(ii)* $p''_i = p'_j$ and $\tau''_i = \tau'_j$, or *(iii)* $p''_i = p_j = p'_k$, for some j, k, and in this case $\tau''_i = \tau'_k[\tau_j]$.

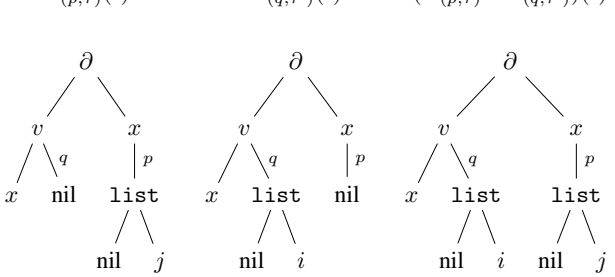

Fig. 7: The extension operator $\Pi_{(q,\tau)} \diamond \Pi_{(q,\tau')}$ which is the combination of the two extension operators $\Pi_{(p,\tau)}$ and $\Pi_{(q,\tau')}$, and its application to the term t.

C. Implementation with strategies.

Instead of considering only positions, we can enrich the definition of the extension operators to incorporate both positions and nested searching *patterns* which are expressions with rewriting variables. A pattern allows one to locate the subexpression on which the extension constructor is applied. Thus the semantics of such extension operators is given as a rewriting strategy. The justification of such enrichment is double. On the one hand, it is much more practical and flexible to handle patterns and positions instead of positions. And on the other hand, we shall define the combination of this enriched formalism, and we prove that the two definitions are equivalent. The grammar of the (enriched) parameters of extension operators follows.

$$\text{(I)} \quad \begin{cases} \mathscr{P} & ::= (\theta, \tau) \mid (\theta, \mathscr{P}) \mid [\mathscr{P}, \ldots, \mathscr{P}] \mid \texttt{IM}(\mathscr{P}) \\ \theta & ::= p \mid u \end{cases}$$

where p is a position, τ is a context, u is a pattern, and IM is an unary constructor. A parameter is well-founded if for all its subparameters of the form $[(u_1, \mathcal{P}_1), \ldots, (u_n, \mathcal{P}_n)]$, where each u_i is a pattern, we have that u_j can not unified with an subterm of u_k, for all $j \neq k$. In all what follows we assume that the parameters are well-founded. Let $\Theta(\mathcal{P})$ to be the set of positions at the root of $\mathcal{P}$ if $\mathcal{P}$ is viewed as a tree.

For a parameter $\mathscr{P}$, the extension operator $\Pi_{\mathscr{P}}$ is defined as a *rewriting strategy*. A rewriting strategy is a combination of the basic strategy constructors ";" which stands for the composition of two strategies, "$\oplus$" which stands for the left choice of two or many strategies, and "InnerMost" which stands for the traversal strategy that applies a given strategy to a term starting from the leaves of this term and goes up to its root and stops at the first successful application. The definition of the extension operator $\Pi_{\mathcal{P}}$ by induction on $\mathcal{P}$ follows.

$$\Pi_{(\theta,\tau)} \stackrel{def}{=} \begin{cases} t \mapsto t[\tau[t_{|\theta}]]_\theta & \text{if } \theta \in \mathcal{P}os \\ u \to \tau[u] & \text{if } \theta = u \end{cases}$$

$$\Pi_{(\theta,\mathscr{P})} \stackrel{def}{=} \begin{cases} t \mapsto t[t']_\theta & \text{if } \theta \in \mathcal{P}os \\ \quad \text{where } t' = \Pi_{\mathscr{P}}\left(t_{|\theta}\right) & \\ (\theta \to \theta\ ); \Pi_{\mathscr{P}} & \text{if } \theta = u \end{cases}$$

$$\Pi_{[\mathscr{P}_1,\ldots,\mathscr{P}_n]} \stackrel{def}{=} \begin{cases} \Pi_{\mathscr{P}_1}; \ldots; \Pi_{\mathscr{P}_n} & \text{if } \Theta(\mathscr{P}_n) \neq \emptyset \\ \Pi_{\mathscr{P}_1} \oplus \ldots \oplus \Pi_{\mathscr{P}_n} & \text{otherwise} \end{cases}$$

$$\Pi_{\texttt{IM}(\mathscr{P})} \stackrel{def}{=} \texttt{InnerMost}(\Pi_{\mathscr{P}})$$

In Figure 7 we informally showed how to combine extension operators whose parameters are of the form (p, τ), where p is a position and τ is a context. We briefly illustrate next how to combine extension operators in general. We take the assumption that we can combine two extension operators only if they have the same structure. Roughly speaking, it means that we can combine a position with a position, a pattern with a pattern, and an "IM" with an "IM". These assumption can be indeed relaxed and we do not discuss this here.

The combination $(u, \tau) \diamond (u', \tau')$ is defined by $(\sigma(u), \tau[\tau'])$, where σ is the most general unifier of u and u'. While the combination $\texttt{IM}(u, \tau) \diamond \texttt{IM}(u', \tau')$ amounts to check whether u can be unified with a subterm of u', to construct a new pattern u'' out of this unification and to compose the related contexts τ and τ'. The combination of two lists of parameters $[(u_1, \mathcal{P}_1), \ldots, (u_n, \mathcal{P}_n)] \Diamond [(u'_1, \mathcal{P}'_1), \ldots, (u'_m, \mathcal{P}'_m)]$, where u_i, u'_j are patterns, is the parameter $[(u''_1, \mathcal{P}''_1), \ldots, (u''_r, \mathcal{P}''_r)]$, where for all $k \in \{1, \ldots, r\}$, either *(i)* $u''_k = u_i$, for some i, and this case $\mathcal{P}''_k = \mathcal{P}_i$, or *(ii)* $u''_k = u'_i$, for some i, and this case $\mathcal{P}''_k = \mathcal{P}'_i$, or *(iii)* $u''_k = \sigma(u_i) = \sigma(u'_j)$, for some i, j with σ being the most general unifier of u_i and u'_j, and this case $\mathcal{P}''_k = \mathcal{P}_i \Diamond \mathcal{P}'_j$. Finally, the combination $\texttt{IM}([(u_1, \mathcal{P}_1), \ldots, (u_n, \mathcal{P}_n)]) \Diamond \texttt{IM}([(u'_1, \mathcal{P}'_1), \ldots, (u'_m, \mathcal{P}'_m)])$ is similar to the previous case except that in the case *(iii)* we try to unify u_i with a subterm of u'_j and vise versa. An immediate consequence of the definition of the combination of extension

operators is the following Proposition.

Proposition 2: The class of compatible extension operators are closed under the combination operation $\diamond$, that is, if Π_1 and Π_2 are extension operators, then their combination $\Pi_1 \diamond \Pi_2$ is an extension operator too.

The correction of the combination of the extension operators whose parameters are given by the grammar (I) is ensured by relating it to the combination of pattern-free extension operators as stated in the following Proposition and not discussed here.

Proposition 3: There exists a mapping ψ that, for every term t and extension parameter $\mathcal{P}$ generated by the grammar (I), constructs a pattern-free parameter $\psi(t, \mathcal{P})$ such that $\Pi_{\mathcal{P}}(t) = \Pi_{\psi(t,\mathcal{P})}(t)$.

The complete framework of the by-extension-combination method requires adding *anti-patterns* to the definition of the parameters of extension operators besides patterns and positions. We do not discuss them here, they are detailed in the extended version of this paper available at https://hal.inria.fr/hal-01223141.

D. Example

As an elementary example, we discuss the extension of the Green rule in Eq. (1) to both the multi-dimensional setting (3) and to the multi-valued setting (4) and the combination of these two extensions that yields (5). Precisely, the formula (1) is trivially extended for any domain $\Omega \subset \mathbb{R}^m$ with $m \in \mathbb{N}^*$,

$$\int_\Omega u \frac{\partial v}{\partial x_j} \, \mathrm{d}\mathbf{x} = \int_\Gamma \mathrm{tr}\,(u)\,\mathrm{tr}\,(v)\, n_j \, \mathrm{d}s - \int_\Omega v \frac{\partial u}{\partial x_j} \, \mathrm{d}\mathbf{x} \qquad (3)$$

for any $j \in \{1, ..., m\}$. Another trivial extension is for any vector valued functions $\mathbf{u} = (u_1, ..., u_n)$, $\mathbf{v} = (v_1, ..., v_n)$ defined on Ω with $n \in \mathbb{N}$,

$$\int_\Omega u_i \frac{\partial v_i}{\partial x} \, \mathrm{d}\mathbf{x} = \int_\Gamma \mathrm{tr}\,(u_i)\,\mathrm{tr}\,(v_i)\, n_j \, \mathrm{d}s - \int_\Omega v_i \frac{\partial u_i}{\partial x} \, \mathrm{d}x \qquad (4)$$

for any $i \in \{1, ..., n\}$. The combination of these two extensions states for $\Omega \subset \mathbb{R}^m$ with $m \in \mathbb{N}^*$ and $\mathbf{u} = (u_1, ..., u_n)$, $\mathbf{v} = (v_1, ..., v_n)$ defined on Ω with $n \in \mathbb{N}$

$$\int_\Omega u_i \frac{\partial v_i}{\partial x_j} \, \mathrm{d}\mathbf{x} = \int_\Gamma \mathrm{tr}\,(u_i)\,\mathrm{tr}\,(v_i)\, n_j \, \mathrm{d}s - \int_\Omega v_i \frac{\partial u_i}{\partial x_j} \, \mathrm{d}\mathbf{x} \qquad (5)$$

for any $i \in \{1, ..., n\}$ and $j \in \{1, ..., m\}$. The definition of the extensions in Ocaml follows those expressed in the above mathematical formulae. To extend Equation (1) into Equation (3), we change the dimension of the domain from 1 into m and then we add index j to Equation (1)'s variable. Similarly, we add index i to the functions of Equation (3) to get Equation (4). The two operators of extensions can be expressed in the user language,

```
Extension "multi-dimensional domains" :
  Index
     j  : "j" [1,JD_]
  Variable
     x_mdim : "x_mdim" [j] []
   Rule 2
     "Ext_mdim": ∂u_/∂x ⇉ ∂u_/∂{x_mdim}
```

and

```
Extension "vector valued functions" :
   Index
     i : "i" [1,ID_]
   Function
      u_vect: "u_vect_" [i] [] []
      v_vect: "v_vect_" [i] [] []
   Rule 2
     "Ext_vect_valued" : u·∂v/∂x_ ⇉ u_vect·∂v_vect/∂x_
```

The table of Figure 8 illustrates the size of the reference proof related to a PDE and three lemmas of [3], and the size of two extension specifications (`ndim` and `vvf`), as well as the ratio between the size of the extension specifications and the reference specifications. We notice that only a small number of extension operators is needed.

	#lines of Ref	#lines of Ext. `ndim`	#lines of Ext. `vvf`	`ndim` %	`vvf` %
PDE (24) of [3]	36	13	16	0.36	0.44
Lemma 20 of [3]	89	14	11	0.16	0.12
Lemma 21 of [3]	140	19	16	0.13	0.11
Lemma 22 of [3]	150	19	20	0.12	0.13

Fig. 8: The ratio between the size of the extension operators and the size of the reference PDE and lemmas of [3].

IV. Application to the Micro-Mirror Array

Figure 11 shows the components of its elementary cell which is divided into two parts: the mirror part and the electrode part. The mirror part is composed of the mirror, two stopper beams with two landing beams on their tips and a suspending beam. The electrode part is composed of an electrode, two landing pads and two pillars.

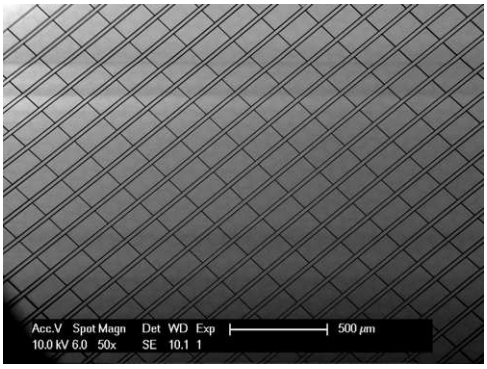

Fig. 9: Real Micro-Mirror Array

A complete modeling should take into account the electric voltages between the micro-mirrors and the electrodes; the thermo-elasticity problem in beams; the heat diffusion in all parts, and the linear frictionless contact problem between stopper beam and landing pads. Here, we only consider the heat transfer in the whole array. Regarding the heat sources, one is coming from the environment such as black bodies at the given temperature or mechanical parts around the component gives a significant contribution to warm up the MMA, while the other coming from the stars and galaxies is very small and probably negligible. To dissipate this heat, the MMA is attached to a heat sink.

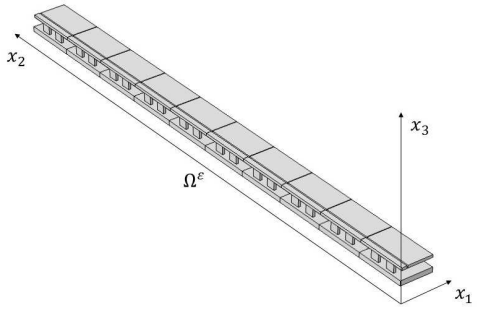

Fig. 10: A MMA with 10x1 cells.

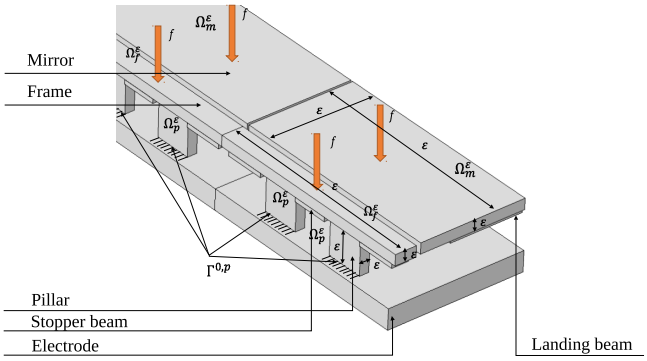

Fig. 11: Scales and heat loads on a cell

V. Homogenized model of the heat equation for the MMA

Starting from the mathematical statement of the heat equation in the MMA, we describe the assumptions taken into account in the asymptotic model derivation, the two-scale transform which is the key mathematical tool, the a priori estimates of the solution, the asymptotic model itself, and the simulation results. These are the operations expected to be done by `MEMSALab` linked to a FEM software package.

A. Mathematical model

We consider a $N \times 1$ array of MMs as shown in Figure 10 which cell represented in Figure 11 is comprised of a micro-mirror including two stopping beams, a part of the frame, two pillars and an electrode made of silicon. After rescaling the array size to a unit length, we denote by ε the order of magnitude of the cell dimensions. This parameter decreases when the number N of cells of the array increases, and we determine, in a sense explained hereafter, an approximation of the temperature field for small values of ε. The region Ω^ε occupied by the device which is split into Ω^ε_m, Ω^ε_f, $\Omega^\varepsilon_p \subset \mathbb{R}^3$ the subregions occupied by the mirrors including the stopping beams, the frame and the pillars respectively. The body heat source r is present in the frame and the mirror, and the thermal conductivity may be anisotropic with matrices $\mathbf{a}^\varepsilon$, $\mathbf{a}^{m,\varepsilon}$, $\mathbf{a}^{f,\varepsilon}$ and $\mathbf{a}^{p,\varepsilon}$ in Ω^ε, Ω^ε_m, Ω^ε_f and Ω^ε_p. The electrodes act as a sink with an imposed ambient temperature, so a Dirichlet boundary condition is imposed on the bottom surfaces $\Gamma^{0,p}$ of the pillars, cf Figure 11. The field θ of the difference of temperature to the ambient temperature is solution to the stationary equation $-\operatorname{div}(\mathbf{a}^\varepsilon \nabla\theta) = r$ in Ω^ε, with $\theta = 0$ on $\Gamma^{0,p}$, and $(\mathbf{a}^\varepsilon\nabla\theta)\cdot\mathbf{n} = 0$ on $\partial\Omega^\varepsilon - \Gamma^{0,p}$, where $\partial\Omega^\varepsilon$ representing the boundary of Ω^ε.

B. Features of the problem

The mathematical properties of the problem are derived by taking into account the special features of the problem: i.e. the characteristics of the geometry and of the coefficients with respect to the small parameter ε. Here, we do not report the mathematical derivation, but simply express the physical assumptions. The connections of the thin micromirrors to the frame are through thin and narrow beams whose effect is equivalent to a connection by low conductivity components. It results in a possibly large temperature variation in the mirrors and suspending beams of the range of ε^{-1}. Regarding the pillars, their small section compared to the surface of the mirrors and the electrodes yields a difference of temperature of the order $O(1)$ between their bottom and top ends which requires a temperature variation in the pillar direction to be in the range of ε^{-1}.

C. Two-scale convergence

Following [1], the two-scale transform operator T maps the physical periodic domain $\Omega^\varepsilon = \Omega^\varepsilon_m \cup \Omega^\varepsilon_f \cup \Omega^\varepsilon_p$ into a two-scale domain $\omega \times \Omega^1$, see Figure 12. The microscopic cell $\Omega^1 = \Omega^1_m \cup \Omega^1_f \cup \Omega^1_p \subset \mathbb{R}^3$ is deduced from any cell Ω^ε_i of the array centered at x^c_i by a translation and a dilation: $\Omega^1 = \{x^1 = (x - x^c_i)/\varepsilon \mid x \in \Omega^\varepsilon_i\}$. The macroscopic domain $\omega \subset \mathbb{R}$ is a segment in the direction x_2 having the length of the array and passing through the centers x^c_i. It is used for refereing to the cells. Precisely, the transformation T is applied to any

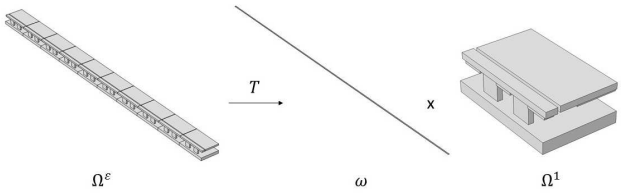

Fig. 12: Ω^ε transformed into $\omega \times \Omega^1$

function w defined on Ω^ε by $(Tw)(x^0, x^1) = w(x^c_i + \varepsilon x^1)$ for any x^0 in a cell $\omega \cap \Omega^\varepsilon_i$ and any $x^1 \in \Omega^1$. We also define the operator B mapping functions defined on $\omega \times \Omega^1$ to functions defined on Ω^ε by $Bv(x) = v\,(x_2, (x - x^c_i)/\varepsilon)$ for any $x \in \Omega$.

We assume that there exists a main temperature field θ^0 and its corrector θ^1 such that,

$$\oint_{\Omega^\varepsilon} \theta B(v)\ \mathrm{d}x = \oint_{\omega\times\Omega^1} (\theta^0 + \varepsilon\theta^1)v\ \mathrm{d}x^0\mathrm{d}x^1 + \varepsilon O(\varepsilon).$$

From a priori estimates we prove the approximations and equalities: $T\,(\varepsilon\partial_{x_\alpha}\theta) = \partial_{x^1_\alpha}\theta^0 + O_w(\varepsilon)$, $T(\partial_{x_3}\theta) = \partial_{x^1_3}\theta^1 + O_w(\varepsilon)$, $\partial_{x^1_3}\theta^0 = 0$ in $\omega \times \Omega^1_m$, $T(\partial_{x_1}\theta) = \partial_{x^1_1}\theta^1 + O_w(\varepsilon)$,

$T(\partial_{x_2}\theta) = \partial_{x_2^0}\theta^0 + \partial_{x_2^1}\theta^1 + O_w(\varepsilon)$, $T(\partial_{x_3}\theta) = \partial_{x_3^1}\theta^1 + O_w(\varepsilon)$, $\nabla_{x^1}\theta^0 = 0$, θ^1 is Ω_f^1-periodic in x_2^1 in $\omega \times \Omega_f^1$ and $T(\partial_{x_\alpha}\theta) = \partial_{x_\alpha^1}\theta^1 + O_w(\varepsilon)$, $T(\varepsilon\partial_{x_3}\theta) = \partial_{x_3^1}\theta + O_w(\varepsilon)$, $\partial_{x_1^1}\theta^0 = \partial_{x_2^1}\theta^0 = 0$ in $\omega \times \Omega_p^1$, and $\alpha \in \{1, 2\}$. The notation $O_w(\varepsilon)$ refers to any weakly vanishing function in the L^2-norm.

D. The Homogenized model

The previous approximations and equalities are plugged in the weak formulation which yields, after some steps, the two-scale model of the MMA. Since the matrix of diffusion is Ω^ε-periodic it has the form $\mathbf{a}^\varepsilon = T(\mathbf{a})$ where $\mathbf{a}(x^1)$ is the matrix of diffusion defined in the reference cell Ω^1. The temperature in the frame $\overline{\theta}^0(x^0) = \theta^0$ in $\omega \times \Omega_f^1$ is extended to the whole array $\omega \times \Omega^1$. The differences $\theta_m^0 = \theta^0 - \overline{\theta}^0$ in $\omega \times \Omega_m^1$ and $\theta_p^0 = \theta^0 - \overline{\theta}^0$ in $\omega \times \Omega_p^1$ satisfy the boundary conditions $\theta_m^0 = 0$ on $\partial\Omega_m^1 \cap \partial\Omega_f^1$ and $\theta_p^0 = 0$ on $\partial\Omega_p^1 \cap \partial\Omega_f^1$. The other equations satisfied by θ_m^0 in each mirror are discoupled from the other parts,

$$\begin{cases} -div_{\overline{x}^1}(\mathbf{a}^m \nabla_{\overline{x}^1}\theta_m^0) = r^{m,0} \text{ in } \omega \times \Omega_m^1, \\ (\mathbf{a}^m \nabla_{\overline{x}^1}\theta_m^0)^T \overline{n} = 0 \text{ on } \partial\Omega_m^1 / \partial\Omega_f^1, \end{cases} \tag{6}$$

where $\overline{x}^1 = (x_1^1, x_2^1)$, $\overline{n} = (n_1, n_2)$, $a_{\alpha\beta}^m = a_{\alpha\beta} - a_{3\beta}a_{\alpha 3}/a_{33}$ is the effective thermal conductivity of the mirror and $r^{m,0}$ is the effective internal heat source. In the pillars, $\theta_p^0 = \overline{\theta}^0 \theta_p'$ where the auxiliary function θ_p' is solution to the one-dimensional boundary value problem,

$$\begin{cases} -\partial_{x_3^1}(a^p \partial_{x_3^1}\theta_p') = 0 \text{ in } \Omega_p^1, \\ \theta_p' = 0 \text{ on } \partial\Omega_p^1 \cap \partial\Omega_f^1, \\ \theta_p' = -1 \text{ on } \Gamma^{0,p}, \end{cases} \tag{7}$$

where $a^p = \sum_{i,j=1}^3 L_i^p a_{ij} L_j^p$ and the vector $L^p = -\begin{pmatrix} a_{11} & a_{21} & 0 \\ a_{12} & a_{22} & 0 \\ 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} a_{31} \\ a_{32} \\ 1 \end{pmatrix}$. The temperature $\overline{\theta}^0$ in the frame solves

$$\begin{cases} -\partial_{x^0}(a^f \partial_{x^0}\overline{\theta}^0) = r^{f,0}, \\ -R_m^f \oint_{\partial\Omega_m^1} (\mathbf{a}^m \nabla_{x^1}\theta_m^0)^T n \, dx^1, \\ -R_m^p \overline{\theta}^0 \oint_{\partial\Omega_p^1} a^p \partial_{x_3^1}\theta_p' \, dx^1 \text{ in } \omega, \\ \overline{\theta}^0 = 0 \text{ on } \partial\omega, \end{cases} \tag{8}$$

with the effective thermal coefficient $a^f = \sum_{i,j=1}^3 (\delta_{i,2} + L_i^f) a_{ij} (\delta_{j,2} + L_j^f)$, $\delta_{i,j}$ the Kronecker delta symbol, and the vector $L^f = \nabla_{x^1} u$. The auxiliary function u is solution to $-\operatorname{div}_{x^1}(\mathbf{a}\nabla_{x^1} u) = -\operatorname{div}(e_2 \mathbf{a})$ and Ω_f^1-periodicity conditions where $e_2 = (0, 1, 0)$.

In the implementation, the temperature θ_m^0 and θ_p' in the mirror and in the pillars are computed by solving Equations (6) and (7). Then, the heat fluxes $\mathbf{a}^m \nabla_{x^1}\theta_m^0$ and $a^p \partial_{x_3^1}\theta_p'$ are used as sources in Equation (8) of the temperature $\overline{\theta}^0$ in the frame. With this method, the microscopic problems are solved cell by cell which reduces dramatically the memory requirement.

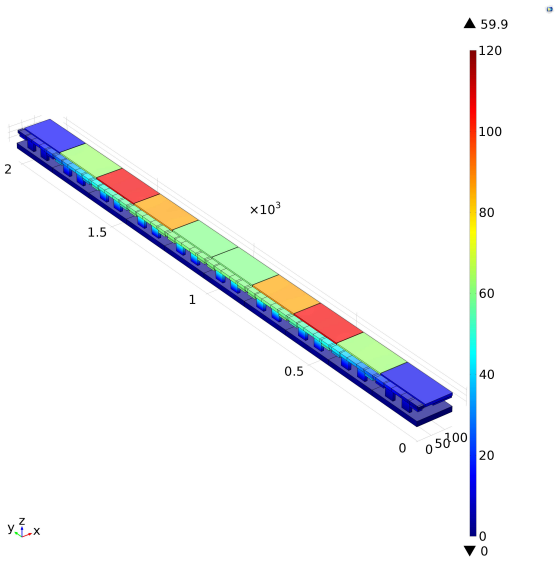

Fig. 13: Result of the homogenized model of the 1x10 MMA simulated in COMSOL. The heat loads is given as [0,5e9,1e10,5e9,0,0,5e9,1e10,5e9,0].

The homogenized model has been implemented for a 1×10 mirror array with a heat source oscillating along the x_2-direction as a sine function. The distribution of temperature is reported in Figure 13. In terms of performances, the estimate of the gain in terms of computational time is not yet precise, but it is more than twenty times for this case and increases rapidly when the array size increases.

VI. Conclusions

This paper presents recent advances in the development of `MEMSALab` including: the extension-combination method, a user language for the specification of proofs, extension operators and their combination and the key points of the construction and implementation of an asymptotic model of the stationary heat equation in a micro-mirror array. The next step will be to complete the extension-combination framework by integrating the anti-patterns and the parametrisations and to develop a library of extensions for generating a family of asymptotic models of MOEMS arrays for astrophysics.

References

[1] 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.

[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] Bin Yang, Walid Belkhir, and Michel Lenczner. Computer-aided derivation of multi-scale models: A rewriting framework. *International Journal for Multiscale Computational Engineering.*, 12(2):91–114, 2014.

[4] M.D. Canonica, F. Zamkotsian, P. Lanzoni, W. Noell, and N. De Rooij. The two-dimensional array of 2048 tilting micromirrors for astronomical spectroscopy. *J. of Micromechanics and Microengineering*, 23(5), 2013.

[5] Y. Bertot, P. Castéran, G. Huet, and C. Paulin-Mohring. *Interactive theorem proving and program development : Coq'Art : the calculus of inductive constructions*. Springer, Berlin, New York, 2004.

[6] Einar B.J. and Christoph L. Theorem reuse by proof term transformation. In *Theorem Proving in Higher Order Logics*.